A Model-Theoretic Approach to Updating Logical Databases (Extended Abstract)

Marianne Winslett Wilkins[*]
Stanford University, Computer Science Dept.

Abstract. We show that it is reasonable to extend the concept of database updates to encompass databases with incomplete information. Our approach embeds the incomplete database and the updates in the language of first-order logic, which we believe has strong advantages over relational tables and traditional data manipulation languages in the incomplete information situation. We present semantics for our update operators, and also provide an efficient algorithm to perform the operations.

1. Introduction

Much attention has been paid to the problem of answering queries in databases containing *null values*, or attribute values that are known to lie in a certain domain but whose value is currently unknown (see e.g. [Imielinski 84], [Reiter 84]). Progress on this front has encouraged research into the problem of updating such databases; as one group of researchers aptly points out [Abiteboul 85], the problem of query answering presupposes the ability to enter incomplete information into the database, and, with any luck, to remove uncertainties when more information becomes available.

Among recent work, this paper has ties to that of Abiteboul and Grahne [Abiteboul 85], who investigate the problem of updates on several varieties (with varying representational power) of *tables*, or relations containing null values and auxiliary constraints other than integrity constraints. They propose a definition for simple updates as set operations on the set of possible complete-information databases represented by two tables, and investigate the relationship between table type and ability to represent the result of an update correctly and completely. They do not consider updates with joins or disjunctions in selection clauses, comparisons between attribute values, or selection clauses referencing tuples other than the tuple being updated. Their conclusion was that only the most powerful and complex version of tables was able to fully support their update operators.

The work presented in this paper is perhaps most similar to that of Fagin et al [Fagin 83, Fagin 84], differing chiefly in the definitions of the meaning of updates and in the inclusion of a constructive algorithm for update computation. We base the semantics of updates on the contents of the models of the theory being updated; Fagin et al lend more importance to the particular formulas currently in the theory, producing a more syntactically oriented approach. The effect of an update in our paradigm is independent of the choice of formulas (other than schema and integrity constraints) used to represent that set of models. Another difference concerns our identification of two levels of formulas in a theory—axioms and non-axioms—and the provision of very different algorithmic manipulations for the two types of formulas during an update.

Reiter [84, 84b] sets forth a proof-theoretic, first-order logic framework for the null value and disjunctive information problems. (Disjunctive information occurs when one knows that one or more of a set of tuples holds true, without knowing which one.) Within this framework one may easily represent many, though not all, of the pieces of information typically encountered when dealing with missing information. Given a relational database, Reiter constructs a *relational theory* whose model corresponds to the world represented by the database. For our purposes here, the advantages of Reiter's framework are four-fold: it allows us to formalize our assumptions about the mechanics of a query and update processor; it allows a clean formalization of incomplete information; it allows us to define the meanings of query and update operators without recourse to intuition or common knowledge; and it frees us from implicit or explicit consideration of implementation issues, by not forcing incomplete information into a tabular for-

[*] AT&T Bell Laboratories Scholar

mat. By framing the update question in this paradigm, we will also gain insights into the more general problem of updating general logical theories.

In the remainder of this paper, we will set forth a simple update capability that covers many useful types of updates in what we call extended relational theories. Extended relational theories, presented in Section 2, are an extension to Reiter's theories for disjunctive information in which predicate constants may appear in formulas in the theory for the database and in which formulas other than simple disjunctions may appear, thus allowing a broader class of models for the theories. In Section 3.1 we set forth a simple data manipulation language, LDML, for extended relational theories, and give model-theoretic definitions of the meaning of LDML updates in Section 3.2. Sections 3.3 and 3.5 present an algorithm, GUA, that implements these semantics. The algorithm is correct [Wilkins 86] in the sense that the alternative worlds produced by updates under this algorithm are the same as those produced by updating each alternative world individually. The algorithm can be extended to cover the case where null values appear in the theory as Skolem constants, in which case the theory may have an infinite set of models. In Section 3.4 we present necessary and sufficient conditions for two LDML updates to be equivalent when applied to any extended relational theory. Finally, Section 3.6 discusses the computational complexity of GUA.

2. Extended Relational Theories

We now give a formal presentation of our extension to Reiter's theories, called *extended relational theories.* Unlike most formalizations of incomplete information, our extended relational theories will be sufficiently powerful to represent any set of relational databases all having the same schema and integrity constraints. The language $\mathcal{L}$ for the theories contains the following strings of symbols:

1. An infinite set of variables (to be used in the axioms of the extended relational theory).
2. A set of constants, possibly empty and possibly infinite. These represent the elements in the domains of database attributes.
3. A finite set of predicates of arity 1 or more, including '='. These represent the attributes and relations of the database.
4. Punctuation symbols '(', ')', and ','.
5. Logical connectives, quantifiers, and truth values: $\wedge, \vee, \neg, \leftrightarrow, \forall, \exists, \mathrm{T}, \mathrm{F}$.
6. An infinite set of 0-ary predicates (predicate constants).

For a given theory $\mathcal{T}$ over this language, $\mathcal{T}$ is an *extended relational theory* if $\mathcal{T}$ has exactly the following wffs:

1. *Unique Name Axioms:* For each pair of constants c_1, c_2 in $\mathcal{L}$, $\mathcal{T}$ contains the unique name axiom $\neg(c_1 = c_2)$.
2. *Equality Axioms:* $\mathcal{T}$ contains the equality axioms for reflexivity, commutativity, and transitivity, and an axiom for substitution of equals for each predicate of arity 1 or more:

$$\forall x(x = x)$$
$$\forall x \forall y((x = y) \rightarrow (y = x))$$
$$\forall x \forall y \forall z(((x = y) \wedge (y = z)) \rightarrow (x = z))$$
$$\forall x_1 \cdots \forall x_n \forall y_1 \cdots \forall y_n((P(x_1, \ldots, x_n) \wedge (x_1 = y_1) \wedge \cdots \wedge (x_n = y_n)) \rightarrow P(y_1, \ldots, y_n)),$$

for all predicates P of arity n.

3. *Completion Axioms:* To implement a version of the closed-world assumption so that we may prove certain ground atomic formulas to be false, we must have axioms stating that the only ground atomic formulas that may be true in a model are those explicitly given somewhere in $\mathcal{T}$. As our extended relational theories do not include any axioms to generate ground atomic formulas via inference, this means that any ground atomic formula not appearing in $\mathcal{T}$ should be false in all models of $\mathcal{T}$. (A different formulation of these axioms can be used to allow for ground atomic formulas generated by inference rules.) More precisely, for each n-ary predicate P of $\mathcal{T}$, either $\mathcal{T}$ contains an axiom of the form

$$\forall x_1 \forall x_2 \ldots \forall x_n \neg P(x_1, x_2, \cdots, x_n),$$

or else for some nonempty set of constants $c_{11}, c_{12}, \ldots, c_{nm}$, $\mathcal{T}$ contains exactly one axiom of the form

$$\begin{aligned}\forall x_1 \forall x_2 \ldots \forall x_n (P(x_1, x_2, \cdots, x_n) \rightarrow \\ ((x_1 = c_{11} \wedge x_2 = c_{12} \wedge \ldots \wedge x_n = c_{1m}) \vee \\ (x_1 = c_{21} \wedge x_2 = c_{22} \wedge \ldots \wedge x_n = c_{2m}) \vee \\ \cdots \quad \vee \\ (x_1 = c_{n1} \wedge x_2 = c_{n2} \wedge \ldots \wedge x_n = c_{nm})))\end{aligned}$$

Further, $(x_1 = c_{i1} \wedge x_2 = c_{i2} \wedge \ldots \wedge x_n = c_{im})$ is a disjunct of the axiom iff $P(c_{i1}, c_{i2}, \ldots, c_{in})$ appears elsewhere in $\mathcal{T}$.

Note that the completion axioms of $\mathcal{T}$ may be derived mechanically from the rest of $\mathcal{T}$.

4. *Non-Axiomatic Section:* The non-axiomatic formulas of $\mathcal{T}$ may be any wffs of $\mathcal{L}$ that do not contain variables. ◇

A discussion of extended relational theories with type axioms (an encoding of the database schema) and dependency axioms is postponed to Section 3.5, because the complications introduced by those axioms are orthogonal to the other issues in updating extended relational theories.

In an implementation of extended relational theories, we would not actually store any of these axioms. Rather, the axioms formalize our intuitions about the behavior of a query and update processor operating on the non-axiomatic part of the database. For example, PROLOG is a query processor that shares our unique name axioms, but has an entirely different closed-world assumption.

Definition. An *alternative world* of a theory $\mathcal{T}$ is a set S of truth valuations for all the ground atomic formulas of $\mathcal{T}$ of arity 1 or more, such that S holds for some model of $\mathcal{T}$. Intuitively, an alternative world is a snapshot of the tuples of a complete-information relational database. The alternative worlds of an extended relational theory look like a set of ordinary relational databases all having the same schema and axioms.

With the inclusion of predicate constants in $\mathcal{L}$ (as a convenience feature that makes updates easier to perform) we depart from Reiter's paradigm. Because predicate constants are "invisible" in alternative worlds, there may not be a one-to-one correspondence between the models of a relational theory and its alternative worlds, as two models may give the same truth valuation to all ground atomic formulas except some predicate constants, and still represent the same alternative world.

3. A Logical Data Manipulation Language (LDML) For Simple Updates

We now present a data manipulation language based on first-order logic, called LDML (Logical Data Manipulation Language). In this section we will consider the use of LDML for the simplest types of updates, which we call *ground updates.* The examples given will all be rather abstract; however, traditional data manipulation languages such as SQL and INGRES may be embedded in LDML.

3.1. LDML Syntax

Let $\mathcal{L}'$ be a language containing all the elements of $\mathcal{L}$ except its predicate constants, variables, and the equality predicate. Let ϕ and ω be wffs over $\mathcal{L}'$, and let t be a ground atomic formula over $\mathcal{L}'$. Then LDML ground updates consist of the following four operations:

INSERT ω WHERE ϕ

DELETE t WHERE $\phi \wedge t$

MODIFY t TO BE ω WHERE $\phi \wedge t$

ASSERT ϕ

Examples. Suppose the database schema contains two relations, Orders(OrderNo, PartNo, Quan) and InStock(PartNo, Quan). Then the following are ground updates:

MODIFY Orders(700,32,9) TO BE Orders(700,32,1)
WHERE T $\wedge$ Orders(700,32,9)

DELETE Orders(700, 32, 9)
WHERE T $\wedge$ Orders(700,32,9)

INSERT Orders(800,32,1000) $\vee$ Orders(800,32,100)
WHERE T

INSERT F WHERE $\neg$InStock(32, 1)
INSERT $\neg$InStock(32, 1) WHERE T

3.2. LDML Semantics For Ground Updates

We define the semantics of an update operating on an extended relational theory $\mathcal{T}$ by its desired effect on the models of $\mathcal{T}$. In particular, the alternative worlds (models minus predicate constants) of the updated relational theory must be the same as those obtained by applying the update separately to each original alternative world. In database terms, this may be rephrased as follows: The database with incomplete information represents a (possibly infinite) set of alternative worlds, or complete-information relational databases, each different and each one possibly the real, unknown world. The correct answers to queries and updates are those obtained by storing a separate database for each alternative world and running query processing in parallel on each separate database, pooling the query results in a final step. A necessary and sufficient guarantee of correctness for any more efficient and practical method of query and update processing is that it produce the same results for queries and updates as the parallel computation method. Equivalently, we require that the diagram below be commutative: both paths from upper-left-hand corner to lower-right-hand corner must produce the same result.

$$\begin{tikzcd}
\mathcal{T} \arrow[r, "\text{has alternative world}"] \arrow[d, "\text{update}"'] & \mathcal{A} \arrow[d, "\text{update}"] \\
\mathcal{T}' \arrow[r, "\text{has alternative world}"] & \mathcal{A}'
\end{tikzcd}$$

The general criteria guiding our choice of semantics are, first, that the semantics agree with traditional semantics in the case where the update request is to insert or delete a single ground atomic formula, or to modify one ground atomic formula to be another. Second, an update is to represent the *most exact and most recent state of knowledge obtainable* about the ground atomic formulas that the update modifies, inserts, or deletes; and the update is to *override all previous information* about these ground atomic formulas. These criteria have a syntactic component: one should not necessarily expect two updates with logically equivalent ω s to produce the same results. For example, the result of inserting the truth value T should be different from inserting $g \vee \neg g$; one update reports no change in the information available about g, and the other update reports that the truth valuation of g is now unknown.

We now present formal definitions of the semantics of ground updates. As the selection parameters ϕ and t for a ground update will evaluate to the same truth values in all models representing the same alternative world (because ϕ and t cannot contain predicate constants), without loss of generality we can define the semantics of INSERT, DELETE, MODIFY, and ASSERT in terms of their effect on models of an extended relational theory, rather than in terms of their effect on alternative worlds. Let B be a ground update, and let $\mathcal{M}$ be a model of the extended relational theory $\mathcal{T}$. Define $\mathcal{M}^{\dagger}$ to be the set of models produced by applying B to $\mathcal{M}$ as follows:

ASSERT ϕ: If ϕ is false in $\mathcal{M}$, then $\mathcal{M}^{\dagger}$ is the empty set; otherwise, $\mathcal{M}^{\dagger}$ contains exactly $\mathcal{M}$.

INSERT ω WHERE ϕ: If ϕ is false in $\mathcal{M}$, $\mathcal{M}^{\dagger}$ contains one model, $\mathcal{M}$. Otherwise, $\mathcal{M}^{\dagger}$ contains exactly every model $\mathcal{M}^*$ such that

(1) $\mathcal{M}^*$ agrees with $\mathcal{M}$ on the truth values of all ground atomic formulas except possibly those in ω; and

(2) ω is true in $\mathcal{M}^*$.

DELETE t WHERE $\phi \wedge t$: If $\phi \wedge t$ is false in $\mathcal{M}$, then $\mathcal{M}^{\dagger}$ contains exactly $\mathcal{M}$. Otherwise, let $\mathcal{M}^*$ be the model that agrees with $\mathcal{M}$ on all ground atomic formulas except t, which is false in $\mathcal{M}^*$; $\mathcal{M}^{\dagger}$ contains exactly $\mathcal{M}^*$.

MODIFY t TO BE ω WHERE $\phi \wedge t$: If $\phi \wedge t$ is false in $\mathcal{M}$, then $\mathcal{M}^{\dagger}$ contains one model, $\mathcal{M}$. Otherwise, let $\mathcal{N}$ be the model created from $\mathcal{M}$ by assigning the truth value F to t. Then $\mathcal{M}^{\dagger}$ contains every model $\mathcal{M}^*$ such that

(1) $\mathcal{M}^*$ has the same truth valuations for all ground atomic formulas as $\mathcal{N}$ does, except possibly those in ω; and

(2) ω is true in $\mathcal{M}^*$.

Example. If we insert $a \vee b$ into $\mathcal{M}$, where a and b are ground atomic formulas, then three models are created: one where $a \wedge b$ is true, one where $a \wedge \neg b$ is true, and one where $\neg a \wedge b$ is true—regardless of whether a or b were true or false in $\mathcal{M}$ originally.

Note that DELETE is a special case of MODIFY and INSERT: DELETE t WHERE $\phi \wedge t$ is equivalent to MODIFY t TO BE $\neg t$ WHERE $\phi \wedge t$, and also equivalent to INSERT $\neg t$ WHERE $\phi \wedge t$. Similarly, ASSERT is a special case of INSERT: ASSERT ϕ is equivalent to INSERT F WHERE $\neg \phi$. MODIFY t TO BE ω WHERE $\phi \wedge t$ is defined as first setting t to be false in $\mathcal{M}$ and then inserting ω; however, this is not equivalent to first deleting t with DELETE and then inserting some form of ω with INSERT, because the selection clause of INSERT cannot in general pick out exactly those models where $\phi \wedge t$ was true before the deletion. (For example, consider two extended relational theories $\mathcal{T}_1$ and $\mathcal{T}_2$ with non-axiomatic sections $\mathrm{T} \vee t$ and t, respectively; and the update MODIFY t TO BE a WHERE $\mathrm{T} \wedge t$. Once t is deleted from $\mathcal{T}_1$ and $\mathcal{T}_2$, the extended relational theories have only one alternative world between them, and are indistinguishable. But the correct insertion to finish the update for $\mathcal{T}_1$ is INSERT $\mathrm{T} \vee a$ WHERE T for $\mathcal{T}_1$ and INSERT a WHERE T for $\mathcal{T}_2$.)

The remarks at the beginning of this section on correctness of update algorithms may be summed up in the following definition:

Definition. The execution of a ground update B against an extended relational theory $\mathcal{T}$ to produce a new theory $\mathcal{T}'$ is correct and complete iff $\mathcal{T}'$ is an extended relational theory and the alternative worlds of $\mathcal{T}'$ are exactly those alternative worlds represented by the union of the models in the $\mathcal{M}^{\dagger}$ sets.

Definition. A *branching update* occurs when some $\mathcal{M}^{\dagger}$ contains more than one model. In such a case the models of $\mathcal{T}$ are said to branch, in that a model $\mathcal{M}$ before the update may map into more than one model and alternative world after the update. Intuitively, an update may cause branching when ω contains the logical operation '$\vee$', as with the ground update INSERT Orders(100, 32, 1) $\vee$ Orders(100, 32, 7) WHERE T.

Branching updates are used to introduce incomplete information into the extended relational theory. ASSERT is the usual method for removing incomplete information when more exact knowledge is obtained.

3.3. An Algorithm for LDML Ground Updates: GUA

Recall that DELETE is a special case of INSERT and MODIFY, and that ASSERT is a special case of INSERT; it suffices to give an algorithm for performing INSERT and MODIFY updates. First recall their syntax:

INSERT ω WHERE ϕ

MODIFY t TO BE ω WHERE $\phi \land t$

We have semantics that describe the effect of an update on the *models* of a theory; the semantics gives no hints whatsoever on how to translate that effect into changes in the extended relational theory. For INSERT ω WHERE ϕ, we cannot do anything so simple as to add $\phi \rightarrow \omega$ to $\mathcal{T}$, because ω probably contradicts the rest of $\mathcal{T}$. For example, if $\mathcal{T}$ contains $\neg a \land \neg b$, then INSERT $a \lor b$ WHERE T should not be interpreted as a request to add $\mathrm{T} \rightarrow a \lor b$ to $\mathcal{T}$! Similarly, for MODIFY, we cannot bodily replace occurrences of t with something like $(\phi \rightarrow \omega) \land (\neg \phi \rightarrow t)$; consider the effect of MODIFY a TO BE $b \lor c$ WHERE T on the non-axiomatic section $a \land \neg b \land (\neg c \lor d)$. Any algorithm for ground updates must be much more sophisticated than these attempts.

Our ground update algorithm GUA may be summarized as follows: *For each ground atomic formula f that appears in t or ω, define a new predicate constant p_f so that p_f is true in exactly those alternative worlds and models where f should be true after the update, and add these definitions to $\mathcal{T}$.* (These predicate constants are not visible externally, i.e., they may not appear in any query posed to the database.) *Then switch all occurrences of f and p_f in $\mathcal{T}$.*

Before a more formal presentation of the algorithm, let us examine its workings in a simple abstract example of a non-branching update. This example contains all the essential elements of the algorithm, and illustrates the principles underlying the algorithm. The reader should obtain a clear understanding of this example before examining algorithm GUA. A similar example for branching updates is given after the presentation of GUA.

Suppose the database schema contains a single relation with at most tuples a and b (such as Orders (700,34,10) and Orders(701,35,10)), and that we have the following two models and alternative worlds:

Model 1: a, b

Model 2: a

Ignoring the axioms for this database, the non-axiomatic section of the extended relational theory for this database must be logically equivalent to

a, $a \lor b$.

Suppose a user presents the following update:

MODIFY a TO BE a' WHERE $b \land a$.

To perform this update, we define new predicate constants p_a and $p_{a'}$ so that p_a is true only in those models of the theory where a should be true after the update, and that $p_{a'}$ is true only where a' should be true after the update. In other words, the new models should be:

Model 1: a, b, $p_{a'}$

Model 2: a, p_a

How do we define these new predicate constants? For each type of update, we use a different formula; the formula for MODIFY will be presented later in this section. For now we can intuitively say that p_a should be true in an alternative world iff a is true there now and we are not going to make a false in that world:

$p_a \leftrightarrow (a \land \neg b)$.

Similarly, $p_{a'}$ should appear in any world where the selection clause of the update is true:

$p_{a'} \leftrightarrow (a \land b)$.

We add these two formulas to the non-axiomatic section of the database, resulting in the desired models:

a, $a \lor b$, $p_a \leftrightarrow (a \land \neg b)$, $p_{a'} \leftrightarrow (a \land b)$.

In the final step of the update, we simultaneously replace all occurrences of p_a by a, and vice versa; and we also switch $p_{a'}$ and a'. The new theory is

p_a, $p_a \lor b$, $a \leftrightarrow (p_a \land \neg b)$, $a' \leftrightarrow (p_a \land b)$,

which (if we juggle the axioms appropriately) has the desired models:

Model 1: p_a, b, a'

Model 2: p_a, a

We now present the ground update algorithm for INSERT, DELETE, and MODIFY in full detail. This first version of the algorithm shows how to perform updates on an extended relational theory without type and dependency axioms; the procedures for use with extended relational theories having those axioms will be given later.

Ground Update Algorithm (GUA)

Input. A ground INSERT or MODIFY update B in LDML and an extended relational theory $\mathcal{T}$ without type and dependency axioms. (Express the deletion DELETE t WHERE $\phi \wedge t$ as the modification MODIFY t TO BE $\neg t$ WHERE $\phi \wedge t$, and express the assertion ASSERT ϕ as INSERT T WHERE $\neg \phi$.)

Output. $\mathcal{T}'$, an updated version of $\mathcal{T}$.

Procedure. A sequence of four steps:

Step 1. Control branching. For each distinct ground atomic formula f of ω, let there be two new predicate constants, which we will call p_f and b_f. For a MODIFY update, if the ground atomic formula of t does not appear in ω, then create one additional new predicate constant, which we will call p_t.

Example. For INSERT $a \vee b$ WHERE c, the new predicate constants are p_a, p_b, b_a, and b_b.

Definition. Given an INSERT or MODIFY ground update B, ω' is formed from ω of B by replacing each ground atomic formula f of ω by its new predicate constant b_f.

Examples. For INSERT $a \vee b$ WHERE c, ω' is $b_a \vee b_b$. For INSERT $(a \vee b) \wedge (c \vee d)$, ω' is $(b_a \vee b_b) \wedge (b_c \vee b_d)$.

Add $\phi \rightarrow \omega'$ (for INSERT) or $(\phi \wedge t) \rightarrow \omega'$ (for MODIFY) to the non-axiomatic section of $\mathcal{T}$, creating $\mathcal{T}'$. (The formula ω' governs the branching of models caused by B.)

Step 2. Add to completion axioms. For any ground atomic formula f appearing in ω, t, or ϕ but not in $\mathcal{T}$, add f to the completion axiom for its database predicate, and add $\neg f$ to the non-axiomatic section of $\mathcal{T}'$.

For example, if the update is INSERT Orders(700, 32, 9) $\vee$ Orders(700, 32, 8) WHERE T, and neither tuple previously appeared in the database, then both must be added to the completion axiom for Orders.

Step 3. Define the update. For each new predicate constant p_f, create a formula as follows to describe the desired effect of the update and add it to the non-axiomatic section of $\mathcal{T}'$:

- For INSERT, the new formula is

$$p_f \leftrightarrow ((\neg \phi \wedge f) \vee (\phi \wedge b_f)). \qquad (1)$$

- For MODIFY, if f appears in ω (as does a in MODIFY b TO BE $a \vee b$) then the new formula is

$$p_f \leftrightarrow ((\neg(\phi \wedge t) \wedge f) \vee (\phi \wedge t \wedge b_f)). \qquad (2)$$

If f appears in t but not in ω (as does a in MODIFY a TO BE $b \vee c$) then the new formula is

$$p_f \leftrightarrow (\neg \phi \wedge f). \qquad (3)$$

Formulas (1) through (3) are an encoding in first-order logic of the semantics of INSERT, MODIFY, and DELETE, as presented in Section 3.2. Intuitively, p_f is true in a model $\mathcal{M}$ iff f should be true in $\mathcal{M}$ after the update.

Step 4. Update. For each new predicate constant p_f, switch all occurrences of f and p_f in the non-axiomatic section of $\mathcal{T}'$. Then the models of $\mathcal{T}'$ represent exactly the alternative worlds that B is defined to produce from $\mathcal{T}$. ◇

Examples. We present an abstract example of a branching update, again restricting our attention to the non-axiomatic section of an extended relational theory $\mathcal{T}$. Suppose the database schema for $\mathcal{T}$ contains a single relation with at most tuples a and b, and that we have the following two alternative worlds:

Model 1: a, b

Model 2: a

Ignoring the axioms for this database, the non-axiomatic section of the logic theory for the database must be logically equivalent to

a, $a \vee b$.

Suppose a user presents the following update:

MODIFY a TO BE $c \vee a$ WHERE $b \wedge a$.

In Step 1, we take ω (i.e., $c \vee a$) and form ω' (i.e., $b_c \vee b_a$) with new predicate constants b_c and b_a. We add $(\phi \wedge t) \rightarrow \omega'$ (i.e., $(b \wedge a) \rightarrow (b_c \vee b_a)$) to the non-axiomatic section of $\mathcal{T}$, creating $\mathcal{T}'$.

In Step 2, if a, b, or c does not appear in the completion axioms of $\mathcal{T}'$, we add it there now. By the definition of an extended relational theory, a and b must already appear in those axioms; we simply add c to its completion axiom and add $\neg c$ to the non-axiomatic section of $\mathcal{T}'$.

In Step 3, we use equation (2) to describe the desired effect of the update on a and c, by defining new predicate constants p_a and p_c that are true in a model $\mathcal{M}$ of $\mathcal{T}'$ iff a and c, respectively, should be true in $\mathcal{M}$ after the update. The non-axiomatic section of $\mathcal{T}'$ is now

a, $a \vee b$

$(b \wedge a) \rightarrow (b_c \vee b_a)$

$\neg c$

$p_a \leftrightarrow ((\neg b \wedge a) \vee (a \wedge b \wedge b_a))$

$p_c \leftrightarrow ((\neg(b \wedge a) \wedge c) \vee (a \wedge b \wedge b_c))$.

The models of $\mathcal{T}'$ are:

Model 1: a, p_a

Model 2: a, b, b_c, p_c

Model 3: a, b, b_a, p_a

Model 4: a, b, b_c, b_a, p_c, p_a.

We are now ready for Step 4. We simultaneously replace each occurrence of p_a and p_c by a and c, respectively; and replace each occurrence of a and c by p_a and p_c, respectively. The new theory is

$$p_a,\ p_a \vee b,\ (b \wedge p_a) \rightarrow (b_c \vee b_a),\ \neg p_c$$
$$a \leftrightarrow ((\neg b \wedge p_a) \vee (p_a \wedge b \wedge b_a))$$
$$c \leftrightarrow ((\neg(b \wedge a) \wedge p_c) \vee (p_a \wedge b \wedge b_c))$$

which has the desired models, representing four alternative worlds:

Model 1: p_a, a

Model 2: p_a, b, b_c, c

Model 3: p_a, b, b_a, a

Model 4: p_a, b, b_c, b_a, c, a

The non-axiomatic section of $\mathcal{T}'$ can be simplified to the two wffs $a \vee b$ and $b \rightarrow (c \vee a)$.

Theorem 1. Given an extended relational theory $\mathcal{T}$ and a legal ground update B, algorithm GUA correctly and completely performs B. In particular,

(1) GUA produces a legal extended relational theory $\mathcal{T}'$;

(2) The alternative worlds of $\mathcal{T}'$ are the same as the alternative worlds produced by directly updating the models of $\mathcal{T}$. ◇

(Proofs of all theorems are presented elsewhere [Wilkins 86].)

3.4. Equivalence of Updates

We now turn to the question of when two updates may be considered equivalent. This is important as it will allow us to measure exactly the role that syntax plays in the semantics we have given for updates.

Definition. If B_1 and B_2 are two updates over a language $\mathcal{L}$, then B_1 and B_2 are *equivalent* if for every extended relational theory $\mathcal{T}$ over $\mathcal{L}$, B_1 applied to $\mathcal{T}$ produces the same set of alternative worlds as B_2 applied to $\mathcal{T}$.

First we show that it suffices to prove conditions on equivalence for INSERT updates, rather than considering ASSERT, INSERT, DELETE, and MODIFY separately.

Theorem 2. If B is a ground deletion of the form DELETE t WHERE ϕ, then B is equivalent to the modification MODIFY t TO BE $\neg t$ WHERE ϕ.

If B is a ground assertion of the form ASSERT ϕ, then B is equivalent to the insertion INSERT F WHERE $\neg\phi$.

If M_1, M_2, B_1, and B_2 are ground modifications of the form

M_1 = MODIFY t TO BE ω_1 WHERE ϕ,

M_2 = MODIFY t TO BE ω_2 WHERE ϕ,

B_1 = INSERT ω_1 WHERE $(\phi)^t_{\mathrm{T}} \wedge \neg t$,

B_2 = INSERT ω_2 WHERE $(\phi)^t_{\mathrm{T}} \wedge \neg t$,

then M_1 and M_2 are equivalent iff B_1 and B_2 are equivalent. ◇

We begin with simple sufficient criteria for equivalence:

Theorem 3. Let B_1 and B_2 be two INSERT ground updates over a language $\mathcal{L}$:

B_1 = INSERT ω_1 WHERE ϕ,

B_2 = INSERT ω_2 WHERE ϕ.

If ω_1 and ω_2 are logically equivalent and the same ground atomic formulas appear in ω_1 and ω_2, then B_1 is equivalent to B_2. ◇

To see that the criteria of Theorem 3 are sufficient but not necessary, consider the two equivalent updates INSERT q WHERE $p \wedge q$ and INSERT p WHERE $p \wedge q$. For necessary and sufficient criteria, we have Theorem 4, which can be summed up intuitively as follows: B_1 and B_2 are equivalent iff ω_1 and ω_2 are satisfied by the same sets of valuations, except that a ground atomic formula g may appear in ω_1 and not in ω_2 (or vice versa) as long as B_1 (resp. B_2) does not change the truth valuation of g.

Theorem 4. Let B_1 and B_2 be two INSERT ground updates over a language $\mathcal{L}$:

B_1 = INSERT ω_1 WHERE ϕ,

B_2 = INSERT ω_2 WHERE ϕ.

Let I be the set of ground atomic formulas appearing in both ω_1 and ω_2. Within any model of ω_1, let v_1 be the set of truth valuations for the ground atomic formulas of I in that model. Let V_1 be the set of valuations v_1 over all models of ω_1. Define v_2 and V_2 analogously for ω_2. Then B_1 and B_2 are equivalent iff

(1) $V_1 = V_2$; and

(2) if the ground atomic formula g appears in ω_1 but not in ω_2 then $(\omega_1 \rightarrow g) \wedge (\phi \rightarrow g)$ or $(\omega_1 \rightarrow \neg g) \wedge (\phi \rightarrow \neg g)$ is valid; and

(3) if g appears in ω_2 but not in ω_1 then $(\omega_2 \rightarrow g) \wedge (\phi \rightarrow g)$ or $(\omega_2 \rightarrow \neg g) \wedge (\phi \rightarrow \neg g)$ is valid. ◇

Examples. A *valuation* v for a wff α is a set of truth assignments to all the ground atomic formulas of α. If ω_1 is p and ω_2 is $p \vee \mathsf{T}$, then ω_2 is satisfied by a valuation that assigns F to p, while ω_1 is not; the two formulas do not satisfy condition (1) for equivalence. Therefore `INSERT` p `WHERE` T is not equivalent to `INSERT` $p \vee \mathsf{T}$ `WHERE` T; they differ on producing models where p is false. For updates `INSERT` p `WHERE` $p \wedge q$ and `INSERT` q `WHERE` $p \wedge q$, V_1 and V_2 are both the empty set, and Theorem 4 correctly predicts equivalence.

What conditions govern equivalence when the two updates have different selection clauses? Intuitively, if B_1 and B_2 are two equivalent ground updates with selection clauses ϕ_1 and ϕ_2, then B_1 must not make any changes in any model where ϕ_2 is false, and vice versa.

Theorem 5. Let B_1 and B_2 be two `INSERT` ground updates over a language $\mathcal{L}$:

B_1 = `INSERT` ω_1 `WHERE` ϕ_1,

B_2 = `INSERT` ω_2 `WHERE` ϕ_2.

Then B_1 and B_2 are equivalent iff

(1) `INSERT` ω_1 `WHERE` $\phi_1 \wedge \phi_2$ is equivalent to `INSERT` ω_2 `WHERE` $\phi_1 \wedge \phi_2$; and

(2) $(\phi_1 \wedge \neg \phi_2) \rightarrow \omega_1$ and $(\phi_2 \wedge \neg \phi_1) \rightarrow \omega_2$ are valid; and

(3) if $\phi_1 \wedge \neg \phi_2$ is satisfiable, then there exists exactly one valuation of ω_1 that makes ω_1 true; and if $\phi_2 \wedge \neg \phi_1$ is satisfiable, then there exists exactly one valuation of ω_2 that makes ω_2 true. ◇

Note that although syntax is important in updates, it does not play a role in the non-axiomatic sections of extended relational theories: if two extended relational theories have the same axioms, then they will have identical sets of alternative worlds after a series of updates iff the non-axiomatic sections of the two theories are logically equivalent.

3.5. Extended Relational Theories with Type and Dependency Axioms

Until now, we have considered extended relational theories without type and dependency axioms, because the complications introduced by those axioms are orthogonal to the other issues in updating extended relational theories. We now expand the definition of an extended relational theory as follows: Distinguish a particular set $\mathcal{A}$ of unary predicates of $\mathcal{T}$ as the *attributes* of $\mathcal{T}$. Then add two requirements to items 1–4 in the definition of an extended relational theory in Section 2:

5. *Type Axioms:* The type axioms encode the schema of the database in logic. For each n-ary predicate P not in $\mathcal{A}$, $\mathcal{T}$ contains exactly one axiom of the form

$$\forall x_1 \forall x_2 \cdots \forall x_n (P(x_1, x_2, \cdots, x_n) \rightarrow (A_1(x_1) \wedge A_2(x_2) \wedge \cdots \wedge A_n(x_n))),$$

where $A_1, A_2, \cdots, A_n$ are predicates in $\mathcal{A}$. Further, each predicate in $\mathcal{A}$ must appear in one or more type axioms, and the non-axiomatic section of $\mathcal{T}$ must always be such that removing the type and dependency axioms from $\mathcal{T}$ *does not change the models of* $\mathcal{T}$. The reasons for this restriction will become apparent when we consider the changes needed in algorithm GUA to handle type axioms.

6. *Dependency Axioms:* $\mathcal{T}$ may optionally contain a set of sentences (wffs with no unbound variables) not containing predicate constants, designated as dependency axioms. However, the non-axiomatic section of $\mathcal{T}$ must always be such that removing the type and dependency axioms from $\mathcal{T}$ *does not change the models of* $\mathcal{T}$. The reasons for this restriction will become apparent when we consider the changes needed in algorithm GUA to handle dependency axioms. ◇

The semantics of updates must be augmented to enforce items 5 and 6. There are a number of reasonable enforcement policies that may be adopted. For dependency axioms in ordinary databases, a dependency axiom violation is usually taken as a signal to repair the database, e.g., by adding additional tuples. In a database with incomplete information, such as an extended relational theory, the dependency axioms also serve the function of automatically weeding out alternative worlds that could not possibly be the actual world. What should be the policy of the extended relational theory update algorithms on this delicate interplay of functions? We choose to delegate this issue to a higher authority, such as a database administrator, and ourselves provide only a mechanism that uses type and dependency axioms to *weed out impossible alternative worlds.* If desired, an additional layer may be incorporated between the user and algorithm GUA (or directly into GUA) to modify update requests in order to save models that would otherwise be inadvertently removed. For example, if the user request is `INSERT` $R(a,b,c)$ `WHERE` T, then the type and dependency layer might automatically convert this to `INSERT` $R(a,b,c) \wedge A_1(a) \wedge A_2(b) \wedge A_3(c)$ `WHERE` T, where the three additional predicates are the attributes of R.

In any case, we modify the update semantics by adding one additional proviso to `INSERT`, `DELETE`, and

MODIFY: A model $\mathcal{M}^*$ produced by a ground update B from $\mathcal{M}$ must also satisfy the type and dependency axioms of $\mathcal{T}$; otherwise, $\mathcal{M}^*$ is removed from $\mathcal{M}^\dagger$.

In this discussion, the type, dependency, unique name, and equality axioms will be permanently fixed for each database schema. It is a simple matter to extend this to allow updates to the axioms such as adding new dependencies, constants, or relations.

For algorithm GUA to handle type and dependency axioms correctly, we need to change Step 2 slightly and add three additional steps at the end of the algorithm:

Step 2′. Add to completion axioms. In addition to the functions of Step 2, for every constant c_1 appearing as a value for attribute A in a ground atomic formula of ω , if c_1 is not listed in the completion axiom for A, then add c_1 to that completion axiom in $\mathcal{T}_1$, and add $\neg A(c_1)$ to the non-axiomatic section of $\mathcal{T}_1$.

For example, if the update is INSERT Orders(700, 32, 9) WHERE T, and there was no order 700 prior to the update, then "700" must be added to the completion axiom for the OrderNo attribute; and similarly for the quantity 9 and the part number 32.

Step 5. Instantiate the type axioms. Although $\mathcal{T}'$ now represents exactly the desired alternative worlds, $\mathcal{T}'$ may not yet be an extended relational theory. Recall that the models of an extended relational theory must not change if the type and dependency axioms are removed from the theory. To maintain this property after an update, the type axioms must be "instantiated" with the new ground atomic formulas in $\mathcal{T}'$ if $(\mathcal{T}' - \text{Dep} - \text{Ty}) \not\models \mathcal{T}'$. If we did not do this, then "illegal" alternative worlds could suddenly become legal again after an update, if the violation was removed by the update. To meet this requirement with a minimum of effort, let $P(c_1, c_2, \ldots, c_n)$ be a ground atomic formula appearing in $\mathcal{T}'$ but not in $\mathcal{T}$. Suppose that the type axiom for P in $\mathcal{T}$ is $\forall x_1 \forall x_2 \cdots \forall x_n (P(x_1, x_2, \ldots, x_n) \rightarrow (A_1(x_1) \wedge A_2(x_2) \wedge \cdots \wedge A_n(x_n)))$. Then if it is the case that

(1) $P(c_1, c_2, \ldots, c_n)$ is in ω and $A_i(c_i)$ is not a formula of $\mathcal{T}'$ for some i; or

(2) $A_i(c_i)$ appears in ω or t and $\omega \not\rightarrow A_i(c_i)$,

add the formula $P(c_1, c_2, \cdots, c_n) \rightarrow (A_1(c_1) \wedge A_2(c_2) \wedge \cdots \wedge A_n(c_n))$ to the non-axiomatic section of $\mathcal{T}'$, if it is not already present.

Step 6. Instantiate the dependency axioms. We consider universally quantified dependencies of a template form, such as a functional or relation-inclusion dependency:

$$\forall x_1 \cdots \forall x_n (\alpha \rightarrow \beta),$$

where α is a conjunction of atomic formulas g_1 through g_m, β is quantifier-free, and x_1 through x_n appear in α. In this case it suffices to instantiate the axiom for those ground atomic formulas that unify with g_i of α . More formally, for every set of constants $c_1, \ldots, c_n$ in $\mathcal{L}$ such that the ground atomic formula

$$((g_i)^{x_1 \ldots x_n}_{c_1 \ldots c_n})$$

appears in $\mathcal{T}'$ for all i, add

$$((\alpha \rightarrow \beta)^{x_1 \ldots x_n}_{c_1 \ldots c_n})$$

to the non-axiomatic section of $\mathcal{T}'$.

If we did not do this, then "illegal" alternative worlds could suddenly become legal again after an update, if the violation was removed by the update. For example, suppose that Q and P are one-place relations, and we have an inclusion dependency $\forall x (P(x) \rightarrow Q(x))$. Then whenever a new tuple is added to P, such as $P(a)$, the new formula $P(a) \rightarrow Q(a)$ should also be inserted unless it can be proved that $Q(a)$ already appears in all models where $P(a)$ is to appear. Similarly, if $Q(a)$ is deleted while $P(a)$ is still in the theory, then the new wff $P(a) \rightarrow Q(a)$ should be added to $\mathcal{T}'$.

Step 7. Add to completion axioms. Modify the non-axiomatic section and completion axioms of $\mathcal{T}'$ as follows: For any ground atomic formula f added to $\mathcal{T}'$ in Steps 5 or 6, add f to the completion axiom for its predicate, and add $\neg f$ to the non-axiomatic section of $\mathcal{T}'$. Also, for every constant c appearing as a value for attribute A in a ground atomic formula of $\mathcal{T}'$, if c is not listed in the completion axiom for A, then add c to that completion axiom in $\mathcal{T}'$, and add $\neg A(c)$ to the non-axiomatic section of $\mathcal{T}'$. $\diamond$

Theorem 1, on the correctness and completeness of GUA, still holds when we incorporate type and dependency axioms. For proofs, again the reader is referred elsewhere [Wilkins 86].

Let us now consider how type and dependency axioms interact with the results in section 3.4 on update equivalence. There is now a spurious way in which two updates B_1 and B_2 over $\mathcal{L}$ may be equivalent, caused by a certain relationship between $\mathcal{L}$, B_1 , and B_2 : B_1 and B_2 may be equivalent solely because certain alternative worlds produced by B_1 , say, and not by B_2 , violate the type axioms (i.e., schema) of *every possible extended relational theory over* $\mathcal{L}$. If this is the case, then augmenting $\mathcal{L}$ by a single one-place predicate will make B_1 and B_2 inequivalent. This is undesirable; if B_1 and B_2 are equivalent over $\mathcal{L}$, we would like them to be equivalent over all extensions of $\mathcal{L}$.

Example. If $\mathcal{L}$ contains one two-place predicate, P_1, and two one-place predicates A_1 and A_2, then INSERT F WHERE T is spuriously equivalent to INSERT $P_1(c_1, c_2) \wedge \neg A_1(c_1) \wedge \neg A_2(c_1)$ WHERE T.

To avoid these spurious equivalences, we will simply augment the definition of equivalence so that two updates over $\mathcal{L}$ are equivalent only if they are equivalent over $\mathcal{L}$ and all extensions of $\mathcal{L}$. With this condition in hand, Theorems 2 through 5 on update equivalence still hold for extended relational theories with type and dependency axioms.

3.6. Cost of Algorithm GUA

Let g be the number of instances of ground atomic formulas in the ground update B; and let R be the size of the predicate with the greatest number of distinct occurrences in the extended relational theory $\mathcal{T}$. If no dependency axioms are present, an optimized form of GUA runs in time $O(g\log(R))$ and increases the size of $\mathcal{T}$ by $O(g)$ worst case. This is not to say that an $O(g\log(R))$ implementation of updates is the best choice; rather, it is advisable to devote extra time to heuristics for minimizing the length of the formulas to be added to $\mathcal{T}$. Nonetheless, a worst-case time estimate for GUA is informative, as it tells us how much time must be devoted to the algorithm proper.

To obtain this estimate, all ground atomic formulas in the non-axiomatic section of $\mathcal{T}$ must appear in indices, with one index per predicate, so that lookup and insertion time is $O(\log(R))$. Predicate constants, however, are referenced through a single separate index. The renaming step (Step 4) is the bottleneck for GUA. To make renaming fast, we assume that the ground atomic formulas of the non-axiomatic section are stored only as pointers. In particular, we assume that all occurrences of a ground atomic formula or predicate constant in the non-axiomatic section of $\mathcal{T}$ are linked together in a list whose head is an index entry, so that renaming may be done rapidly. Additionally, the names of ground atomic formulas cannot be physically stored with the non-axiomatic wffs they appear in; however, the non-axiomatic wffs may contain pointers into a separate name space where names of ground atomic formulas are kept.

Finally, we assume that the schema is fixed, i.e., that the number of predicates is a constant.

The cost of Step 6 (dependency checking) depends entirely on the type of dependency axioms. We derive costs for the simplest types of dependencies here, which can be given optimized enforcement algorithms.

If the dependency axiom is a functional dependency, then the cost of Step 6 is $O(gR)$ worst case (when every updated tuple seems to conflict with every other tuple in its relation) and $O(g\log(R))$ best case (when no conflicts occur). If the dependency axiom is a relation-inclusion dependency, then the cost is also is $O(gR)$ worst case (when the removal of a tuple seems to invalidate every tuple in some other relation) and $O(g\log(R))$ best case (when no conflicts occur). The same cost functions hold for a multivalued dependency as well.

4. Summary and Conclusion

We have defined extended relational theories as extensions of Reiter's theories for disjunctive information. Formulas in the body of an extended relational theory may be any ground wffs, and may contain auxiliary predicate constants that are not part of the database schema, thereby increasing the representational power of Reiter's theories. Within this context, we set forth a simple data manipulation language, LDML, and give model-theoretic definitions of the meaning of LDML updates. We concentrate on the concept of a ground update, or an LDML update without variables; updates with variables can be reduced to the problem of performing a set of ground updates simultaneously. We present an algorithm for performing ground updates, and prove it correct in the sense that the alternative worlds produced by updates under this algorithm are the same as those produced by updating each alternative world individually. For a particular extended relational theory $\mathcal{T}$, this algorithm runs in time proportional to the product of the number of atomic formulas in the update request and the logarithm of the size of the predicate with the largest number of distinct ground atomic formulas in $\mathcal{T}$; this is the same asymptotic cost as for ordinary complete-information database updates.

We conclude that, first, one may extend the concept of a database update to databases with incomplete information in a natural way; second, that first-order logic is a fruitful paradigm for the investigation; and third, that one may construct an algorithm to perform these updates with a reasonable running time.

An important topic that we have not found room to discuss here is the simplification of extended relational theories, as they grow steadily longer under the update algorithms presented. This is a vital concern, since it is the possibility of simplification that makes the LDML algorithms more attractive than simply keeping an indexed history of past updates and recomputing the state of the theory on each new query. A heuristic algorithm for simplification will be a vital part of any implementation of these algorithms, and is at the core of the implementation coded by the author.

6. Acknowledgements

This paper would not have been possible without the aid of of Arthur Keller, the encouragement of Christos Papadimitriou, and the stylistic preferences of Moshe "$\forall\epsilon\exists\delta$" Vardi.

7. References

[Abiteboul 85] S. Abiteboul, G. Grahne, "Update Semantics for Incomplete Databases," *Proc. VLDB Conf.*, Stockholm, August 1985.

[Fagin 84] R. Fagin, G. M. Kuper, J. D. Ullman, and M. Y. Vardi, "Updating Logical Databases," *Proc. of the 3rd ACM PODS*, April 1984.

[Fagin 83] R. Fagin, J. D. Ullman, and M. Y. Vardi, "On the Semantics of Updates in Databases," *Proc. of the 2nd ACM PODS*, April 1983.

[Imielinski 84] T. Imielinski and W. Lipski, "Incomplete Information in Relational Databases," *Journal of the ACM*, **31**:4, October 1984.

[Reiter 84] R. Reiter, "Towards a Logical Reconstruction of Relational Database Theory," in M. Brodie, J. Myopoulos, and J. Schmidt (eds.) *On Conceptual Modelling*, Springer-Verlag, 1984.

[Reiter 84b] R. Reiter, "Logical Foundations for Database Theory," *Conference On Theory in Data Bases*, Benodet, France, July 1984.

[Vardi 85] M. Vardi, "Querying Logical Databases," *Proc. of the 4th ACM PODS*, March 1985.

[Wilkins 86] M. Wilkins, "A Model-Theoretic Approach to Updating Logical Databases," Computer Science Department Technical Report, Stanford University, January 1986.