AN APPROACH TO THE ORGANIZATION OF TAXONOMIES

by

CRAIG DAVID BISHOP

B.Sc. University of British Columbia

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

in
THE FACULTY OF GRADUATE STUDIES
Department of Computer Science

We accept this thesis as conforming
to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

March, 1981

© Craig David Bishop, 1981
In presenting this thesis in partial fulfillment of the requirements for an advanced degree at the University of British Columbia, I agree that the Library shall make it freely available for reference and study. I further agree that permission for extensive copying of this thesis for scholarly purposes may be granted by the Head of my department or his representative. It is understood that copying or publication of this thesis for financial gain shall not be allowed without my written permission.

Craig Bishop
Abstract

The ISA hierarchy used by present day inferential database systems is deficient in that it does not represent a variety of domain relationships (type relationships). Such hierarchies associate explicitly defined sets with the leaves of a tree. Non-leaf nodes in the tree inherit members from their child nodes. In keeping with the use of sets, this paper gives motivation for having type relationships other than subset. Included are union, intersection and a disjointness condition.

A formalism for the typed database (TDB) is given, using the monadic first order predicate calculus as its theoretical basis. Non-unit formulae represent intensional (general) information about the world being represented and unit ground clauses represent extensional (specific) information. Predicates represent types, and constant symbols represent set members. This is connected to the set concept via predicate extensions. The extension is that set of constant symbols which are provable as arguments to the given predicate.

Given the so-called concreteness condition and consistency of the TDB, the desired set theoretic relationships (union, intersection and disjointness) of predicate extensions follow. This strengthens the link between the formalism of the predicate calculus and the more natural set representation.

A canonical form of a TDB is shown that admits an appropriate machine representation. Using this is can be determined if a constant symbol as an
argument to a given predicate is provable (domain membership) in constant time. An update algorithm is developed and is shown to be correct in that it maintains concreteness and consistency. Thus the TDB is shown to be a practical generalization of the ISA hierarchy but is considerably more expressive.
## Table of Contents

<table>
<thead>
<tr>
<th>Section A</th>
<th>Introduction</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Section B</td>
<td>Motivation</td>
<td>4</td>
</tr>
<tr>
<td>Section C</td>
<td>Formal Preliminaries</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>The TDB</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>Extension</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Acyclicity of a TDB</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>ALPHA, BETA and TAU</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Diagram C.1 - Sample TDB Intension</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>Lemma C.1 - Generalization</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Corollary C.1 - Containment</td>
<td>13</td>
</tr>
<tr>
<td></td>
<td>Lemma C.2 - Intersection</td>
<td>13</td>
</tr>
<tr>
<td>Section D</td>
<td>The Definite Component of a TDB</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>The Definite Component $\Delta(T)$</td>
<td>15</td>
</tr>
<tr>
<td></td>
<td>Concreteness</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>The Propositional Component $PROP_T$</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>Lemma D.1 - Propositional Equivalence</td>
<td>18</td>
</tr>
<tr>
<td></td>
<td>Dead Clauses $DEAD_T$</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>Lemma D.2 - Dropping of Dead clauses</td>
<td>19</td>
</tr>
<tr>
<td></td>
<td>Theorem D.1 - $T - \Delta(T)$ Equivalence</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td>Lemma D.3 - Union</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>Lemma D.4 - Disjointness</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>Corollary D.1 - Disjoint Union</td>
<td>25</td>
</tr>
</tbody>
</table>
# Table of Contents

### Section E Transformation of a TDB to its Reduced Form  
- The Horn Component $H(T)$  
- Algorithm E.1 - $\text{DOMAIN}_T$  
- Sample Run of Algorithm E.1  
- Theorem E.1 - Algorithm E.1 Correctness  
- Corollary E.1 - $\Delta(T)$ - $\text{DOMAIN}_T$ Equivalence  
- The Reduced or Canonical TDB, $R(T)$  
- Lemma E.1 - Spanning  
- Theorem E.2 - $T - R(T)$ Equivalence  
- Corollary E.2  
- Corollary E.3 - Concreteness of $R(T)$  
- Corollary E.4 - Consistency of $R(T)$  

### Section F Updating a TDB  
- Algorithm F.1 - Update  
- Theorem F.1 - Algorithm F.1 Correctness  
- Sample Run of Algorithm F.1  

### Summary and Conclusions  

### Bibliography  

### Appendix A Dictionary of Symbols
Acknowledgements

I wish to thank Dr. Raymond Reiter of the University of British Columbia for his perceptive and critical analysis of the ideas that went into this paper. His dedication and insight into inferential database theory has been a continuing source of enlightenment.

I would also like to thank Bell Northern Research of Ottawa for the use of their word processing facilities in the preparation of this manuscript.
Section A  Introduction

In recent works on relational database systems, especially as they arise in artificial intelligence, the need for a database of domains (using Codd's definition of Domain[^3]) has become apparent. In simple data processing applications, the acceptance or rejection of data in a domain is based on a decision procedure for that domain such as correct alpha/numeric form or membership in a predefined set. This makes the primary assumption that each domain is independent of all others. This may be realistic for domains in which each entry of one domain is more or less unrelated to other domains (like numeric ones). However, work in artificial intelligence and inferential database systems acknowledges the dependencies among domains. Winograd led the way by using an ISA hierarchy to relate domains of objects and their properties. eg. a MOVEABLE-OBJECT ISA PHYSICAL-OBJECT[^15].

McSkimin and Minker formalize the ISA hierarchy by defining a semantic graph as a labelled tree in which the root node is the universal category (domain) and the leaves are primitive categories[^7,8,9]. In their formalism the elements, which are associated with the primitive categories, are inherited by the parent categories. As can be seen in their examples these trees tend to be "leafy" with many categories, some with unnatural names (such as non/human/male/mam). There is much duplication of category names but no mechanism is given for insuring that duplicate categories contain the same elements.
Also, McSkimin and Minker define a partitioning (disjointness condition between categories) without defining an update algorithm that will maintain the partition. In short, though they define the ISA heirarchy, they do not include a mechanism for enforcing data consistency.

Reiter has adopted the idea of a typed database formalized in the monadic predicate calculus\textsuperscript{[10,11]}. He replaces the idea of an element being a member of a domain with the proof theoretic notion of provability. Here the domain is a monadic predicate in which the element is its argument. The structure, mathematics and algorithms for this typed database are beyond the scope of his paper.

It is the purpose of this paper to propose a structure for Reiter's typed database and develop the necessary mathematics and algorithms.

A domain in Codd's terminology will hereafter be referred to as a TYPE. An element will be referred to as an INSTANCE. For example, HUMAN is a TYPE for which JOHN is an INSTANCE. JOHN is said to have the property HUMAN. A type database (TDB) is then a logical theory in which groups of TYPES (and hence INSTANCES) satisfy certain mathematical relationships called type relationships.

In order for a TDB to be practical it must satisfy certain conditions:

1) It must be descriptively rich, capturing type relationships other than simple set containment.
2) It must be structured so as to encourage only the most natural information content.

3) It must admit algorithms that are computationally efficient in both time and space.

4) It must admit updates in such a way as to prevent a user from violating a type relationship. (i.e. enforce integrity constraints)

Section B motivates the inclusion of type relationships other than set containment. Section C then defines a TDB. Condition 2 is addressed in section C which defines an acyclic TDB and introduces a graphic notation. Structuring is completed in section D by imposing the so-called concreteness condition on TDBs. With this and consistency all the desired set theoretic properties of the TDB follow. Section E is devoted to an algorithm for reducing a given TDB to one which is efficient in both time and space. The final section details an update algorithm which ensures that concreteness and consistency of the TDB are maintained during updates.
Section B Motivation

To see why a more robust set of type relationships than simple set containment is important consider the following statements:

1) JANE is a WOMAN; hence JANE is a FEMALE (ISA)
2) JANE is a FEMALE; hence JANE is a WOMAN
   or JANE is a GIRL (but not both)
3) a) JOHN is a BACHELOR; hence JOHN is a MAN (ISA)
   b) JOHN is a BACHELOR; hence JOHN is SINGLE (ISA)
4) JOHN is a MAN and JOHN is SINGLE; hence JOHN is a BACHELOR

Statements 3a and 3b bear a resemblance to statement 1 in that both display set containment. However, in view of statements 2 and 4 we see that somehow FEMALE is composed of WOMAN and GIRL - WOMAN and GIRL being disjoint types. In contrast MAN and SINGLE intersect to give BACHELOR. The standard ISA hierarchy does not provide the formal equivalent of statements 2 and 4.

An inference of type 2 is important in that we would not want JANE to have the property FEMALE without her having one of the properties WOMAN or GIRL. Similarly we would not want JANE to have both the properties WOMAN and GIRL. As seen in the following sections, "concreteness" satisfies the former condition, consistency the latter.

The type relationship of type 4 is important when we want to define a TYPE as having INSTANCES which are common to a set of TYPES. In an airline reservation system JOHN might have the properties NON-SMOKING, COACH,
MOVIE-WATCHING and 747 from which the system could infer that JOHN is a member of CABIN-B.

The way the above statements pair suggests the need for only two classes of type relationships:

A) The TYPE FEMALE is equivalent to the disjoint union of the TYPES WOMAN and GIRL

B) The TYPE BACHELOR is equivalent to the intersection of the TYPES MAN and SINGLE

The union in A need not be disjoint, for example:

C) The TYPE PEOPLE is equivalent to the union of the TYPES LEFT-HANDED and RIGHT-HANDED (allowing for the ambidextrous)

Graphically these are represented as:

A) \[
\begin{array}{c}
\text{FEMALE} \\
\bigtriangleup \\
\text{WOMAN} \\
\text{GIRL}
\end{array}
\]

B) \[
\begin{array}{c}
\text{MAN} \\
\bigtriangleup \\
\text{SINGLE} \\
\text{BACHELOR}
\end{array}
\]

C) \[
\begin{array}{c}
\text{PEOPLE} \\
\bigtriangleup \\
\text{RIGHT-HANDED} \\
\text{LEFT-HANDED}
\end{array}
\]

Set theoretically these can be represented as:

A) FEMALE = WOMAN \bigtriangleup GIRL

B) BACHELOR = MAN \cap SINGLE

C) PEOPLE = RIGHT-HANDED \cup LEFT-HANDED
In the graphical representation, the more restricted TYPE is drawn below the TYPE of which it is a subset. The four original statements can now be written in terms of sets:

1) JANE $\in$ WOMAN implies JANE $\in$ FEMALE
2) JANE $\in$ FEMALE implies JANE $\in$ WOMAN $\cup$ GIRL
3) JOHN $\in$ BACHELOR implies JOHN $\in$ MAN $\cap$ SINGLE
4) JOHN $\in$ MAN $\cap$ SINGLE implies JOHN $\in$ BACHELOR
The set notation of the previous section is convenient for visualizing the TYPE relationships. The first order predicate calculus, however, offers a well developed proof theory\cite{2,13}. For this reason we shall develop our results within a first order framework. Refer to appendix A for the interpretation given to special symbols used in what follows.

**Definition** TYPE Database (TDB)

A TDB is a 4-tuple \((C, P, I, E)\) satisfying:

1) \(C\) is a finite set of **constants**.

2) \(P\) is a finite set of unary predicate signs, called **TYPES**.

3) \(E\) is a finite set of atomic formulae of the form \(Pc\) where \(P \in P\) and \(c \in C\).

\(E\) is called the **extension** of the TDB.

4) \(I\) is a finite set of well formed formulae of the form:

   i) \((x) P x \equiv P_1 x \land P_2 x \land \ldots P_n x\) or

   ii) \((x) P x \equiv P_1 x \lor P_2 x \lor \ldots P_n x\)

     where \([P, P_1, P_2, \ldots P_n] \subseteq P\).

     In the event that \(I\) contains a formula of the form (ii), it may also contain all of the formulae:

     iii) \((x) \sim (P_i x \land P_j x)\) for each \(1 \leq i < j \leq n\).

     In which case the formula (ii) is said to be **exclusive**.

\(I\) is called the **intension** of the TDB.
For Example:

\[(C, P, I, E)\] is a TDB where:

\[C = \{\text{John, Mary}\}\] (INSTANCES)
\[P = \{\text{FEMALE, WOMAN, GIRL, MAN, SINGLE, BACHELOR}\}\] (TYPES)
\[I = \{(x) \text{BACHELOR } x \equiv \text{MAN } x \land \text{SINGLE } x, (x) \text{FEMALE } x \equiv \text{WOMAN } x \lor \text{GIRL } x, (x) \sim (\text{WOMAN } x \land \text{GIRL } x)\}\] (intension)
\[E = \{\text{WOMAN Mary, MAN John, SINGLE John}\}\] (extension)

Notation:

1) For brevity, the variable 'x' will be dropped where understood.
2) \[(C, P, I, E)\] will be referred to as T's constants, predicates, intension and extension where clarification is required.
3) Members of C will be written in lower case, members of P in upper case.
4) For reasons of notational convenience we sometimes write \[W \in I\] where \[W\] is actually a clause\[^{[13]}\].
5) A TDB will be said to be consistent whenever \[I \cup E\] is.
6) When we write \[T \models W\] we will mean \[I \cup E \models W\] for any formula \[W\].
7) \[R(U, V)\] will refer to the resolvents\(^{[13]}\) of the formulae \[U\] and \[V\].

With this definition of a TDB we can replace the notion of an INSTANCE being a member of a TYPE (\[\text{John } \in \text{BACHELOR}\]) with the predicate calculus notion of provability (\[I \cup E \models \text{BACHELOR John}\]). However, we would still like to view a type as a set with certain set theoretic relationships.
Definition  Extension

The extension of $P \in P$ written $||P||_T$ is defined:

$$||P||_T \equiv \{ c \mid c \in C, I \cup E \models Pc \}$$

The subscript on $||P||_T$ will be dropped where implicit. Hence we can write $c \in ||P||$ for $T \models Pc$. The set theoretic relationships between extensions of the last section do not automatically follow. We will give conditions under which each of the relationships given do follow.

Acyclicity of a TDB

The idea of a TYPE viewed as an ISA hierarchy is firmly related to the ideas of generalization and restriction. Logically we would like to think of one type containing another (ie. $||P|| \subseteq ||Q||$). This leads to the notion of a hierarchy of TYPES, motivating the following acyclicity definition[6].

Definitions  ALPHA, BETA and TAU

The relations $\alpha_T$, $\beta_T$ and $\tau_T$ are defined on a TDB, $T$, over $P \times P$:

1) $P \alpha_T Q$ iff $Q \equiv P_1 \lor P_2 \lor \ldots \lor P_n \in I$ such that $P = P_i$ for some $i$, $1 \leq i \leq n$

2) $P \beta_T Q$ iff $P \equiv Q_1 \land Q_2 \land \ldots \land Q_n \in I$ such that $Q = Q_i$ for some $i$, $1 \leq i \leq n$

3) $\tau_T \equiv \alpha_T \cup \beta_T$

The subscripts on $\alpha_T$, $\beta_T$ and $\tau_T$ will be dropped where implicit. A TDB is said to be acyclic iff $\tau_T^+$ is irreflexive. ie.
The graphical representation of Section B can be adopted with

representing \((x) \, P_x \equiv P_1 x \land P_2 x \land \ldots \land P_n x\) \((4i)\) and with

representing \((x) \, P_x \equiv P_1 x \lor P_2 x \lor \ldots \lor P_n x\) \((4ii)\) and finally

representing \((x) \, P_x \equiv P_1 x \lor P_2 x \lor \ldots \lor P_n x, \, (x) \sim (P_1 x \land P_j x)\) \((4ii \text{ and } 4iii)\).

Diagram C.1 gives an example of this graphical representation of I.

By convention, if \(P \text{ TAU } Q\) then \(P\) is directly below \(Q\) in the graph. Thus a TDB is acyclic when and only when its graph is acyclic. Throughout this paper the acyclicity of all TDBs is assumed.

The following lemma gives us the property of set containment desired:
Diagram C.1, A Sample I
Lemma C.1  Generalization

If $T$ is a TDB, $[P, Q] \subseteq P$ and $P \ TAU^+ Q$

then $T \vdash (x) Px \supset Qx$

Proof: By induction on $n$ it is sufficient to prove

$P \ TAU Q$ implies $T \vdash (x) Px \supset Qx$

Case $n = 1$  $P \ TAU Q$

Subcase a  $P$ ALPHA $Q$  then $Q \equiv P_1 \lor P_2 \lor ... \lor P_n \in I$

such that $P = P_i$, $1 \leq i \leq n$

In clausal form $P_i \lor Q \in I$ or $(x) Px \supset Qx \in I$

Subcase b  $P$ BETA $Q$  then $P \equiv Q_1 \land Q_2 \land ... \land Q_n \in I$

such that $Q = Q_i$, $1 \leq i \leq n$

In clausal form $P \lor Q_i \in I$ or $(x) Px \supset Qx \in I$

Case $n > 1$  $P \ TAUN Q$

Induction assumption:

$P \ TAUN^{n-1} Q_1$ implies $T \vdash (x) Px \supset Q_1x$ for any $Q_1 \in P$

Since $P \ TAUN Q$ then for some $Q_1 \in P$,

$P \ TAUN^{n-1} Q_1$ and $Q_1 \ TAU Q$

Then $T \vdash (x) Px \supset Q_1x$ by induction assumption and

$T \vdash (x) Q_1x \supset Qx$ by case $n = 1$

Therefore $T \vdash (x) Px \supset Qx$ QED
**Corollary C.1  Containment**

If \( T \) is a TDB, \([P, Q] \subseteq P\) and \( P \tau^+ Q\)

then \( \|P\| \subseteq \|Q\| \)

In terms of the graph of \( I \), we are assured that if a path exists between \( P \) and \( Q \) in a 'vertical' direction then every INSTANCE of the TYPE \( P \) is an INSTANCE of the TYPE \( Q \). We can see now why acyclicity is the only practical choice for if \( P \tau^+ Q \) and \( Q \tau P \) then \( \|P\| \subseteq \|Q\| \subseteq \|P\| \) or \( \|P\| = \|Q\| \).

This would certainly be of no use in a practical TDB.

The following lemma establishes the relationship between set intersection and part 4i in the definition of a TDB.

**Lemma C.2  Intersection**

For a TDB, \( T \), if \( P \equiv P_1 \wedge P_2 \wedge \ldots P_n \in I \)

then \( \|P\| = \|P_1\| \cap \|P_2\| \cap \ldots \|P_n\| \)

**Proof:**

**Case 1** \( c \in \|P\| \) (or \( T \vdash Pc \))

In clausal form \( \overline{P_x} \vee P_1 x \in I \)

for each \( i, 1 \leq i \leq n \)

By resolution \( T \vdash P_1 c \) implying \( c \in \|P_1\| \) or

\( c \in \|P_1\| \cap \|P_2\| \cap \ldots \|P_n\| \)
Case 2  \[ c \in \| P_1 \| \cap \| P_2 \| \cap \ldots \cap \| P_n \| \]

(or T |- P_i c for each i, 1 <= i <= n)

In clausal form \[ P \lor \neg P_1 \lor \neg P_2 \lor \ldots \lor \neg P_n \land c \}

By resolution T |- Pc implying c \in \| P \|

Section D develops a parallel result relating the union of extensions to the formula \[ P \equiv P_1 \lor P_2 \lor \ldots \lor P_n . \]
It has been recognized by Reiter\cite{11} and Minker\cite{9} that restricting inten-
sional information to Horn clauses (clauses containing at most one positive
literal) effectively controls the size of the proof search space without
overly constraining the expressiveness of the formalism. This can be seen
by viewing Horn clauses as procedures\cite{5} in a non-deterministic programming
language similar to MICRO-PLANNER\cite{14}. I.e., the formula,

\[ P_1 \land P_2 \land \ldots \land P_n \supset P, \]

could be programmed,

\[ \text{THE\_CONSEQUENT\_OF} (P_1, P_2, \ldots, P_n) \text{ IS } P; \]

and the clause,

\[ \overline{P_1} \lor \overline{P_2} \lor \ldots \lor \overline{P_n}, \]

could be programmed,

\[ \text{THE\_CONSEQUENT\_OF} (P_1, P_2, \ldots, P_n) \text{ IS INCONSISTENCY.} \]

Proofs done in this way are reasonably efficient as only unit consequents
are ever generated. Thus the maximum number of procedure firings is bounded
by the number of elements in P.

In this section we will consider clauses of the former form (definite
clauses) to develop the desired result on union of extensions.

\textbf{Definition} \hspace{1cm} \textbf{Definite Component}

Given a TDB, \( T = (C, P, I, E) \), the \textit{definite component} of \( T, \Delta(T) \), is
defined as the TDB, \( (C, P, \{ W \in I \mid W \text{ is a definite clause} \}, E) \).
Note that $E$ contains only definite clauses. We can now see exactly what parts of a TDB are definite:

<table>
<thead>
<tr>
<th>$W \in I$</th>
<th>Clausal Form</th>
<th>Definite?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\bar{P} \lor \bar{P}_2 \lor \ldots \lor \bar{P}_n$</td>
<td>$\bar{P} \lor \bar{P}_1$, $\bar{P} \lor \bar{P}_2$, $\ldots$, $\bar{P} \lor \bar{P}_n$</td>
<td>yes</td>
</tr>
<tr>
<td>$\bar{P}_1 \lor \bar{P}_2 \lor \ldots \lor \bar{P}_n \lor P$</td>
<td>$\bar{P}_1 \lor \bar{P}_2 \lor \ldots \lor \bar{P}_n \lor P$</td>
<td>yes</td>
</tr>
<tr>
<td>$\bar{P} \lor {P}_1 \lor {P}_2 \lor \ldots \lor {P}_n$</td>
<td>$\bar{P} \lor \bar{P}_1 \lor \bar{P}_2 \lor \ldots \lor \bar{P}_n$</td>
<td>no</td>
</tr>
<tr>
<td>$\bar{P} \lor Q$</td>
<td>$\bar{P} \lor Q$</td>
<td>no</td>
</tr>
</tbody>
</table>

We would like a result parallel to lemma C.2 but relating unions of extensions to the formula $P \equiv P_1 \lor P_2 \lor \ldots \lor P_n$. However, as the following example shows, it would not necessarily hold:

```
\begin{center}
\begin{tikzpicture}
    \node (female) at (0,0) {FEMALE};
    \node (woman) at (1,-1) {WOMAN};
    \node (girl) at (1,-2) {GIRL};
    \node (jane) at (0.5,-3) {\{jane\}};
    \draw (female) -- (woman);
    \draw (female) -- (girl);
    \draw (woman) -- (jane);
    \draw (girl) -- (jane);
\end{tikzpicture}
\end{center}
```

$C = \{\text{jane}\}$

$P = \{\text{FEMALE, GIRL, WOMAN}\}$

$I = \{\text{FEMALE} \equiv \text{WOMAN} \lor \text{GIRL},$
\hspace{1cm} $\sim (\text{WOMAN} \land \text{GIRL}) \}$

$E = \{\text{FEMALE jane}\}$

$$\|\text{FEMALE}\| = \{\text{jane}\} \quad \text{and} \quad \|\text{WOMAN}\| = \|\text{GIRL}\| = \{\}$$

since $T \not\models \text{WOMAN jane}$ and $T \not\models \text{GIRL jane}$.

So not only do we not have $\|\text{FEMALE}\| = \|\text{WOMAN}\| \cup \|\text{GIRL}\|$ but we have a TDB in which unnatural information is contained. It is hard to conceptualize jane as a "WOMAN or GIRL" without her being either a WOMAN or a GIRL. Actually,
jane is one of the two, we just can't determine which. For these reasons we insist on a TDB being "concrete".

**Definition Concreteness**

1) The formula $P \equiv P_1 \lor P_2 \lor \ldots \lor P_n \in I$ is **concrete** iff for every $c \in C$, $\Delta(T) \models P_c$ implies there is an $i$ such that $\Delta(T) \not\models P_i c$.

2) A TDB is **concrete** iff every formula of the above form is concrete.

We could have defined concreteness in terms of provability from $T$ rather than $\Delta(T)$. Unfortunately as we shall see, it turns out that at update time it would then be excessively difficult to maintain concreteness.

The disadvantage in the approach taken is that equivalence must be established between $T$ and $\Delta(T)$ before the result on union of extensions can be attained. Theorem D.1 accomplished this.

The Davis and Putnam result[^4] used in the proof of this theorem applies only for the propositional calculus. The following definition and lemma rectify this.

**Definition Propositional Component**

Given a TDB, $T$, and $c \in C$ we define the **propositional component** of $T$ with respect to $c$, $\text{PROP}_{Tc}$ as:

$$\text{PROP}_{Tc} \equiv [P | P_c \in E] \cup [W | (x)Wx \in I]$$

($(x)Wx$ represents any formula in $I$ with free variable $x$)
Using the last example:

\[ \text{PROP}_{T} \text{jane} = [\text{FEMALE}, \text{FEMALE} \equiv \text{WOMAN} \lor \text{GIRL}, \sim (\text{WOMAN} \land \text{GIRL})] \]

**Lemma D.1**  
**Propositional Equivalence**

Given a consistent TDB, T, c \( \in \) C and P \( \in \) P

then \( T \models P_c \) iff \( \text{PROP}_{Tc} \models P \)

**Proof:**

**Part 1** Assume \( E \cup I \cup [P_c] \) is unsatisfiable. By Herbrand's theorem, if we replace each clause in I with the set of clauses obtained by replacing the free variable 'x' with each member of C then the remaining set is unsatisfiable. Call this the ground form of I. Since \( I \cup E \) was originally consistent, the ground form of \( I \cup E \) is. \( [P_c] \) cannot resolve against any clause which does not have c's as its ground constant. Therefore all the constants in C except c are irrelevant to the inconsistency. If we construct the clause set

\[ [Q \mid Qc \in E] \cup [W \mid Wc \in \text{ground form of I}] \cup [P] \]

then we have a propositionally unsatisfiable set. Therefore,

\( \text{PROP}_{Tc} \models P \).

**Part 2** Assume \( \text{PROP}_{Tc} \cup [P] \) is unsatisfiable. Clearly

\( [Wc \mid W \in \text{PROP}_{Tc} \cup [P]] \) is unsatisfiable. This is a ground substitution of \( E \cup I \cup [P_c] \). Therefore \( T \models P_c \).
Theorem D.1 also requires that one factor out certain non-definite clauses. These are the ones that allow Jane to be a "WOMAN or GIRL" without the necessity of her being one of WOMAN or GIRL.

**Definition**  
**Dead Clauses**  
Given a TDB, $T$ and $c \in C$ then the **dead clauses** of $T$ with respect to $c$, $\text{DEAD}_{Tc}$ is defined:

$$\text{DEAD}_{Tc} \equiv [P \supset P_1 \lor P_2 \lor \ldots \lor P_n \in \text{PROP}_{Tc} \mid \text{PROP}_{\Delta(T)c} \not\vdash P, \ n > 1]$$

**Lemma D.2**  
**Dropping of Dead Clauses**  
Given a concrete TDB, $T$, $c \in C$ and $W \in \text{DEAD}_{Tc}$ then $\text{PROP}_{Tc} \sim \text{DEAD}_{Tc} \vdash W$.

**Proof:**

$$W = P \supset P_1 \lor P_2 \lor \ldots \lor P_n \in \text{DEAD}_{Tc}$$

and $\text{PROP}_{\Delta(T)c} \not\vdash P$.

By concreteness for some $i, 1 \leq i \leq n$,

$$\text{PROP}_{\Delta(T)c} \not\vdash P_i.$$  

$P_i$ subsumes $W$ so $\text{PROP}_{\Delta(T)c} \not\vdash W$.

Now $\text{DEAD}_{Tc}$ contains no definite clauses so

$\text{PROP}_{\Delta(T)c} \subset \text{PROP}_{Tc} \sim \text{DEAD}_{Tc}$ giving

$\text{PROP}_{Tc} \sim \text{DEAD}_{Tc} \vdash W$. QED
We will now prove equivalence of $T$ and $\Delta(T)$. The strategy will be to categorize the set of clauses produced by resolution within $PROP_T \sim DEAD_T$ and show that if a unit $Q_c$ is provable from $T$ then $Q$ will be in that set.
Theorem D.1  \( T - \Delta(T) \) Equivalence

Given a consistent concrete TDB, \( T, c \in C \) and \( Q \in P \)
then \( T \vdash Qc \iff \Delta(T) \vdash Qc. \)

Proof:

Assume \( \text{PROP}_{Tc} \vdash Q. \)
Then \( \text{PROP}_{Tc} \sim \text{DEAD}_{Tc} \vdash Q \) by Lemma D.2.
So \( \text{PROP}_{Tc} \sim \text{DEAD}_{Tc} U \overline{[Q]} \) is unsatisfiable.

Define the following sets iteratively:

\[
\begin{align*}
T_0 & = \text{PROP}_{Tc} \sim \text{DEAD}_{Tc} U \overline{[Q]} \\
T_{i+1} & = T_i \text{ when } Q \in T_i \text{ or } \\
& \text{ when } T_i \text{ contains no positive units, otherwise } \\
& \quad = T_i \sim [X v P v Y \varepsilon T_i] \sim [X v \overline{P} v Y \varepsilon T_i] U \\
& \quad [X v Y \mid X v \overline{P} v Y \varepsilon T_i] \\
& \text{ where } P \text{ is the lexicographically first positive unit} \\
& \text{ in } T_i \text{ and } X \text{ and } Y \text{ are negative predicate and positive} \\
& \text{ predicate disjunctions. (possibly null)}
\end{align*}
\]

The following is an invariant statement on \( T_i \):

If \( W \in T_i \) then

A) \( W = P \) such that \( \text{PROP}^{\Delta(T)c} \vdash P \) for some \( P \in P \) \text{ or }

B) \( W = P_j \) such that \( P \equiv P_1 v P_2 v \ldots P_n \equiv I \) \text{ is exclusive and }

\( \text{PROP}^{\Delta(T)c} \vdash P_i (i \neq j) \) \text{ or }

C) \( W = Q_1 \lor Q_2 \lor \ldots \lor Q_m \lor P \) (\( m \geq 1 \)) such that
\[
P \equiv P_1 \land P_2 \land \ldots \land P_n \in I \quad \text{and}
\]
\([Q_1, Q_2, \ldots, Q_m] \subseteq [P_1, P_2, \ldots, P_n]\) \quad \text{and}
\[
PROP_{\Delta(T)^c} \vdash W \quad \text{or}
\]

D) \( W \in T_0 \)

The invariant statement is proved by the following subproof:

Subproof: By induction on \( i \)

Case \( i = 0 \) Clear by condition D

Case \( i > 0 \) If \( W \in T_{i+1} \) then \( W \in T_i \) \quad \text{or}

\[
W = R(U, V) \quad \text{for} \quad [U, V] \subseteq T_i \quad \text{with} \quad U \quad \text{as the lexicographically first unit in} \quad T_i
\]

\[
PROP_{\Delta(T)} \vdash U \quad \text{by A and D above.}
\]

\( V \) is of the form B, C or D above:

B) \( V = \overline{P_j} \) such that \( P \equiv P_1 \lor P_2 \lor \ldots \lor P_n \in I \)
which is exclusive \quad \text{and}
\[
PROP_{\Delta(T)^c} \vdash \overline{P_i} \quad (i \neq j)
\]
\[
U = \overline{P_j} \quad \text{and} \quad PROP_{\Delta(T)} \vdash \overline{P_j}
\]
But \( \overline{P_i} \lor \overline{P_j} \in I \) by exclusiveness.
This violates the consistency of \( T \), so \( V \neq \overline{P_j} \).

C) \( V = Q_1 \lor Q_2 \lor \ldots \lor Q_m \lor P \) (\( m \geq 1 \))
\[
P \equiv P_1 \land P_2 \land \ldots \land P_n \in I \quad \text{and}
\]
\([Q_1, Q_2, \ldots, Q_m] \subseteq [P_1, P_2, \ldots, P_n]\) \quad \text{and}
\[
PROP_{\Delta(T)^c} \vdash V.
\]
U = Q_i for some i, 1 \leq i \leq m.

Clearly W has the same form as C with \( \text{PROP}^{\Delta(T)} \models W \)
unless \( m = 1 \) in which case \( W = P \) with \( \text{PROP}^{\Delta(T)} \models P \)
which is in the form A.

D) \( V \in \text{PROP}_T \sim \text{DEAD}_T \cup \overline{Q} \)

Case 1 \( V = \overline{P_1} \lor P_2 \in I \), trivially \( \text{PROP}^{\Delta(T)} \models V \).

\[ U = P_1 \] so \( \text{PROP}^{\Delta(T)} \models P_2 \) which is of form A.

Case 2 \( V = \overline{P_1} \lor P_2 \lor \ldots \lor P_n \lor P \in I \)

See C above.

Case 3 \( V = \overline{P_i} \lor \overline{P_j} \) (i < j)

where \( P \equiv P_1 \lor P_2 \lor \ldots \lor P_n \in I \) which is exclusive.

Assume without loss of generality that \( U = P_1 \).

Then \( W = \overline{P_j} \) which is in the form B.

Case 4 \( V = \overline{P} \lor P_1 \lor P_2 \lor \ldots \lor P_n, U = P \).

Since \( \text{PROP}^{\Delta(T)} \models P \) then \( V \in \text{DEAD}_T \).

But then \( V \notin T_0 \) which is a contradiction.

Case 5 \( V = \overline{Q} \)

But then \( U = Q \) was the lexicographically first unit in

\( T_i \) so by construction \( T_{i+1} = T_i \).

This completes the subproof of the invariant statement on \( T_i \).

As a corollary, \( T_i \) contains no non-unit positive clauses.

By the results of Davis and Putnam\(^[4]\), \( T_{i+1} \) is unsatisfiable whenever \( T_i \) is.

Since \( T_0 \) is unsatisfiable, by induction for all \( i \), \( T_i \) is unsatisfiable.
Also, $T_i$ contains a positive unit for every $i$, for if not then the interpretation in which every predicate is taken as false is a model for $T_i$.

Note that $T_{i+1}$ has at least one less predicate symbol than $T_i$ (the lexicographically first unit in $T_i$) unless $T_{i+1} = T_i$. Since $T_i$ always contains a positive unit and since $P$ is finite, then $T_{i+1} = T_i$ for some $i$. At this point $Q \in T_i$ since $T_i$ contains a positive unit. By the invariant statement on $T_i$,

$$\text{PROP} \Delta(T)^c \vdash Q \quad \text{QED.}$$

We can now relate part 4ii in the definition of a TDB to the union of extensions.

**Lemma D.3 Union**

Given a consistent concrete TDB, $T$ and a formula

$$P \equiv P_1 \lor P_2 \lor \ldots \lor P_n \in I$$

then $\|P\| = \|P_1\| \cup \|P_2\| \cup \ldots \cup \|P_n\|$.

**Proof:**

**Part 1** $c \in \|P\|$ or $T \vdash Pc$.

By theorem D.1, $\Delta(T) \vdash Pc$.

By concreteness, for some $i$, $\Delta(T) \vdash P_i^c$ giving $T \vdash P_i^c$ or $c \in \|P_i\|$.

**Part 2** $c \in \|P_i\|$ or $T \vdash P_i^c$ for some $i$.

In clausal form $\overline{P_i} \lor P \in I$.

By resolution $T \vdash Pc$ or $c \in \|P\|$.
Until now we have not differentiated between disjoint and non-disjoint union of extensions of the form \( P \equiv P_1 \lor P_2 \lor \ldots \lor P_n \in I \). However, in most (but not all) applications we would like disjointness strictly enforced.

**Lemma D.4 Disjointness**

Given a consistent TDB, \( T \), and an exclusive formula \( P \equiv P_1 \lor P_2 \lor \ldots \lor P_n \in I \)

then for each \( i, j, 1 \leq i < j \leq n \), \( \|P_i\| \cap \|P_j\| = \{\} \).

**Proof:** Let \( c \in \|P_i\| \cap \|P_j\| \).

Then \( T \models \neg P_i \land c \) and \( T \models \neg P_j \land c \).

But \( \neg P_i \lor \neg P_j \in I \).

This violates consistency, so no such \( c \) exists.

**Corollary D.1 Disjoint Union**

Given a consistent concrete TDB, \( T \), and an exclusive formula

\( P \equiv P_1 \lor P_2 \lor \ldots \lor P_n \in I \)

then \( \|P\| = \|P_1\| \oplus \|P_2\| \oplus \ldots \oplus \|P_n\| \).

In summary:

1) For conjunctions \( \|P\| = \|P_1\| \land \|P_2\| \land \ldots \land \|P_n\| \)

Furthermore if \( T \) is consistent and concrete:

2) For disjunctions \( \|P\| = \|P_1\| \lor \|P_2\| \lor \ldots \lor \|P_n\| \)

3) For exclusive disjunctions \( \|P\| = \|P_1\| \oplus \|P_2\| \oplus \ldots \oplus \|P_n\| \)
Section E  
Transformation of a TDB to its Reduced Form

One use of a TDB is to answer the question "Is $T \models P_c$". Reiter requires this for his "typed unification algorithm" as described in [10,11]. Algorithm E.1 below computes $[P \mid T \models P_c]$ given $c \in C$, for a concrete TDB, and detects any inconsistency in $\text{PROP}_{T,c}$.

As $[P \mid T \models P_c]$ may contain a large number of $P$ for any given $c$, we define a reduced form of $T$, $R(T)$, which contains only the $P_c$'s necessary to span $E$. This corresponds to associating INSTANCES with primitive semantic categories in [7,8,9]. Thus, we can determine if $c \in \|P\|$ by looking at the primitive categories below $P$.

We have shown that formulae of the form $\overline{P} \lor P_1 \lor P_2 \lor \ldots \lor P_n \in I$ can be removed with no effect when $T$ is both concrete and consistent. This motivates computing on the Horn component of concrete TDBs.

Definition  
Horn Component

Given a TDB, $T = (C, P, I, E)$, define the Horn component of $T$, $H(T)$, as

$$H(T) = (C, P, \{W \in I \mid W \text{ is a Horn clause}\}, E)$$

Note that $E$ contains only Horn clauses. Looking at this case by case:

<table>
<thead>
<tr>
<th>$W \in I$</th>
<th>Clausal Form</th>
<th>Definite?</th>
<th>Horn?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P \equiv P_1 \land P_2 \land \ldots \land P_n$</td>
<td>$\overline{P} \lor P_1, \overline{P} \lor P_2, \ldots, \overline{P} \lor P_n$</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>$P_1 \lor P_2 \lor \ldots \lor P_n \lor P$</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>$P \equiv P_1 \lor P_2 \lor \ldots \lor P_n$</td>
<td>$\overline{P} \lor P_1, \overline{P} \lor P_2, \ldots, \overline{P} \lor P_n \lor P$</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>$\overline{P} \lor P_1 \lor P_2 \lor \ldots \lor P_n$</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>$\neg (P \land Q)$</td>
<td>$\overline{P} \lor \overline{Q}$</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

The following algorithm computes $[P \mid T \models P_c]$ given $c \in C$
Algorithm E.1  
For computing $\text{DOMAIN}_{T_c}$ given $c \in C$

\[
T_0 \leftarrow \text{PROP}_{H(T)^c} \\
S_0 \leftarrow [ ] \\
\text{FOR } i = 0 \text{ to infinity DO} \\
\quad \text{BEGIN} \\
\quad \quad \text{IF } T_i \text{ contains no positive unit} \\
\quad \quad \quad \text{THEN } \text{DOMAIN}_{T_c} \equiv S_i, \\
\quad \quad \quad \quad k \leftarrow i, \text{ EXIT} \\
\quad \quad \text{P} \leftarrow \text{the lexicographically first positive unit in } T_i \\
\quad \quad \text{IF } \neg P \in T_i \\
\quad \quad \quad \text{THEN } \text{PROP}_{T_c} \text{ is inconsistent,} \\
\quad \quad \quad \quad \text{DOMAIN}_{T_c} \text{ is undefined, EXIT} \\
\quad \quad S_{i+1} \leftarrow S_i \cup [P] \\
\quad \quad T_{i+1} \leftarrow T_i \sim \left[ X \lor P \lor Y \in T_i \right] \sim \left[ X \lor \neg P \lor Y \in T_i \right] \\
\quad \quad \quad \quad U \left[ X \lor Y \mid X \lor \neg P \lor Y \in T_i \right] \\
\quad \quad \text{where } X \text{ and } Y \text{ are disjunctions of negative and positive predicates respectively (possibly null).} \\
\quad \text{END}
\]

Example: Using the I of diagram C.1 and letting:

\[ E = [\text{SINGLE john, MALE john}] \]

and $c = \text{john}$, the algorithm will produce the following values:
P

0

MALE
MAN ⊇ MALE

BOY ⊇ MALE

MALE ⊇ PERSON

~(MALE ∧ FEMALE)

MALE ∧ ADULT ⊇ MAN

MALE ∧ CHILD ⊇ BOY

1

PERSON

FEMALE ⊇ PERSON

ADULT ⊇ PERSON

CHILD ⊇ PERSON

2

SINGLE

BACHELOR ⊇ SINGLE

MAIDEN ⊇ SINGLE

SINGLE ⊇ ADULT

~(SINGLE ∧ MARRIED)

MAN ∧ SINGLE ⊇ BACHELOR

SINGLE ∧ WOMAN ⊇ MAIDEN

3

ADULT

MAN ⊇ ADULT

WOMAN ⊇ ADULT

MARRIED ⊇ ADULT

~(ADULT ∧ CHILD)

ADULT ⊇ MAN

FEMALE ∧ ADULT ⊇ WOMAN

4

MAN

BACHELOR ⊇ MAN

MAN ⊇ BACHELOR

5

BACHELOR

BACHELOR
T_6 contains no positive unit. Thus:

$$\text{DOMAIN}_{T\text{-john}} = \{\text{MALE, PERSON, SINGLE, ADULT, MAN, BACHELOR}\}$$

The algorithm can be implemented by hash indexing predicate symbols into a copy of the formulae in I in which they appear (only one copy is made for each formula). For efficiency, positive units are kept in a separate list. When a formula of the form $X \lor P \lor Y \in I$ is to be dropped it is tagged as deleted. When a formula of the form $X \lor \neg P \lor Y$ is to be replaced by $X \lor Y$ the predicate symbol $P$ is deleted from the copy. If a positive unit results it is added to the positive unit list. If the null clause results, $\text{DOMAIN}_{T\text{-c}}$ is declared to be undefined and the original clause $P \lor \neg Q$ that caused the contradiction can be printed for inspection.

If we assume that the time taken to delete the symbol $P$ from $X \lor \neg P \lor Y$ is constant, the time taken to add a symbol to the positive unit list is constant and the hash indexing is performed in constant time (by design) then the following is a bound on the time complexity[^1] of the algorithm:

$$O\left(\sum_{W \in \text{PROP}_H(T)^c} \# \text{predicate symbols in } W\right)$$

Theorem E.1 Algorithm E.1 Correctness

Given a concrete TDB, $T$, and $c \in C$

If $\text{PROP}_{T\text{-c}}$ is consistent

then $\text{DOMAIN}_{T\text{-c}} = \{P \mid T \models P\}$

otherwise $\text{DOMAIN}_{T\text{-c}}$ is undefined

Proof:

First, note that the class of Horn clauses is closed under resolution
(ie. \(R(W_1, W_2)\) is Horn whenever \(W_1\) and \(W_2\) are). It follows that the only positive clauses in \(T_i\) for all \(i\) are units. This is needed to apply the Davis and Putnam\(^4\) theorem below.

**Case A**  \(\text{PROP}_{T^c}\) is consistent

**Subcase 1**  \(P \in \text{DOMAIN}_{T^c}\).

For some \(i\), \(P \in T_i\) giving \(\text{PROP}_{H(T)^c} \vdash \overline{P}\)

and hence \(\text{PROP}_{T^c} \vdash \overline{P}\)

**Subcase 2**  \(T \vdash P_c\)

Assume \(P \notin T_i\) for all \(i\)

Define \(H'(T) = H(T) \cup \{\overline{P}\}\)

Starting the algorithm with \(H'(T)\) rather than \(H(T)\) we produce \(T_i'\) rather than \(T_i\).

Note that when defined,

\[T_i' = T_i \cup \{\overline{P}\}\]

The algorithm will terminate in one of two ways:

1) \([P, \overline{P}] \subseteq T_i'\) for some \(i\)

\([Q, \overline{Q}] \notin T_i'\) when \(Q \in P\) and \(Q \neq P\)

since \(T_0\) is consistent)

But then \(P \in T_i'\) contradiction

2) \(T_k'\) contains no positive unit; but then the interpretation in which every predicate symbol is false is a model for \(T_k'\). Since \(T \vdash P_c\) by theorem D.1, \(\Delta(T) \vdash P_c\) giving \(H(T) \vdash P_c\) or \(T_0 \cup \{P\} = T_0\) is unsatisfiable. By Davis and Putnam\(^4\), \(T_k'\) is unsatisfiable which is a contradiction. Therefore \(P \in T_i\) for some \(i\).

The algorithm must terminate since \(T_{i+1}\) contains at least one fewer predicate symbol than \(T_i\).

**Case B**  \(\text{PROP}_{T^c}\) is unsatisfiable

**Statement:**  \(\text{PROP}_{H(T)^c}\) is unsatisfiable

**Subproof:** By lemma D.2 \(\text{PROP}_{T^c} \sim \text{DEAD}_{T^c}\) is unsatisfiable. If
we define $T_0$ as $\text{PROP}^c \dashv \text{DEAD}^c$ and let the algorithm define $T_i$ for each $i$, then the invariant statements of theorem D.1 hold with one addition: $W \in T_i$ may be the null clause.

The algorithm will terminate in one of two ways:

1) $[P, \overline{P}] \subseteq T_i$ for some $i$ where $P$ is the lexicographically first unit in $T_i$. By statements A and B of theorem D.1,

$$\text{PROP}^c \Delta(T) \vdash P$$

and

for some $Q \in P$, $\overline{P} \lor \overline{Q} \in I$ and $\text{PROP}^c \Delta(T) \vdash Q$.

Since $\overline{P} \lor \overline{Q} \in H(T)$ and $\Delta(T) \subseteq H(T)$, $\text{PROP}^c H(T)$ is inconsistent.

2) By the results of Davis and Putnam[4], for each $i$, $T_i$ is unsatisfiable. Since $T_{i+1}$ contains at least one less literal than $T_i$, for some $k$, $T_k$ contains no positive literals. The interpretation in which every predicate symbol is false is a model for $T_k$ unless $T_k$ contains the null clause. Since $T_k$ is unsatisfiable it must contain the null clause. $T_0$ doesn't contain the null clause so for some $i$, $[P, \overline{P}] \subseteq T_i$ where $P$ is the lexicographically first unit $T_i$. In this case though, the algorithm will have terminated as in case A.

This ends the subproof.

As in the subproof, the algorithm will not terminate with $T_k$ containing no units (Starting with $T_0 = \text{PROP}^c H(T)$ unsatisfiable).

Therefore, $[P, \overline{P}] \subseteq T_i$ for some $i$. The algorithm will stop with $\text{DOMAIN}^c_T$ undefined. QED

This ends the correctness proof of algorithm E.1.
The following corollary is used in the proof of the update algorithm.

**Corollary E.1**  \( \Delta(T) - \text{DOMAIN}_T \) Equivalence

Given a TDB, \( T \) (not necessarily concrete or consistent) and \( c \in C \)

1) If \( P \in \text{DOMAIN}_{Tc} \) then \( \Delta(T) \vdash Pc \)

2) If \( \Delta(T) \vdash Pc \) then \( P \in \text{DOMAIN}_{Tc} \) or \( \text{DOMAIN}_{Tc} \) is undefined

**Proof:**

**Part 1** \( P \in \text{DOMAIN}_{Tc} \)

The invariant statements of theorem D.1 hold with the addition that \( W \) may be the null clause.

Therefore \( \Delta(T) \vdash Pc \)

**Part 2** \( \Delta(T) \vdash Pc \)

Hence \( H(T) \vdash Pc \).

If \( T \) is consistent then by theorem E.1, Case A, subcase 2,

\( P \in T_i \) for some \( i \) (we do not need theorem D.1 here) or

\( P \in \text{DOMAIN}_{Tc} \).

If \( T \) is unsatisfiable then by theorem E.1, Case B, \( H(T) \) is unsatisfiable and \( \text{DOMAIN}_{Tc} \) is undefined. QED

As a TDB, \( T \), becomes large, \( \text{[DOMAIN}_{Tc} \mid c \in C \) may become very large. The following allows us to save only the \( Pc \) that span \( E \). I.e., store only the 'lowest' extensions in the graph.

**Definition**  The Reduced or Canonical TDB

Given a consistent concrete TDB, \( T = (C, P, I, E) \), the reduced TDB, \( R(T) \), is defined:

\[
R(T) = (C, P, I, \{Pc \mid P \in \text{DOMAIN}_{Tc} \text{ and if } Q \in \text{DOMAIN}_{Tc} \text{ then } \not\triangleright Q \text{ and } \not\triangleright P})
\]
Using the example of diagram C.1:

\[ [P \mid Pc \in E] = [\text{SINGLE, MALE}] \]
\[ \text{DOMAINT}_{Tc} = [\text{MALE, PERSON, SINGLE, ADULT, MAN, BACHELOR}] \]

but \[ [P \mid Pc \in R(T)\text{'s extension}] = [\text{BACHELOR}] \]

This is a good space reduction. In fact, in no case is \( R(T)\text{'s extension} \) larger than \( T\text{'s} \). Note that we cannot assume that the concreteness or consistency of \( R(T) \) follows from that of \( T \).

**Lemma E.1**  
**Spanning**

Given a consistent concrete TDB, \( T \),

if \( P \in \text{DOMAIN}_{Tc} \) then for some \( Q \in P \) with \( Qc \in R(T)\text{'s extension} \),

\[ Q \# P \]

Proof: Assume for each \( n \geq 0 \) with \( Qc \in R(T)\text{'s extension} \), that

not \( Q \# P \)

By definition of \( R(T)\text{'s extension} \), \( Pc \in R(T)\text{'s extension} \)

Choosing \( n = 0, Q = P \) we have a contradiction for \( Q \# P \).

\( R(T)\text{'s extension} \) is said to span \( [Pc \mid T \vdash Pc] \).

Getting back to the original problem, "Is \( T \vdash Pc? \)" we see that we need only scan \( R(T)\text{'s extension} \) for \( Qc \) such that \( Q \# P \).

**Theorem E.2**  
**\( T \vdash R(T) \) Equivalence**

Given a consistent concrete TDB, \( T \),

then if \( Pc \in R(T)\text{'s extension} \) then \( T \vdash Pc \)

and if \( Pc \in E \) then \( R(T) \vdash Pc \)

Proof:

Part 1) Assume \( Pc \in R(T)\text{'s extension} \).

\( P \in \text{DOMAIN}_{Tc} \) so by theorem E.1, \( T \vdash Pc \)
Part 2) Assume $P_c \in E$

or $P \in \text{DOMAIN}_{Tc}$.

By the spanning lemma (E.1) for some $Q \in P$,

such that $Q_c \in R(T)$'s extension, $Q \tau\alpha^* P$.

By the generalization lemma (C.1), $R(T) \models P_c$ QED

Corollary E.2

Given a consistent concrete TDB, $T$, $c \in C$, $P \in P$,

$T \models P_c$ iff $R(T) \models P_c$

Corollary E.3  Concreteness of $R(T)$

Given a consistent concrete TDB, $T$, $R(T)$ is concrete

Proof:

Let $P \equiv P_1 \vee P_2 \vee \ldots \vee P_n \in I$

Assume for some $c \in C, \Delta(R(T)) \models \neg P_c$

Then $T \models P_c$ by theorem E.2

and $\Delta(T) \models P_c$ by theorem D.1

and $\Delta(T) \models P_{1c}$ by $T$'s concreteness (for some $i$, $1 \leq i \leq n$)

This gives $P_{1c} \in \text{DOMAIN}_{Tc}$ by theorem D.1

By the spanning lemma (E.1), for some $Q_c \in R(T)$'s extension,

$Q \tau\alpha^* P_1$

By the generalization lemma (C.1) (which resolves only definite clauses), $\Delta(R(T)) \models P_{1c}$ QED

Corollary E.4  Consistency of $R(T)$

Given a consistent concrete TDB, $T$, $R(T)$ is consistent

Proof:

Let $S = \{P_c \mid P \in \text{DOMAIN}_{Tc}\} \cup$

$\overline{\{P_c \mid P \in P \text{ and } P \in \text{DOMAIN}_{Tc}\}}$

$S$ is a model for $H(T)[12]$. 
If it is not a model for $T$ then for some

$$P \equiv P_1 \lor P_2 \lor \ldots \lor P_n \in I,$$

$P \in \text{DOM}_{T^C}$ and each $P_i \in \text{DOM}_{T^C}$, $1 \leq i \leq n$

This contradicts concreteness, so $S$ is a model for $T$.

$R(T)$ is concrete so similarly $S$ is a model for $R(T)$. QED

Thus $R(T)$ is a suitable replacement for a consistent TDB, $T$. 
Section F  Updating a TDB

Thus far we have shown that a TDB will have the desired set theoretic properties if concreteness and consistency are maintained. We have given a canonical (reduced) form of the TDB that can be efficiently computed while conserving both space and time in answering "Is T \models Pc?". We must now specify an algorithm that will update the canonical TDB while guaranteeing concreteness and consistency.

Given a concrete and consistent TDB, T, and c \in C the algorithm accepts additions to and deletions from DOMAIN_T^c so as to leave the resulting TDB concrete and consistent. It is assumed that originally T = R(T). The algorithm will produce T' = R(T').

Algorithm F.1 For updating a canonical TDB T, given c \in C

UPDATE_T^c:
BEGIN
WRITE (c, "has the root properties:", [P | Pc \in E])
S: INPUT (A \subset P (additions) and D \subset P (deletions))
IF [Pc | P \in D] \notin E THEN
WRITE (D \sim [P | Pc \in E], "cannot be deleted, try again")
GOTO S
L: T' \leftarrow (C, P, I, E U [Pc | P \in A] \sim [Pc | P \in D])
Calculate DOMAIN_T,c
CONCRETE \leftarrow TRUE
E: IF DOMAIN_T,c is undefined (because \overline{P} v \overline{Q} \in I) THEN
WRITE (c, "cannot be both a", P, "and a", Q, "try again")
GOTO S
C: FOR EACH $P \equiv P_1 \lor P_2 \lor \ldots \lor P_n \in I$ such that 
\begin{align*}
P & \in \text{DOMAIN}_{T'}^c \text{ but no } P_i \in \text{DOMAIN}_{T'}^c \\
\text{WRITE ("A", } P, \text{ "must also be one of", } P_1, P_2, \ldots P_n, \\
\text{"add one of them")}
\end{align*}

CONCRETE $\leftarrow$ FALSE

IF NOT CONCRETE THEN

INPUT ($A' \subseteq P$ (additions))

$A \leftarrow A \cup A'$

GOTO L

Calculate $R(T')$ given $\text{DOMAIN}_{T'}^c$ and $T'$

$T' \leftarrow R(T')$.

WRITE ("update accepted")

END

Note that in the step marked $\ast$, $\text{DOMAIN}_{T'}^c$ for $c_1 \not= c$ need not 
be recalculated as $\text{DOMAIN}_{T'}^c = \text{DOMAIN}_{T}^c$. The following theorem 
gives the update properties desired.

**Theorem F.1**  
**Algorithm F.1 Correctness**

Given a consistent concrete and canonical TDB, $T$ and $c \in C$ then the 
resultant canonical TDB, $T'$, is consistent and concrete.

**Proof:**

**Part 1  Concreteness**

Let $P \equiv P_1 \lor P_2 \lor \ldots \lor P_n \in I$ and $\Delta(T') \vdash P_c$

By corollary E.1, since $\text{DOMAIN}_{T'}^c$ is defined in order for 
algorithm termination, $P \in \text{DOMAIN}_{T'}^c$. Since the algorithm 
terminated (CONCRETE is TRUE), for some $i$, $P_i \in \text{DOMAIN}_{T'}^c$.

Corollary E.1 gives $\Delta(T') \vdash P_i c$.  

Part 2  Consistency

If $T'$ is not consistent then since $T$ is consistent,
PROP$_{T'}$c must be unsatisfiable. Theorem E.1 applies since $T'$ is concrete saying DOM$_{T'}$c is undefined. This contradicts termination of the algorithm. Therefore $T'$ is consistent.

By corollaries E.3 and E.4, R($T'$) is concrete and consistent. QED

The following is a sample update using the I of diagram C.1 and an empty $E$.

<table>
<thead>
<tr>
<th>Input</th>
<th>Response</th>
<th>$E$ or $E'$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Update john</td>
<td>john has the root properties [ ]</td>
<td>$E = [ ]$</td>
</tr>
<tr>
<td>$A=[ADULT]$</td>
<td>An ADULT must be one of MARRIED or SINGLE, add one.</td>
<td>$E' = [ADULTjohn]$</td>
</tr>
<tr>
<td>$A=[MALE, SINGLE]$</td>
<td>Update accepted</td>
<td>$E' = [ADULTjohn, MALEjohn, SINGLEjohn]$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$E = [BACHELORjohn]$</td>
</tr>
<tr>
<td>Update john</td>
<td>john has the root properties [BACHELOR]</td>
<td>$E = [BACHELORjohn]$</td>
</tr>
<tr>
<td>$A=[MARRIED]$ and $D=[SINGLE]$</td>
<td>[SINGLE] cannot be deleted, try again</td>
<td>$E' = [MARRIEDjohn, BACHELORjohn]$</td>
</tr>
<tr>
<td>$A=[MARRIED]$</td>
<td>john cannot be both MARRIED and SINGLE, try again</td>
<td>$E' = [MARRIEDjohn]$</td>
</tr>
<tr>
<td>$A=[MARRIED]$ and $D=[BACHELOR]$</td>
<td>An ADULT must be one of MALE or FEMALE, add one</td>
<td>$E' = [MARRIEDjohn]$</td>
</tr>
<tr>
<td>$A=[MALE]$</td>
<td>Update accepted</td>
<td>$E' = [MARRIEDjohn, MALEjohn]$</td>
</tr>
</tbody>
</table>
It may seem redundant that we must "Add MALE" when we did not explicitly delete MALE. However, when BACHELOR is deleted, the algorithm has no way in general of determining if MAN and/or SINGLE are to remain in the TDB. Rather than trying to second guess the user, the algorithm asks him to correct any dropped information.

Before the first update, when $E = \emptyset$, for the update theorem to apply, $T$ must be concrete and consistent. It is certainly concrete. In order for it to be consistent, $I$ must be consistent. The interpretation in which all predicate symbols are false under the interpretation is a model for $I$, so it is consistent. Thus by induction on the update theorem we are assured that if we start with $E = \emptyset$ then $T$ will always be concrete and consistent after the update.

Although there are no inconsistencies in $I$, there may be constraints that prevent certain kinds of updates from taking place. By this we mean that given some $P \in P$, for no $c \in C$ is $Pc \in E$ while $T$ is consistent. I.e. $\|P\| = \emptyset$ always.

For example, consider the following:

![Diagram](image)

Clearly, $I \models (x) \overline{P_1}x$.

Therefore if the TDB is to be kept consistent, $\|P_1\| = \emptyset$ always. This is clearly a case of bad structuring of $I$. It can be avoided by ensuring that
for all \( P \in P \), \( I \cup [P_x] \) is consistent. This can be guaranteed by letting \( E = [P_c] \) for any \( c \in C \) and calculating \( \text{DOMAIN}_{Tc} \). If undefined, then \( I \) is badly structured above \( P \).
Summary and Conclusions

A generalized structure has been proposed for Reiter's typed database \[10\] to replace the more restricted notion of the ISA hierarchy. It provides a rich structuring facility and imposes only the natural requirements of so-called concreteness and consistency. Results have been given to ensure that the database has the desired set theoretic properties. An update algorithm has been specified that guarantees concreteness and consistency along with a technique for testing if a specific element is a member of a given domain. A means for testing a non-atomic formula against the database was not included and is an open topic.

While traditional relational databases suggest no need for a typed database, work in artificial intelligence and inferential database systems often are deficient because of the need for a more interdependant domain structure. Since the TDB provides the most rudimentary level of inferencing it must be well structured and mathematically complete in order that inferencing mechanisms built on it stand on firm ground. It must not limit the descriptive power of a host inferential database system, thereby reducing its applicability. We suggest that the proposed structure and algorithms for a typed database are adequate for managing the domain structure of inferential database systems.
Bibliography


Appendix A  Dictionary of Symbols

Sets:

[ ] - Used in set definitions as a replacement for { }
ε, ∈ - Set membership and non-membership
⊂, ⊊ - Set inclusion and non-inclusion
∩, ∪ - Set intersection and union
⊕ - Disjoint union
∼ - Set difference
| - Set qualification (read "such that")
∥p∥ - Extension of a predicate, (see section C)

Predicate Calculus Formulae:

∧ - Conjunction symbol
∨ - Disjunction symbol
⇒ - Implication
≡ - Equivalence
¬ - Negation
¬P - Also negation
(x) - Universal quantification
|- - Symbols for provability and non-provability
R(W₁, W₂) - The resolvents of formulae W₁ and W₂[13]
Mathematical Relations:[6]

ALPHA, BETA and TAU

- Relations defined in section C

$R^n$ - Composition of relation $R$, n times

$R^+$ - Transitive closure of relation $R$

$R^*$ - Reflexive transitive closure of relation $R$

General Usage:

$\equiv$ - read "is defined as", in definitions and algorithms

$\bigcirc$ - computational complexity of an algorithm (order complexity)[1]