# ON THE INTEGRITY OF TYPED FIRST ORDER DATA BASES

by

RAYMOND REITER

TECHNICAL REPORT 80-6

1980 APRIL

DEPARTMENT OF COMPUTER SCIENCE
THE UNIVERSITY OF BRITISH COLUMBIA
VANCOUVER, BRITISH COLUMBIA V6T 1W5

ABSTRACT

A typed first order data base is a set of first order formulae, each quantified variable of which is constrained to range over some type. Formally, a type is simply a distinguished monadic relation, or some Boolean combination of these. Assume that with each data base relation other than the types is associated an integrity constraint which specifies which types of individuals are permitted to fill the argument positions of that relation. The problem addressed in this paper is the detection of violations of these integrity constraints in the case of data base updates with universally quantified formulae. The basic approach is to first transform any such formula to its so-called reduced typed normal form, which is a suitably determined set of formulae whose conjunction turns out to be equivalent to the original formula. There are then simple criteria which, when applied to this normal form, determine whether that formula violates any of the argument typing integrity constraints.

--------------------

This work was supported by the National Science and Engineering Research Council of Canada through operating grant A 7642.

Key Words and Phrases

consistency, data bases, deductive information retrieval, first order logic, integrity, type constraints, type data base, typed formulae, typed normal form.

# ON THE INTEGRITY OF TYPED FIRST ORDER DATA BASES

by

Raymond Reiter
Department of Computer Science
University of British Columbia

## 1. INTRODUCTION

It is difficult to conceive of a naturally occuring relation which is unconstrained with respect to the kinds of individuals which may legitimately satisfy that relation[1]. Thus, in speaking about the relation "x is the husband of y" we all of us understand that x must be a male human, and y a female human. At best there is something peculiar about the statement "Mary is the husband of Susan", presumably because the individual "Mary" violates the universally accepted constraint that the first argument of the husband relation must be male.

This simple example illustrates what appears to be a universal characteristic of such argument constraints on relations and that is that each such constraint is itself either a simple unary relation, for example $MALE(\cdot)$ , or a Boolean combination of such simple unary relations, for example $[MALE \wedge HUMAN](\cdot)$. Given a suitable stock of such simple unary relations, it is now straightforward to formally represent the argument constraints of the husband relation as a first order formula:

$$(x\ y)[HUSBAND\text{-}OF(x,y) \supset MALE(x) \wedge HUMAN(x) \wedge FEMALE(y) \wedge HUMAN(y)] \qquad (1.1)$$

---

[1] The equality relation appears to be the only exception to this observation.

In this paper we shall view such formulae as integrity constraints of a particular kind; they specify the allowable arguments to a relation. Any attempt to update a data base with a fact which violates such integrity constraints, for example an attempted update with HUSBAND-OF (Mary, Susan), will be rejected. For the example at hand it is not difficult to see why the update must be rejected since to accept it is to accept, by (1.1), the fact MALE (Mary). Of course, in order that a data base detect the inconsistency of MALE (Mary) it must have available some facts about MALEs, Mary etc. At the very least, it must know $\sim$MALE (Mary) or, what is more likely, it has available the specific fact FEMALE (Mary) as well as the general fact $(x) \sim$ [MALE $(x) \wedge$ FEMALE $(x)$] from which $\sim$MALE (Mary) can be deduced. Accordingly, the entire data base must contain as a subcomponent a data base consisting of both specific and general facts about the unary relations which enter into the integrity constraints of the form (1.1). We refer to this sub-data base as the type data base.

In addition to this type data base, there will be information about the remaining relations. In a conventional relational data base [Date 1977] this information can be viewed as a set of ground atomic formulae in a first order theory, and the domains associated with a given relation $R$ are simply those unary relations which restrict the allowable arguments of $R$. In the deductive first order data bases of the kind treated in [Kellogg et al. 1978, Kowalski 1979, Minker 1978, Reiter 1978] general facts about data base relations are also allowed so that one is permitted to store, for example:

$$(x\ y)[\text{HUSBAND-OF}(x,y) \supset \text{WIFE-OF}(y,x)] \qquad (1.2)$$

Answers to queries are then obtained by a process of deduction from the first order data base. In [Minker 1978, Reiter 1977, 1978] the class of formulae permitted in a first order data base is generalized to admit typed variables so that, in the notation of [Reiter 1978] and of this paper, (1.2) would be represented by:

$$(x/MALE \wedge HUMAN)(y/FEMALE \wedge HUMAN)[HUSBAND\text{-}OF(x,y) \supset WIFE\text{-}OF(y,x)] \qquad (1.3)$$

Here the universally quantified variables x and y are restricted to range over instances of the unary relations (or types as we shall henceforth call them) MALE ∧ HUMAN and FEMALE ∧ HUMAN respectively.

For first order data bases containing general facts of the form (1.2) or (1.3) the enforcement of suitable relational argument typing is not as straightforward as it is in the case of conventional non deductive relational data bases. As an example, consider the integrity constraints:

$$\left.\begin{array}{l}(x\ y)[OFFSPRING(x,y) \supset HUMAN(x) \wedge HUMAN(y)] \\ (x\ y)[MOTHER(x,y) \supset HUMAN(x) \wedge FEMALE(x) \wedge HUMAN(y)] \\ (x\ y)[FATHER(x,y) \supset HUMAN(x) \wedge MALE(x) \wedge HUMAN(y)]\end{array}\right\} \qquad (1.4)$$

together with a type data base:

$$\left.\begin{array}{l}(x)[HUMAN(x) \supset MALE(x) \vee FEMALE(x)] \\ (x) \sim[MALE(x) \wedge FEMALE(x)]\end{array}\right\} \qquad (1.5)$$

Now consider an update of this kinship data base with the general fact:

$$(x/HUMAN)(y/HUMAN)[OFFSPRING(x,y) \supset MOTHER(y,x) \vee FATHER(y,x)] \qquad (1.6)$$

Should this update be accepted? One possible intuition (which we shall see turns out to be wrong) is that the variable y is constrained by the MOTHER relation to be FEMALE and by the FATHER relation to be MALE so the update should be rejected. Another possible intuition (which turns out to be right) holds that (1.6) is equivalent to the two formulae

$(x/\mathrm{HUMAN})(y/\mathrm{HUMAN} \wedge \mathrm{FEMALE})[\mathrm{OFFSPRING}(x,y) \supset \mathrm{MOTHER}(y,x)]$

$(x/\mathrm{HUMAN})(y/\mathrm{HUMAN} \wedge \mathrm{MALE})[\mathrm{OFFSPRING}(x,y) \supset \mathrm{FATHER}(y,x)]$

so the update should be accepted. Either way, the example hopefully indicates that the enforcement of correct argument typing poses some difficulties in the case of first order data bases.

The purpose of this paper is to show, in the case of first order data bases, how a type data base, representing the known specific and general facts about types, can be used to enforce integrity constraints of the form (1.4) thereby ensuring that all arguments to a relation will be of the right type. The method is not completely general. First, as it is described in this paper, it applies only to function free data bases, although the approach will generalize to first order data bases with function signs. Secondly, it applies only to ground literals, or to formulae whose prenex normal forms involve only universal quantifiers. Since universally quantified prenex form formulae (e.g. (1.2), (1.6)) are extremely common in first order data base applications, the method is of some practical consequence.

## 2. FORMAL PRELIMINARIES

We shall be dealing with a first order language _without_ function signs. Hence, assume given the following:

1. _Constant Signs:_ $c_1, c_2, \ldots,$

In the intended interpretation, constant signs will denote individual entities, e.g., part-33, John-Doe, etc.

2. _Variables:_ $x_1, x_2, \ldots,$

3. Logical Connectives: $\wedge$ (and), $\vee$ (or), $\sim$ (not), $\supset$ (implies), $\equiv$ (equivalence)

4. Predicate Signs: P, Q, R, ...

With each predicate sign P is associated an integer $n \geq 0$ denoting the number of arguments of P . P will be called an n-ary predicate sign. We assume the predicate signs to be partitioned into two classes:

(i) A class of unary predicate signs, which will be called simple types. Not all unary predicate signs, need be simple types. In the intended interpretation, simple types (e.g. MALE, HUMAN) as well as various Boolean combinations of these, called types (e.g. MALE $\wedge$ HUMAN) will be used to restrict the allowable ranges of variables occurring in data base formulae as well as to specify integrity constraints on the allowable arguments of predicates.

(ii) The class of remaining predicate signs, which will be called common predicate signs. In the intended interpretation, common predicate signs will denote data base relations, e.g. FATHER, HUSBAND-OF.

The set of types is the smallest set satisfying the following:

(a) A simple type is a type.

(b) If $\tau_1$ and $\tau_2$ are types, so also are $\tau_1 \wedge \tau_2$ , $\tau_1 \vee \tau_2$ , $\sim\tau_1$ .

We shall have occasion to view types as predicates taking arguments. Accordingly, we make the following definition: If t is a variable or constant sign, $\tau$ a non simple type, and $\tau_1$ and $\tau_2$ types then

(i) If $\tau$ is $\tau_1 \wedge \tau_2$ , $\tau(t)$ is $\tau_1(t) \wedge \tau_2(t)$

(ii) If $\tau$ is $\tau_1 \vee \tau_2$ , $\tau(t)$ is $\tau_1(t) \vee \tau_2(t)$

(iii) If $\tau$ is $\sim\tau_1$ , $\tau(t)$ is $\sim\tau_1(t)$ .

5. Quantifiers:

If $x$ is a variable then $(x)$ is a universal quantifier and $(Ex)$ is an existential quantifier.

## 2.1 The Syntax of Data Base Formulae

We define the following syntactic objects:

1. Terms

A term is either a variable or constant sign.

2. Common Literals

If $P$ is an n-ary common predicate sign and $t_1, \ldots, t_n$ terms, then $P(t_1, \ldots, t_n)$ is a common atomic formula. Both $P(t_1, \ldots, t_n)$ and $\sim P(t_1, \ldots, t_n)$ are common literals.

3. Typed Well Formed Formulae (Twffs)

The set of twffs is the smallest set satisfying;

(i) A common literal is a twff.

(ii) If $W_1$ and $W_2$ are twffs, so also are $\sim W_1$, $W_1 \wedge W_2$, $W_1 \vee W_2$, $W_1 \supset W_2$.

(iii) If $W$ is a twff, and $\tau$ a type, then $(x)[\tau(x) \supset W]$ and $(Ex)[\tau(x) \wedge W]$ are twffs. These will be denoted by $(x/\tau)W$ and $(Ex/\tau)W$ respectively. $(x/\tau)$ is a restricted universal quantifier and $(Ex/\tau)$ is a restricted existential quantifier.

Examples of twffs are (1.3) and (1.6). In this paper we consider only closed twffs i.e. twffs with no free variables.

## 2.2 The Type Data Base

The type data base is where all information about types resides. Formally, we define a type data base (TDB) to be any finite set of closed first order formulae all of whose predicate signs are simple types and which satisfies the following $\tau$-completeness property:

For each simple type $\tau$ and each constant $c$, either $TDB \vdash \tau(c)$[1] or $TDB \vdash \sim\tau(c)$.

This $\tau$-completeness property is the appropriate formalization of the requirement that for each data base individual and for all simple types, we know to which type that individual belongs and to which it does not belong. For the TDB (1.5) of Section 1, if HUMAN (Maureen) were all we are given about Maureen then the TDB would not be $\tau$-complete since neither TDB $\vdash$ FEMALE (Maureen) nor TDB $\vdash$ ~FEMALE (Maureen). If instead we were given FEMALE (Maureen) then the TDB would be $\tau$-complete since HUMAN (Maureen), FEMALE (Maureen) and ~MALE (Maureen) are all derivable.

We are not seriously proposing that, in an implementation of a question-answering system, the TDB be represented as a set of first order formulae. There are far more efficient and perspicuous representations of the same facts. One such representation involving sematic networks is thoroughly discussed in [McSkimin 1976, McSkimin and Minker 1977]. A different approach is described in [Bishop and Reiter 1980]. Since such representations, and their associated procedures, are beyond the intended scope of this paper, we do not discuss them here. Regardless of how the information of the TDB is represented, there is one central observation which can be made:

Formally, the TDB is a set of formulae of the monadic predicate calculus. As

---

[1] In general, if $A$ is a set of first order formulae and $W$ is a first order formula, then $A \vdash W$ means that $W$ is provable from the formulae of $A$.

is well known [Hilbert and Ackermann 1950], the monadic predicate calculus is decidable i.e. there exists an algorithm which determines, for any formula $W$, whether or not $TDB \vdash W$. This must remain true regardless of how the TDB is represented. Henceforth, we shall assume the availability of such a decision procedure for the TDB. An efficient decision procedure for a large and natural class of TDB's is described in [Bishop and Reiter 1980].

If $\tau$ is a type, defined $|\tau|_{TDB} = \{c \mid c$ is a constant sign and $TDB \vdash \tau(c)\}$. When the TDB is clear from context, we shall write $|\tau|$ instead of $|\tau|_{TDB}$.

The notion of a type data base as applied to deductive question-answering has been independently proposed in [McSkimin 1976, McSkimin and Minker 1977]. What we have been calling simple types and types, McSkimin and Minker call primitive categories and Boolean category expressions respectively. While McSkimin and Minker do not explicitly make the $\tau$-completeness assumption it appears to be implicit in the ways they use the type data base.

## 2.3 Predicate Argument Type Constraints

We shall assume that with each n-ary common predicate sign $P$ there is an associated predicate argument type constraint of the form:

$$(x_1,\ldots,x_n)[P(x_1,\ldots,x_n) \supset \tau_P^1(x_1) \wedge \ldots \wedge \tau_P^n(x_n)] \tag{2.1}$$

where $\tau_P^1,\ldots,\tau_P^n$ are types. This will be viewed as an integrity constraint specifying that the $i$-th argument of $P$ must always satisfy the type $\tau_P^i$. The formulae (1.4) of Section 1 are examples of such constraints.

## 3. Updates with Universally Quantified Twffs

Our objective in this section is to show how a universally quantified prenex normal form twff may be tested for integrity with respect to the set of predicate argument type constraints of the form (2.1).

### 3.1 The Formula INT(W)

We begin by noting that

$$\vdash (2.1) \supset (\vec{x})[P(\vec{x}) \equiv P(\vec{x}) \wedge \tau_P^1(x_1) \wedge \ldots \wedge \tau_P^n(x_n)]$$

Hence, if $W$ is a twff, and $INT(W)$ is obtained from $W$ by replacing each common atomic formula $P(t_1, \ldots, t_n)$ by $P(t_1, \ldots, t_n) \wedge \tau_P^1(t_1) \wedge \ldots \wedge \tau_P^n(t_n)$ then

$$PATC \vdash W \equiv INT(W)$$

where PATC is the set of all predicate argument type constraints of the form (2.1) associated with the common predicate signs of the data base. This means that instead of updating the data base with a twff $W$, we can choose instead to update with the equivalent (as far as the integrity constraints are concerned) formula $INT(W)$ .

Example 3.1

(i) With reference to the predicate argument type constraints (1.4), if $W$ is MOTHER (Mary, John) then $INT(W)$ is

HUMAN (Mary) ∧ FEMALE (Mary) ∧ HUMAN (John) ∧ MOTHER (Mary, John).

If $W$ is ~MOTHER (Bill, Mary) then $INT(W)$ is

~[HUMAN (Bill) ∧ FEMALE (Bill) ∧ HUMAN (Mary) ∧ MOTHER (Bill, Mary)].

(ii) If $W$ is

$$(x/\tau)(y/\theta)[P(x,y) \supset \sim Q(a,y) \vee R(x,x)]$$

then $INT(W)$ is

$$(x/\tau)(y/\theta)[\tau_P^1(x) \wedge \tau_P^2(y) \wedge P(x,y) \supset \sim[\tau_Q^1(a) \wedge \tau_Q^2(y) \wedge Q(a,y)] \vee [\tau_R^1(x) \wedge \tau_R^2(x) \wedge R(x,x)]] .$$

Clearly, $INT(W)$ imposes on $W$ the integrity constraint that each predicate argument satisfy the corresponding argument types for that predicate. Our approach to data base integrity will be to consider the effects of updating the data base with $INT(W)$. This update will be rejected if the addition of $INT(W)$ to the data base

(i) leads to an inconsistency with respect to the TDB or

(ii) provides no new information, in a sense to be defined below.

On the other hand, if $INT(W)$ leads to no integrity violations, then the data base will be updated with $INT(W)$.[1] Thus, in the process of creating or updating a data base, the user will enter a twff $W$. A subsystem responsible for maintaining the integrity of the data base will transform $W$ to $INT(W)$. If $INT(W)$ violates no integrity constraints, the data base will be updated with $INT(W)$. There is a strong analogy here between our proposal for data base integrity and compilers for strongly typed programming languages like PASCAL or ALGOL 68. In such languages, all variables must be typed, just as all variables in twffs are assigned types. Furthermore, in typed programming languages, the formal parameters of a procedure must be typed, and any attempt

---

[1]Actually, as we shall see, the data base is not updated with $INT(W)$, but with a set of simpler, but logically equivalent formulae.

to bind an argument of conflicting type to a formal parameter will be rejected by the compiler. Under our approach to integrity, predicates correspond to procedures, and predicate argument types to parameter types. At "compile time" i.e. when an attempted update of the data base is made, the integrity "compiler" will seek out conflicting "argument-parameter" types. Should any be found, the update will be rejected.

### 3.2 Updates Involving Constants

With no loss in generality, assume that the data base is to be updated with a twff I in prenex normal form, so that I has the form $(\vec{x}/\vec{\tau})W$[1], where W is quantifier free. Assume further that W is in conjunctive normal form. Thus I is of the form

$$(\vec{x}/\vec{\tau})[C_1 \wedge C_2 \wedge \ldots \wedge C_m]$$

where each $C_i$ is a disjunct of common literals. This, in turn, is equivalent to

$$(\vec{x}/\vec{\tau})C_1 \wedge (\vec{x}/\vec{\tau})C_2 \wedge \ldots \wedge (\vec{x}/\vec{\tau})C_m .$$

Thus, the original update is equivalent to the m updates $(\vec{x}/\vec{\tau})C_i$ , $i = 1, \ldots, m$ . Our position will be that if any of these m twffs violates an integrity constraint, then the original twff I will be rejected. Thus, again with no loss in generality, we consider updates of the form $(\vec{x}/\vec{\tau})C$ where $C = L_1 \vee \ldots \vee L_k$ is a disjunct of common literals. By virtue of the discussion of Section 3.1 we can equivalently consider the effects of updating the data base with

$$\begin{aligned} INT((\vec{x}/\vec{\tau})C) &= (\vec{x}/\vec{\tau})INT(C) \\ &= (\vec{x}/\vec{\tau})[INT(L_1) \vee \ldots \vee INT(L_k)] \end{aligned}$$

---

[1] $(\vec{x}/\vec{\tau})W$ denotes $(x_1/\tau_1) \ldots (x_n/\tau_n)W$ . We admit the case $n = 0$ in which case the twff is quantifier free.

We consider first the case where some literal, say $L_1$ , contains a constant $c$ .

Case 1. $L_1$ is positive, say $L_1$ is $P(c,t_2, \ldots ,t_m)$ for terms $t_2,\ldots,t_m$ . Then

$$INT(C) = [\tau_P^1(c) \wedge \tau_P^2(t_2) \wedge \ldots \wedge \tau_P^m(t_m) \wedge P(c,t_2,\ldots,t_m)] \vee INT(L_2) \vee \ldots \vee INT(L_k) .$$

Suppose $TDB \vdash \sim\tau_P^1(c)$ . Then

$$TDB \vdash INT(C) \equiv [INT(L_2) \vee \ldots \vee INT(L_k)]$$

i.e. the information about $L_1$ in $C$ is irrelevant! We interpret this as an integrity violation. Notice in particular the case $k = 1$ , namely when $C$ is a single literal $L_1$ . In that case $TDB \vdash INT(C) \equiv false$ so that an attempted update with $(\vec{x}/\vec{\tau})INT(C)$ would lead to a genuine data base inconsistency.

Case 2. $L_1$ is negative, say $L_1$ is $\sim P(c,t_2,\ldots,t_m)$ for terms $t_2,\ldots,t_m$ . Then

$$INT(C) = \sim\tau_P^1(c) \vee \sim\tau_P^2(t_2) \vee \ldots \vee \sim\tau_P^m(t_m) \vee \sim P(c,t_2, \ldots t_m) \vee INT(L_2) \vee \ldots \vee INT(L_k) .$$

Suppose $TDB \vdash \sim\tau_P^1(c)$ . Then $TDB \vdash INT(C)$ i.e. $INT(C)$ is vacuous; it contains no new information. This we treat as an integrity violation.

These observations lead to the following:

Integrity Rule 1

Reject any attempted update of the data base with a twff $(\vec{x}/\vec{\tau})C$ where $C$ is a disjunct of common literals whenever

(i) a constant sign $c$ occurs in $C$ , say as the $i$ - th argument of a common predicate sign P , and

(ii) $c \notin |\tau_P^i|$ .

As we shall see, an attempted update which passes Rule 1 may still violate further integrity constraints. However, notice that, in Case 1 above, if $(\vec{x}/\vec{\tau})C$ passes Rule 1 then TDB $\not\vdash \sim\tau_P^1(c)$ . By the $\tau$-completeness of the TDB , this means TDB $\vdash \tau_P^1(c)$ so that

$$\text{TDB} \vdash \text{INT}(C) \equiv [\tau_P^2(t_2) \wedge \ldots \wedge \tau_P^m(t_m) \wedge P(c,t_2,\ldots,t_m)] \vee \text{INT}(L_2) \vee \ldots \vee \text{INT}(L_k) .$$

If $(\vec{x}/\vec{\tau})C$ passes Rule 1 by virtue of Case 2, then we similarly obtain

$$\text{TDB} \vdash \text{INT}(C) \equiv [\sim\tau_P^2(t_2) \vee \ldots \vee \sim\tau_P^m(t_m) \vee \sim P(c,t_2,\ldots,t_m)] \vee \text{INT}(L_2) \vee \ldots \vee \text{INT}(L_k) .$$

In either case, INT(C) is equivalent to a formula which is independent of the type literal $\tau_P^1(c)$ , so that an update with $(\vec{x}/\vec{\tau})\text{INT}(C)$ is equivalent to one in which all literals in INT(C) of the form $\tau_P^i(c)$ have been deleted.

### 3.3 Typed Normal Form

For subsequent integrity tests, we require the following propositional identity:

$$\sim(U_1 \wedge M_1) \vee \ldots \vee \sim(U_r \wedge M_r) \vee (W_1 \wedge L_1) \vee \ldots \vee (W_k \wedge L_k)$$

$$\equiv \bigwedge_{(i_1,\ldots,i_k)\in\{0,1\}^k} \left\{ \begin{array}{l} U_1 \wedge \ldots \wedge U_r \wedge W_1^{i_1} \wedge \ldots \wedge W_k^{i_k} \supset [\sim M_1 \vee \ldots \vee \sim M_r \\ \vee\, i_1 L_1 \vee \ldots \vee i_k L_k] \end{array} \right.$$

where

$W^i = W$ if $i = 1$

$\quad = \sim W$ if $i = 0$

and

$iL = L$ if $i = 1$

$\quad = 0$ (false) if $i = 0$ .

In particular, if $U_1, \ldots, U_r, W_1, \ldots, W_k$ are types in the variable $x$ , then

$$(x/\tau)(\vec{y}/\vec{\theta})[\sim(U_1(x) \wedge M_1) \vee \ldots \vee \sim(U_r(x) \wedge M_r) \vee (W_1(x) \wedge L_1) \vee \ldots \vee (W_k(x) \wedge L_k)]$$

$$\equiv \bigwedge_{(i_1,\ldots,i_k)\varepsilon\{0,1\}^k} \left\{ \begin{array}{l} (x/\tau \wedge U_1 \wedge \ldots \wedge U_r \wedge W_1^{i_1} \wedge \ldots \wedge W_k^{i_k})(\vec{y}/\vec{\theta}) \\ [\sim M_1 \vee \ldots \vee \sim M_r \vee i_1L_1 \vee \ldots \vee i_kL_k] \; . \end{array} \right. \tag{3.1}$$

Now our concern is with attempted updates with twffs of the form $(x/\tau)(\vec{y}/\vec{\theta})C$ where $C$ is a disjunct of common literals, say

$$C = \sim A_1 \vee \ldots \vee \sim A_r \vee B_1 \vee \ldots \vee B_k$$

with the A's and B's positive literals. Thus INT(C) has the form

$$\text{INT}(C) = \sim(U_1(x) \wedge M_1) \vee \ldots \vee \sim(U_r(x) \wedge M_r) \vee (W_1(x) \wedge L_1) \vee \ldots \vee (W_k(x) \wedge L_k)$$

where $U_i$ is a conjunct of the those predicate argument types corresponding to an occurrence of $x$ in $A_i$ (and hence $U_i$ is a type), and $M_i$ is $A_i$ conjoined with type literals corresponding to occurrences of constants or of variables other than $x$ in $A_i$ . Similarly for $W_i$ and $L_i$ respectively. For example, if the formula is $(x/\tau)(y/\theta)C$ where

$$C = \sim P(x,a,y) \vee \sim Q(x,y) \vee P(b,y,y) \vee Q(x,x)$$

then

$$\text{INT}(C) = \sim[\tau_P^1(x) \wedge \tau_P^2(a) \wedge \tau_P^3(y) \wedge P(x,a,y)] \vee \sim[\tau_Q^1(x) \wedge \tau_Q^2(y) \wedge Q(x,y)]$$

$$\vee\ [\tau_P^1(b) \wedge \tau_P^2(y) \wedge \tau_P^3(y) \wedge P(b,y,y)] \vee [\tau_Q^1(x) \wedge \tau_Q^2(x) \wedge Q(x,x)]$$

so that

$$U_1 = \tau_P^1 \qquad M_1 = \tau_P^2(a) \wedge \tau_P^3(y) \wedge P(x,a,y)$$

$$U_2 = \tau_Q^1 \qquad M_2 = \tau_Q^2(y) \wedge Q(x,y)$$

$$W_1 = 1 \text{ (true)} \qquad L_1 = \tau_P^1(b) \wedge \tau_P^2(y) \wedge \tau_P^3(y) \wedge P(b,y,y)$$

$$W_2 = \tau_Q^1 \wedge \tau_Q^2 \qquad L_2 = Q(x,x)$$

In general, using (3.1), it follows that $(x/\tau)(\vec{y}/\vec{\theta})C$ can be represented by the right side of (3.1) i.e. as a conjunct of $2^k$ formulae such that no $M$ or $L$ involves a type literal in $x$. For the example at hand, we obtain 4 such formulae whose conjunct is equivalent to the original:

$$(x/\tau \wedge \tau_P^1 \wedge \tau_Q^1 \wedge 1 \wedge \tau_Q^1 \wedge \tau_Q^2)(y/\theta)[\sim M_1 \vee \sim M_2 \vee L_1 \vee L_2]$$

$$(x/\tau \wedge \tau_P^1 \wedge \tau_Q^1 \wedge 0 \wedge \tau_Q^1 \wedge \tau_Q^2)(y/\theta)[\sim M_1 \vee \sim M_2 \vee L_2]$$

$$(x/\tau \wedge \tau_P^1 \wedge \tau_Q^1 \wedge 1 \wedge \sim(\tau_Q^1 \wedge \tau_Q^2))(y/\theta)[\sim M_1 \vee \sim M_2 \vee L_1]$$

$$(x/\tau \wedge \tau_P^1 \wedge \tau_Q^1 \wedge 0 \wedge \sim(\tau_Q^1 \wedge \tau_Q^2))(y/\theta)[\sim M_1 \vee \sim M_2]$$

Now for each of the $2^k$ formulae obtained by applying (3.1) to $(x/\tau)(\vec{y}/\vec{\theta})C$ we can repeat this process with respect to the y's until finally, we obtain a conjunct $K$ of formulae with restricted universal quantifiers, and in which the only occurrences of types are in the restricted quantifier, or as type literals of the form $\tau(a)$ where $a$ is a constant sign. Assuming that the original twff $(x/\tau)(\vec{y}/\vec{\theta})C$ has passed the Integrity Rule 1 of Section 3.1, we can, by the remarks following that rule, delete all occurrences of type literals $\tau(a)$ from $K$. The resulting set of twffs in this conjunct is called the <u>typed normal form</u> of $(x/\tau)(\vec{y}/\vec{\theta})C$ .

Example 3.2

1. $(x/\tau)[\sim P(x,x) \vee Q(x,a)]$

has typed normal form

$(x/\tau \wedge \tau_P^1 \wedge \tau_P^2 \wedge \tau_Q^1)[\sim P(x,x) \vee Q(x,a)]$

$(x/\tau \wedge \tau_P^1 \wedge \tau_P^2 \wedge \sim\tau_Q^1)[\sim P(x,x)]$

2. $(x/\tau)[P(x,x) \vee Q(x,a)]$

has typed normal form

$(x/\tau \wedge \tau_P^1 \wedge \tau_P^2 \wedge \tau_Q^1)[P(x,x) \vee Q(x,a)]$

$(x/\tau \wedge \tau_P^1 \wedge \tau_P^2 \wedge \sim\tau_Q^1)[P(x,x)]$

$(x/\tau \wedge \sim(\tau_P^1 \wedge \tau_P^2) \wedge \tau_Q^1)[Q(x,a)]$

$(x/\tau \wedge \sim(\tau_P^1 \wedge \tau_P^2) \wedge \sim\tau_Q^1)$ FALSE

3. $(x/\tau)[\sim P(x,x) \vee \sim Q(x,a)]$

has typed normal form

$(x/\tau \wedge \tau_P^1 \wedge \tau_P^2 \wedge \tau_Q^1)[\sim P(x,x) \vee \sim Q(x,a)]$

4. $(x/\tau)(y/\theta)[\sim P(x,y) \vee Q(x,y)]$

has typed normal form

$(x/\tau \wedge \tau_P^1 \wedge \tau_Q^1)(y/\theta \wedge \tau_P^2 \wedge \tau_Q^2)[\sim P(x,y) \vee Q(x,y)]$

$(x/\tau \wedge \tau_P^1 \wedge \tau_Q^1)(y/\theta \wedge \tau_P^2 \wedge \sim\tau_Q^2)[\sim P(x,y)]$

$(x/\tau \wedge \tau_P^1 \wedge \sim\tau_Q^1)(y/\theta \wedge \tau_P^2)[\sim P(x,y)]$

5. $(x/\tau)(y/\theta)[\sim P(x,y) \vee \sim Q(x,y)]$

has typed normal form

$$(x/\tau \wedge \tau_P^1 \wedge \tau_Q^1)(y/\theta \wedge \tau_P^2 \wedge \tau_Q^2)[\sim P(x,y) \vee \sim Q(x,y)]$$

6. $(x/\tau)(y/\theta)[P(x,y) \vee Q(x,y)]$

has typed normal form

$$(x/\tau \wedge \tau_P^1 \wedge \tau_Q^1)(y/\theta \wedge \tau_P^2 \wedge \tau_Q^2)[P(x,y) \vee Q(x,y)]$$

$$(x/\tau \wedge \tau_P^1 \wedge \tau_Q^1)(Y/\theta \wedge \tau_P^2 \wedge \sim\tau_Q^2)[P(x,y)]$$

$$(x/\tau \wedge \tau_P^1 \wedge \tau_Q^1)(y/\theta \wedge \sim\tau_P^2 \wedge \tau_Q^2)[Q(x,y)]$$

$$(x/\tau \wedge \tau_P^1 \wedge \tau_Q^1)(y/\theta \wedge \sim\tau_P^2 \wedge \sim\tau_Q^2) \text{ FALSE}$$

$$(x/\tau \wedge \tau_P^1 \wedge \sim\tau_Q^1)(y/\theta \wedge \tau_P^2)[P(x,y)]$$

$$(x/\tau \wedge \tau_P^1 \wedge \sim\tau_Q^1)(y/\theta \wedge \sim\tau_P^2) \text{ FALSE}$$

$$(x/\tau \wedge \sim\tau_P^1 \wedge \tau_Q^1)(y/\theta \wedge \tau_Q^2)[Q(x,y)]$$

$$(x/\tau \wedge \sim\tau_P^1 \wedge \tau_Q^1)(y/\theta \wedge \sim\tau_Q^2) \text{ FALSE}$$

$$(x/\tau \wedge \sim\tau_P^1 \wedge \sim\tau_Q^1)(y/\theta) \text{ FALSE}$$

Now notice that if an update is attempted with $(\vec{x}/\vec{\tau})C$ where $C$ is a disjunct of literals, then each twff in its typed normal form is of the form $(\vec{x}/\vec{\theta})\hat{C}$ where $\hat{C}$ is disjunct of some, or all, of the literals of $C$. Hence, $\hat{C}$ contains no types so that $(\vec{x}/\vec{\theta})\hat{C}$ is a twff and thus a respectable candidate for inclusion in the data base.

It is natural, therefore, to consider updating the data base with all the twffs in the typed normal form of $(\vec{x}/\vec{\tau})C$. Before doing so, let us consider a typical twff $(\vec{x}/\vec{\theta})\hat{C}$ in this typed normal form. Suppose, for some component $\theta_i$ of $\vec{\theta}$, that $TDB \vdash (x) \sim \theta_i(x)$. In that case, the twff $(\vec{x}/\vec{\theta})\hat{C}$ is vacuously true; it contains no new information, and hence is irrelevant to the update. We define a twff $(\vec{x}/\vec{\theta})C$ to be <u>vacuous</u> iff for some component $\theta_i$ of $\vec{\theta}$ it is the case that $TDB \vdash (x) \sim \theta_i(x)$. Given a typed normal form, its <u>reduced</u> form is obtained by deleting all vacuous twffs. Our approach to data base updates, then, is as follows:

Given an attempted update with $(\vec{x}/\vec{\tau})C$, form its reduced typed normal form. Assuming that this reduced form satisfies certain integrity constraints, to be described below, we then update the data base with all of the twffs in this reduced form.

Before we discuss integrity constraints as they apply to reduced type normal forms, it is worth taking a closer look at the notion of a vacuous twff. In particular, notice that $TDB \vdash (x) \sim \theta_i(x)$ is not equivalent to $|\theta_i| = \phi$. The former implies the latter (assuming a consistent TDB) but not conversely. For example, suppose the TDB consists of the following facts:

$(x)HUMAN(x) \supset ANIMATE(x)$

$ANIMATE\ (fido)$

$\sim HUMAN\ (fido)$

Then $|HUMAN| = \phi$, yet it is not the case that $TDB \vdash (x) \sim HUMAN(x)$. On the other hand, $TDB \vdash (x) \sim (HUMAN(x) \wedge \sim ANIMATE(x))$ and indeed $|HUMAN \wedge \sim ANIMATE| = \phi$. Now we were careful, in defining the notion of a vacuous twff, to require the stronger condition $TDB \vdash (x) \sim \theta_i(x)$ rather than the weaker $|\theta_i| = \phi$. To see why, consider an attempt to update with "Everyone likes Fido":

$(x/HUMAN)LIKE(x,Fido)$

Assume $\tau^{1}_{LIKE}$ = HUMAN. Then this has typed normal form :

$$(x/HUMAN)LIKE(x,fido) \tag{3.2}$$

$$(x/HUMAN \wedge \sim HUMAN)\ FALSE$$

The latter is clearly vacuous and is deleted in forming the reduced typed normal form. Under the definition of vacuous twff, the former is not vacuous and hence is retained. However, had we defined the notion of a vacuous twff to require $|\theta_i| = \phi$ , then (3.2) would also be deleted in forming the reduced form of the original update i.e. the entire update would be rejected. Now it is indeed true that for <u>this</u> TDB, the twff (3.2) contains no information. But this is so only because currently the TDB knows of no humans. Should the TDB be subsequently updated with a new fact, say HUMAN (John), (3.2) would no longer be information-free. In other words, $|HUMAN| = \phi$ is contingent on the extension of the TDB, and is not a universal fact about the world. Furthermore, any rejection of (3.2) because it is currently information-free would not be immune to subsequent updates of the TDB with facts like HUMAN (John); once the TDB contains such a fact, the rejected formula suddenly becomes relevant. For these reasons, we defined the notion of a vacuous twff as we did. Any such twff is indeed information-free, but only by virtue of general rather than contingent facts about the world.

Now, consider an attempted update with $(\vec{x}/\vec{\tau})C$ . As we remarked earlier, each twff in its reduced typed normal form is of the form $(\vec{x}/\vec{\theta})\hat{C}$ where $\hat{C}$ is a disjunct of some, or all, of the common literals of C . Suppose that C contains a common literal L which appears in none of the twffs in this reduced typed normal form. Then L is irrelevant to the attempted update. We interpret this as an integrity violation; at best there is something questionable about the attempted update. Finally, suppose that the reduced typed normal form contains a twff of the form $(\vec{x}/\vec{\theta})$FALSE. By (3.1) this is possible iff C is a disjunct of positive literals. In this case asserting $(\vec{x}/\vec{\theta})$FALSE is equivalent to updating the TDB with

$$(x_1)\sim\theta_1(x_1) \vee (x_2)\sim\theta_2(x_2) \vee \ldots \vee (x_n)\sim\theta_n(x_n) \qquad (3.3)$$

Clearly, we cannot permit the original update if (3.3) is inconsistent with the TDB. On the other hand, if (3.3) is consistent with the TDB, but not provable, then it is a new fact for the TDB and, since this is a subtle consequence of the attempted update, the user should be asked about the relevance of (3.3) for the TDB.

Integrity Rule 2

Suppose the data base is to be updated with $(\vec{x}/\vec{\tau})$C and that C contains a common literal L which occurs in none of the twffs of the reduced typed normal form of $(\vec{x}/\vec{\tau})$C . Then reject the attempted update. Otherwise, there are two possibilities;

(i) The reduced typed normal form contains no twff of the form $(\vec{x}/\vec{\theta})$FALSE. Then update the data base with all of the twffs in this reduced typed normal form.

(ii) There is a twff of the form $(\vec{x}/\vec{\theta})$FALSE, so that C is a disjunct of positive literals. If (3.3) is inconsistent with the TDB, reject the update. If (3.3) is provable from the TDB, ignore it. Otherwise ask the user whether (3.3) is an appropriate update for the TDB. If so, make that update. If all such TDB updates are acceptable, update the data base with the remaining twffs of the reduced typed normal form.

Example 3.3

Consider an attempted update with example (1.5) of Section 1, namely with:

$$(x/\text{HUMAN})(y/\text{HUMAN})[\text{OFFSPRING}(x,y) \supset \text{MOTHER}(y,x) \vee \text{FATHER}(y,x)] \tag{3.4}$$

Assume

$$\tau^1_{\text{OFFSPRING}} = \tau^2_{\text{OFFSPRING}} = \tau^2_{\text{FATHER}} = \tau^2_{\text{MOTHER}} = \text{HUMAN}$$

$$\tau^1_{\text{MOTHER}} = \text{HUMAN} \wedge \text{FEMALE}$$

$$\tau^1_{\text{FATHER}} = \text{HUMAN} \wedge \text{MALE}$$

and assume further that

$$\text{TDB} \vdash (x)\sim[\text{MALE}(x) \wedge \text{FEMALE}(x)] \tag{3.5}$$

After some simplification, and using (3.5), we obtain the reduced typed normal form of (3.4):

$$(x/\text{HUMAN})(y/\text{HUMAN} \wedge \text{FEMALE})[\text{OFFSPRING}(x,y) \supset \text{MOTHER}(y,x)] \tag{3.6}$$

$$(x/\text{HUMAN})(y/\text{HUMAN} \wedge \text{MALE})[\text{OFFSPRING}(x,y) \supset \text{FATHER}(y,x)] \tag{3.7}$$

These satisfy Integrity Rule 2, so the original twff (3.4) is acceptable, and we update the data base with (3.6) and (3.7).

Notice, incidentally, how the reduced typed normal form decomposes the original twff (3.4) into just the right conceptual "chunks" with respect to the types of the TDB. Thus (3.6) and (3.7) are clearer, and more to the point than the original twff. Notice also that while the original twff is not a Horn formula, the twffs of its reduced typed normal form are Horn. Since there are many representational and computational advantages to Horn representations in data base theory (See e.g. [Kowalski 1979]) this Horn decomposition is a fortunate consequence of reduced typed normal forms. Of course, reduced typed normal forms do not always yield Horn formulae, but it is comforting to know that they do on occasion. Moreover, it is easy to see, from (3.1), that Horn formulae never yield non Horn components in their typed normal form, so that reduction to normal form preserves the Horn property.

Example 3.4

Consider an attempted update with

$(x/HUMAN \wedge MALE)(y/HUMAN \wedge MALE)[BROTHER(x,y) \supset SISTER(y,x)]$

Assuming

$\tau^1_{BROTHER} = HUMAN \wedge MALE$

$\tau^1_{SISTER} = HUMAN \wedge FEMALE$

$\tau^2_{BROTHER} = \tau^2_{SISTER} = HUMAN$

the typed normal form is

$$(x/HUMAN \wedge MALE)(y/HUMAN \wedge MALE \wedge FEMALE)[BROTHER(x,y) \supset SISTER(y,x)] \quad (3.8)$$

$$(x/HUMAN \wedge MALE)(y/HUMAN \wedge MALE \wedge \sim FEMALE)\sim BROTHER(x,y) \quad (3.9)$$

$$(x/HUMAN \wedge MALE \wedge \sim HUMAN)(y/HUMAN \wedge MALE)\sim BROTHER(x,y) \quad (3.10)$$

(3.10) is clearly vacuous. (3.8) is vacuous by (3.5). Hence, the reduced typed normal form consists of (3.9) so by Integrity Rule 2, the update is rejected.

Example 3.5

Consider an attempted update with

(x/HUMAN)BROTHER(x,John)

where the BROTHER relation satisfies the same predicate argument type constraints as in Example 3.4. This has typed normal form

(x/HUMAN $\wedge$ MALE)BROTHER(x,John)

(x/HUMAN $\wedge$ ~MALE)FALSE

This latter formula is equivalent to a TDB update with

(x)[~HUMAN(x) $\vee$ MALE(x)] (3.11)

By Integrity Rule 2, if (3.11) is consistent with the TDB, then the user should be asked whether to update the TDB with (3.11); presumably it will be rejected whence so also will be the original update. On the other hand, if the TDB contains

(x)~[MALE(x) $\wedge$ FEMALE(x)]

HUMAN (Mary) FEMALE (Mary)

then (3.11) is inconsistent with the TDB and the system would automatically reject the original update.

## 4. Discussion and Conclusions

We have focussed in this paper upon a special class of integrity constraints, namely those which specify, for every data base relation, the allowable arguments to the relation. The primary vehicle for the analysis of these constraints is the notion of a type data base, together with the reduced typed normal form of a universally quantified twff. This normal form enjoys a number of desirable properties:

1. There is an algorithm for obtaining it.

2. There are simple criteria which, when applied to a formula's typed normal form, determine whether that formula violates any argument typing integrity constraints (Integrity Rule 2).

3. The conjunction of the formulae in the reduced typed normal form is logically equivalent to the original formula (modulo the TDB and integrity constraints).

4. As discussed in Example 3.3, the reduced typed normal form often decomposes the original formula into just the right conceptual "chunks". Moreover, non Horn formulae may decompose into Horn "components", while Horn formulae never yield non Horn formulae in their normal forms.

5. In view of 3., a formula may be represented in the data base by its reduced typed normal form. In view of 4., this is a good thing to do.

McSkimin and Minker have independently observed the utility of predicate argument typing in maintaining the integrity of a first order data base [McSkimin 1976], [McSkimin and Minker 1977]. Their approach differs significantly from ours, however, and in some respects is less general. Both approaches diverge with respect to what constitutes an acceptable update of the data base. For example, the update of Example 3.3 would be rejected under their approach, whereas we find it acceptable. Moreover, McSkimin and Minker would not detect possible TDB integrity violations arising from twffs of the form $(\vec{x}/\vec{\theta})$FALSE in the reduced typed normal form. For example, they would accept the update of Example 3.5 whereas we find it unacceptable.

There are several directions in which the results of this paper might be extended:

1. Our approach applies only to universally quantified twffs. Is there a normal form for arbitrarily quantified twffs?

2. We have considered only twffs with no function signs. How might the notion of typed functions be incorporated into the theory?

3. The class of predicate argument type constraints considered in this paper, namely those of the form (2.1), is not as general as one might like. Frequently, corresponding to a constraint like (2.1), there is a natural refinement of the constraint which does not fit the pattern of (2.1), but which should be enforced. For example, in a personnel world one might define the constraint

   $(x\ y)[\text{EMPLOYED-IN}(x,y) \supset \text{EMPLOYEE}(x) \wedge \text{DEPT}(y)]$

   which is of the form (2.1). This has the natural refinement

   $(x\ y)[\text{EMPLOYED-IN}(x,y) \supset \text{SALES-PERSON}(x) \wedge \text{SALES-DEPT}(y)$

   $\vee\ \text{CLERICAL-PERSON}(x) \wedge \text{ACCOUNTING-DEPT}(y)]$

   which violates the pattern (2.1) and hence cannot be accommodated by the methods of this paper. The natural approach here is to seek a normal form corresponding to predicate argument type constraints of the form:

$$(x_1,\ldots,x_n)[P(x_1,\ldots,x_n) \supset \tau_1^1(x_1) \wedge \ldots \wedge \tau_n^1(x_n) \vee \ldots \vee \tau_1^k(x_1) \wedge \ldots \wedge \tau_n^k(x_n)] .$$

4. Related to the refinement problem is the specialization problem. Frequently, a type constraint of the form (2.1) will have various specializations. For example, in an education domain, we might have the relation ELECTIVE(x,y), denoting that course x is an elective for the program y :

   $$(x\ y)[\text{ELECTIVE}(x,y) \supset \text{COURSE}(x) \wedge \text{PROGRAM}(y)] \qquad (4.1)$$

   The computer science program, however, is more particular:

   $(x)[\text{ELECTIVE}(x,\text{CS}) \supset \text{SECOND-YEAR-COURSE}(x) \wedge \text{MATH}(x)$

   $\vee\ [\text{THIRD-YEAR-COURSE}(x) \vee \text{FOURTH-YEAR-COURSE}(x)] \wedge \text{ARTS}(x)]$

Similarly, there will be specialization of (4.1) for all of the other degree programs. How might we simultaneously enforce the general constraint (4.1) together with all of its specializations?

5. Many relations naturally take sets as arguments. For example, in an education domain, the relation PREREQUISITES(x,y) would take a set of courses x as the prerequisites for a course y . This integrity constraint might be denoted by

   $(x\ y)[\mathrm{PREREQUISITES}(x,y) \supset \mathrm{SET\text{-}OF(COURSE)}(x) \wedge \mathrm{COURSE}(y)]$

   How might such constraints be enforced?

   One can imagine a similar need for the treatment of sequences.

Acknowledgement

This work was done with financial assistance from the National Science and Engineering Research Council of Canada, under grant A 7642.

REFERENCES

Bishop, C. and Reiter, R. (1980). On taxonomies, Dept. of Computer Science, Univ. of British Columbia, Technical Report, forthcoming.

Date, C.J. (1977). An Introduction to Data Base Systems, Second Edition, Addison-Wesley, Reading, Mass., 1977.

Hilbert, D. and Ackermann, W. (1950). Principles of Mathematical Logic. Chelsea, New York.

Kellogg, C., Klahr, P. and Travis, L. (1978). Deductive planning and pathfinding for relational data bases, in Logic and Data Bases, H. Gallaire and J. Minker eds., Plenum Press, New York, 179-200.

Kowalski, R. (1979). Logic for Problem Solving, North Holland Publishing Co., New York.

McSkimin, J.R. (1976). The Use of Semantic Information in Deductive Question-Answering Systems. Ph.D. Thesis, Dept. of Computer Science, Univ. of Maryland, College Park, Md.

McSkimin, J.R. and Minker, J. (1977). The use of a semantic network in a deductive question-answering system, Technical Report TR-506, Dept. of Computer Science, Univ. of Maryland, College Park, Md.

Minker, J. (1978). An experimental relational data base system based on logic, in Logic and Data Bases, H. Gallaire and J. Minker eds., Plenum Press, New York, 107-147.

Reiter, R. (1977). An approach to deductive question-answering, Tech. Report 3649, Bolt Beranek and Newman Inc., Cambridge, Mass.

Reiter, R. (1978). Deductive question-answering on relational data bases, in Logic and Data Bases, H. Gallaire and J. Minker eds., Plenum Press, New York, 149-177.