Equality and Domain Closure in First-Order Databases

RAYMOND REITER

University of British Columbia, Vancouver, British Columbia

ABSTRACT A class of first-order databases with no function signs is considered. A closed database DB is one for which the only existing individuals are those explicitly referred to in the formulas of DB. Formally, this is expressed by including in DB a domain closure axiom $(x)x = c_1 \vee \cdots \vee x = c_p$, where $c_1, \ldots, c_p$ are all of the constants occurring in DB. It is shown how to completely capture the effects of this axiom by means of suitable generalizations of the projection and division operators of relational algebra, thereby permitting the underlying theorem prover used for query evaluation to ignore this axiom

A database is E-saturated if all of its constants denote distinct individuals. It is shown that such databases circumvent the usual problems associated with equality, which arise in more general databases.

Finally, it is proved for Horn databases and positive queries that only definite answers are obtained, and for databases with infinitely many constants that infinitely long indefinite answers can arise.

KEY WORDS AND PHRASES: equality, closed databases, deductive question-answering, first-order logic, theorem proving, relational databases, projection operator, division operator, definite answers, indefinite answers

CR CATEGORIES 3.69, 3.79, 5 21

1. *Introduction*

It is well known that the equality relation presents severe computational problems for general-purpose first-order theorem provers [2, 18]. Since deductive question-answering systems [e.g., 7] appeal to theorem provers for answering queries, we can expect these same problems to arise for general first-order databases. The purpose of this paper is to define a natural restricted class of databases, specifically databases with no function signs, and to investigate their properties with respect to the equality relation.

A common feature of deductive question-answering systems [e.g., 3] is that the database DB is treated as closed. By this we mean that the only existing individuals are those explicitly referred to in the formulas of DB. Formally, this is expressed by including in DB a domain closure axiom $(x)x = c_1 \vee \cdots \vee x = c_p$, where $c_1, \ldots, c_p$ are all of the constants occurring in DB. The presence of such an axiom would normally create severe difficulties for a first-order theorem prover in view of the fact that in most applications the number of constants will be very large. One of the results of this paper is that this axiom may be omitted without affecting the answers to a query provided that we invoke suitable generalizations of the projection and division operators of relational algebra theory. Another way of viewing this result is that a closed, function-free, first-order database is an appropriate generalization of conventional relational databases [e.g., 6].

A characteristic of first-order databases is that there can arise so-called indefinite answers to queries, i.e., answers x of the form x is A_1 or x is A_2 or ..., but there is not enough

This paper was written with the financial support of the National Research Council of Canada under Grant A7642 Much of this research was done while the author was visiting at Bolt, Beranek and Newman Inc, Cambridge, Mass

Author's address Department of Computer Science, University of British Columbia, Vancouver, British Columbia, V6T 1W 5, Canada

information available to characterize x more precisely. The possibility of such answers considerably complicates the calculation of the projection and division operators mentioned earlier. We prove in this paper that for a large and natural class of databases, namely the Horn databases, no such indefinite answers can arise, in which case the projection and division operators assume particularly simple forms.

It is often natural to view a database as being E-saturated, i.e., all of the constants of the database denote distinct individuals. We show that when this is the case, the usual problems associated with the equality relation do not arise for existentially quantified queries.

We also consider a natural generalization of databases to those with countably infinitely many constants and show that certain pathologies can then arise. Specifically, we exhibit a closed E-saturated database with infinitely many constants, together with a query which has an infinitely long indefinite answer.

We conclude with a discussion in which we argue

(i) that the function-free assumption for databases, together with domain closure, are both necessary for a computationally feasible treatment of equality, and
(ii) that the theory of this paper will accommodate functions provided they are total, so that the function-free assumption is not quite so severe as it initially appears to be.

2. *Formal Preliminaries*

We shall be dealing with a first-order language with equality but *without* function signs. Hence, assume as 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).

(4) *Predicate signs*: $P, Q, R, \ldots, =$. 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, as well as various Boolean combinations of these called types, will be used to restrict the allowable ranges of variables occurring in queries and in the database.
(ii) The class of remaining predicate signs, which will be called *common predicate signs*. In particular, there is a distinguished binary common predicate sign, $=$, which will function as the equality predicate. In the intended interpretation, common predicate signs will denote database relations; for example, the supplies relation in an inventory domain might be denoted by the common predicate sign SUPPLIES where SUPPLIES(x, y) denotes "x supplies y."

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

(a) A simple type is a type.
(b) If τ_1 and τ_2 are types, so also are $\tau_1 \wedge \tau_2$, $\tau_1 \vee \tau_2$, $\bar{\tau}_1$.

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, τ a nonsimple type, and τ_1 and τ_2 types, then

(i) if τ is $\tau_1 \wedge \tau_2$, $\tau(t)$ is $\tau_1(t) \wedge \tau_2(t)$,
(ii) if τ is $\tau_1 \vee \tau_2$, $\tau(t)$ is $\tau_1(t) \vee \tau_2(t)$,
(iii) if τ is $\bar{\tau}_1$, $\tau(t)$ 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 DATABASE FORMULAS. We define the following syntactic objects:

(1) *Terms.* A *term* is either a variable or constant sign.

(2) *Literals.* If P is an n-ary predicate sign and $t_1, \ldots, t_n$ are terms, then $P(t_1, \ldots, t_n)$ and $\sim P(t_1, \ldots, t_n)$ are *literals.* They are *common literals* iff P is a common predicate sign. They are *type literals* iff P is a simple type (and hence a unary predicate sign).

(3) *τ-wffs.* The set of τ-wffs is the smallest set satisfying:

(i) A type literal is a τ-wff.
(ii) If W_1 and W_2 are τ-wffs, 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 τ-wff, so also are $(x)W$ and $(Ex)W$.

(4) *Typed well-formed formulas (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 τ a type, then $(x)[\tau(x) \supset W]$ and $(Ex)[\tau(x) \wedge W]$ are twffs. These are denoted by $(x/\tau)W$ and $(Ex/\tau)W$, respectively. (x/τ) is a *restricted universal quantifier*, and (Ex/τ) is a *restricted existential quantifier.*

2.2 DATABASES. A database is defined to consist of two disjoint components—a type database of τ-wffs where all information about types resides, and a principal database of twffs which contains all information about the remaining predicates of the theory. In addition, we require that no formulas of a database contain an existential quantifier in its prenex normal form. This means that no Skolem functions are introduced when database formulas are converted to quantifier-free clausal form. Since we have begun by assuming a function-free, first-order language, we do not have to deal with functions either implicitly (via their introduction as Skolem functions) or explicitly (as objects permitted by the first-order language). As we shall see (Sections 4 and 7 below), this "function-free assumption" has important consequences for a computationally feasible treatment of equality.

Formally then, a *principal database* is any finite set of twffs whose prenex normal forms contain no existential quantifiers and which contains the following equality axioms:

E1. $(x)x = x$.
E2. $(x)(y)x = y \supset y = x$.
E3 $(x)(y)(z)x = y \wedge y = z \supset x = z$.
E4. For each n-ary predicate sign P,

$$(x_1) \cdots (x_n)(y_1) \cdots (y_n)x_1 = y_1 \wedge \cdots \wedge x_n = y_n \wedge P(x_1, \ldots, x_n) \supset P(y_1, \ldots, y_n).$$

Let T be a set of τ-wffs. T is said to be *τ-complete* iff for each constant sign c and each simple type τ, $T \vdash \tau(c)$ or $T \vdash \sim\tau(c)$ where, in general, $W \vdash A$ denotes that A is provable in the first-order theory W. A *type database* (TDB) is any τ-complete finite set of τ-wffs with the property that when transformed to prenex normal form, no τ-wff has an existential quantifier.

In the intended application a TDB will represent a classification scheme for the objects in the domain being represented. For example, a particular payroll application might require the following types: HUMAN, EMPLOYEE, MANAGER, DEPARTMENT, etc. The TDB would then contain appropriate general facts about these types, such as

(x)EMPLOYEE$(x) \supset$ HUMAN(x)
(x)MANAGER$(x) \supset$ EMPLOYEE(x)
(x)HUMAN$(x) \supset \sim$DEPARTMENT(x)
etc.,

as well as specific facts like

MANAGER(John-Doe)
DEPARTMENT(103)
etc.

The τ-completeness assumption is thus the appropriate formalization of the requirement that for each object and for all classes, we know to which classes it belongs and to which it does not belong. In the payroll example we can infer that John-Doe is a MANAGER, and EMPLOYEE, a HUMAN, and not a DEPARTMENT. If all we know about Mary-Smith is that she is an EMPLOYEE, we would not have τ-completeness, since her status as a MANAGER remains unknown.

We are not seriously proposing that in an implementation of a question-answering system, the TDB be represented as a set of τ-wffs. There are far more efficient and perspicuous representations of the same facts. One such representation involving semantic networks is thoroughly discussed in [12, 13]. A different approach is described in [1]. 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 formulas of the monadic predicate calculus. As is well known [9], the monadic predicate calculus is decidable, i.e., there exists an algorithm which for any τ-wff W determines whether or not TDB $\vdash W$. This must remain true regardless of how the TDB is represented. Henceforth we assume the availability of such a decision procedure for the TDB. An efficient decision procedure for a large and natural class of TDBs is described in [1].

If τ is a type, define $|\tau|_{\text{TDB}} = \{c \mid c \text{ is a constant sign and TDB} \vdash \tau(c)\}$. When the TDB is clear from context, we write $|\tau|$ instead of $|\tau|_{\text{TDB}}$. If $\tau = \tau_1, \ldots, \tau_n$ is a sequence of types, define $|\tau| = |\tau_1| \times \cdots \times |\tau_n|$.

The notion of a type database as applied to deductive question-answering has been independently proposed in [12, 13]. What we have been calling simple types and types, McSkimin and Minker [13] call primitive categories and Boolean category expressions, respectively. While McSkimin and Minker do not explicitly make the τ-completeness assumption, it appears to be implicit in the ways they use the type database. We shall see (Section 2.4 below) why τ-completeness is a highly desirable property of the TDB.

Finally then, if TDB and PDB are type and principal databases, respectively, TDB $\cup$ PDB is defined to be a *database.*

Example 2.1. The following is a simple fragment of an education database with simple types COURSE, CALCULUS, CS, HISTORY, TEACHER, STUDENT:

TDB

An axiom specifying that calculus, computer science, and history are each courses:

$$(x)\text{CALCULUS}(x) \vee \text{CS}(x) \vee \text{HISTORY}(x) \supset \text{COURSE}(x)$$

Axioms specifying the pairwise disjointness of appropriate simple types, e.g.,

$$(x) \sim (\text{COURSE}(x) \wedge \text{TEACHER}(x))$$
$$(x) \sim (\text{TEACHER}(x) \wedge \text{STUDENT}(x))$$
etc.

Simple type instances:

$$|\text{CALCULUS}| = \{\text{C100, C200}\}$$
$$|\text{CS}| = \{\text{CS100, CS200, CS300}\}$$
$$|\text{HISTORY}| = \{\text{H100, H200}\}$$
$$|\text{TEACHER}| = \{A, B, C\}$$
$$|\text{STUDENT}| = \{a, b, c, d\}$$

PDB

(1) A teaches all calculus courses.

$$(x/\text{CALCULUS})\text{TEACH}(A, x)$$

(2) B teaches all computer science courses.

$(x/\text{CS})\text{TEACH}(B, x)$

(3) If teacher x teaches course y and student z is enrolled in y, then x is a teacher of z.

$(x/\text{TEACHER})(y/\text{COURSE})(z/\text{STUDENT})\text{TEACH}(x, y) \wedge \text{ENROLLED}(z, y) \supset \text{TEACHER-OF}(z, x)$

TEACH	
C	H100
C	H200

ENROLLED	
a	C100
a	CS100
b	C200
b	CS200
b	CS300
c	C100
c	H200
d	H100

2.3 QUERIES. If DB is a database, then a *query* for DB is any expression of the form

$$\langle x_1/\tau_1, \ldots, x_n/\tau_n \mid (q_1 y_1/\theta_1) \cdots (q_m y_m/\theta_m) W(x_1, \ldots, x_n, y_1, \ldots, y_m)\rangle,$$

where $(q_i y_i/\theta_i)$ is (y_i/θ_i) or (Ey_i/θ_i), the τ's and θ's are types, and $W(x_1, \ldots, x_n, y_1, \ldots, y_m)$ is a quantifier-free twff whose only free variables are $x_1, \ldots, x_n, y_1, \ldots, y_m$ and which contains only constant signs occurring in DB. The sequence of variables $x_1, \ldots, x_n$ is called the *index* of the query. For brevity, we shall usually denote typical queries by $\langle \mathbf{x}/\tau \mid (q\mathbf{y}/\boldsymbol{\theta}) W(\mathbf{x}, \mathbf{y})\rangle$. Notice that according to our definition of a twff, no type occurs in $W(\mathbf{x}, \mathbf{y})$, types are associated only with the quantifiers and index of a query.

Intuitively, a typical query denotes the set of n-tuples $\mathbf{x}$ such that $\mathbf{x} \in |\tau|$ (i.e., the x's satisfy the type restrictions given by τ) and such that $(q\mathbf{y}/\boldsymbol{\theta}) W(\mathbf{x}, \mathbf{y})$ holds.

The following are some sample queries for the database of Example 2.1:

(i) Who are a's teachers?

$\langle x/\text{TEACHER} \mid \text{TEACHER-OF}(a, x)\rangle$

(ii) Who teaches all history courses?

$\langle x/\text{TEACHER} \mid (y/\text{HISTORY})\text{TEACH}(x, y)\rangle$

(iii) Who teaches calculus but not history?

$\langle x/\text{TEACHER} \mid (Ey/\text{CALCULUS})(z/\text{HISTORY})\text{TEACH}(x, y) \wedge \sim\text{TEACH}(x, z)\rangle$

2.4 THE SEMANTICS OF QUERIES. The semantics of queries is defined relative to a database DB. It is tempting to make the following definition: Let $Q = \langle \mathbf{x}/\tau \mid (q\mathbf{y}/\boldsymbol{\theta}) W(\mathbf{x}, \mathbf{y})\rangle$. An n-tuple of constant signs $\mathbf{c} = (c_1, \ldots, c_n)$ is an answer to Q (with respect to a database DB) iff

$$\text{DB} \vdash \tau_1(c_1) \wedge \cdots \wedge \tau_n(c_n) \wedge (q\mathbf{y}/\boldsymbol{\theta}) W(\mathbf{c}, \mathbf{y}),$$

where in general $T \vdash W$ denotes that W is provable in the first-order theory T. Unfortunately, this defines an inappropriate notion of answer in the case of incompletely specified worlds.

Example 2.2. Suppose it is known of an individual John only that he is either in Boston or New York.

$$\text{LOCATION (John, Boston)} \vee \text{LOCATION (John, NY)}$$

Consider the query, "Where is John?"

$Q = \langle x/\text{CITY} \mid \text{LOCATION(John, } x)\rangle$

By the above definition of an answer, Q has no answer. Clearly, a response like, "There is one such city; it is either Boston or New York, but I don't know which," is far more desirable. Such answers arise in incompletely specified worlds. Since such ambiguities seem to be an inherent property of our knowledge about many domains, a question-answering system should be capable of coping with incomplete knowledge of this kind.

Accordingly, we propose a revised definition of answer, one that provides for the indefinite answers which arise in incompletely specified worlds.

Let $Q = \langle \mathbf{x}/\tau \mid (qy/\theta)W(\mathbf{x}, \mathbf{y})\rangle$. A set of n-tuples of constant signs $\{\mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(m)}\}$ is an *answer to* Q *(with respect to DB)* iff

$$\mathrm{DB} \vdash \bigvee_{i \leq m} \tau(\mathbf{c}^{(i)}) \wedge (qy/\theta)W(\mathbf{c}^{(i)}, \mathbf{y}),$$

where $\tau(\mathbf{c})$ denotes $\tau_1(c_1) \wedge \cdots \wedge \tau_n(c_n)$. It is easy to see that by the τ-completeness assumption on the TDB, this condition is equivalent to the following two conditions:

(1) For $i = 1, \ldots, m$, $\mathrm{TDB} \vdash \tau(\mathbf{c}^{(i)})$, i.e., $\mathbf{c}^{(i)} \in |\tau|$.
(2) $\mathrm{DB} \vdash \bigvee_{i \leq m} (qy/\theta)W(\mathbf{c}^{(i)}, \mathbf{y})$.

The next example shows that these two conditions correspond to type checking and more conventional theorem proving, respectively. The computational advantage of this decomposition is that the former task is decidable; it may be effected *independently of the theorem-proving task* and may invoke whatever data structures and specialized routines that are deemed optimal for the task. Notice that this decomposition fails in the absence of τ-completeness of the TDB.

Example 2.3. Assume a single type τ, a single common predicate P, and the simple database:

TDB: $\tau(a) \vee \tau(b)$; TDB is not τ-complete.
PDB: $P(a)$, $P(b)$.

Then $Q = \langle x/\tau \mid P(x)\rangle$ has the single answer $\{a, b\}$, but condition (1) above fails on the two components a and b of the answer $\{a, b\}$.

Instead of denoting an answer as a set of tuples $\{\mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(m)}\}$, we prefer the more suggestive notation $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)}$ and shall refer to such expressions as *disjunctive tuples*. Notice that if $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)}$ is an answer to Q and $\mathbf{c}$ is any n-tuple of constant signs satisfying the type constraint τ, then so also is $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)} + \mathbf{c}$ an answer to Q. This suggests the need for the following definitions:

(1) An answer $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)}$ to Q is *minimal* iff for no i, $1 \leq i \leq m$, is $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(i-1)} + \mathbf{c}^{(i+1)} + \cdots + \mathbf{c}^{(m)}$ an answer to Q.
(2) If $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)}$ is a minimal answer to Q, then

(i) if $m = 1$, it is a *definite answer* to Q;
(ii) if $m > 1$, it is an *indefinite answer* to Q.

A minimal indefinite answer $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)}$ to $\langle \mathbf{x}/\tau \mid (qy/\theta)W(\mathbf{x}, \mathbf{y})\rangle$ has the interpretation: $\mathbf{x}$ is either $\mathbf{c}^{(1)}$ or $\mathbf{c}^{(2)}$ or $\cdots$ or $\mathbf{c}^{(m)}$, but there is no way, given the information in the database, of determining which

We can now define the semantics of our query language. The *value of a query* Q *(wrt DB)*, $\|Q\|_{\mathrm{DB}}$, is the set of minimal answers to Q. When the DB is contextually obvious, we shall often write $\|Q\|$ instead of $\|Q\|_{\mathrm{DB}}$.

This paper is not concerned with methods for query evaluation. The interested reader is referred to [15, 16] for an approach to query evaluation which is complete in the sense that all minimal answers, including indefinite ones, will be returned.

3. *Closed Databases*

Let $\{c_1, \ldots, c_p\}$ be all of the distinct constant signs occurring in the formulas of a database DB. These are finite in number since there are just finitely many formulas in DB. Then

the following is the *domain closure axiom* for DB:

$$(x)x = c_1 \vee \cdots \vee x = c_p.$$

If DB contains its domain closure axiom, then DB is called a *closed database.*

In effect, a closed database DB restricts the universe of discourse to just those individuals denoted by the constant signs of DB. As far as DB is concerned, no other individuals exist.

As we shall see in the next section, the presence or absence of a domain closure axiom for DB is irrelevant in the case of existential queries; the set of answers for such queries remains the same whether or not this axiom is in force. This is not true for queries with universal quantifiers. To see why, consider the database of Example 2.1 and the query $\langle x/\text{TEACHER} \mid (y/\text{HISTORY})\text{TEACH}(x, y)\rangle$, i.e., "Who teaches all history courses?" Since $|\text{HISTORY}| = \{\text{H100, H200}\}$ and since C teaches both these courses, we would intuitively expect C to be an answer to this query. But in the absence of a domain closure axiom for this database, no proof of $(y)\text{HISTORY}(y) \supset \text{TEACH}(C, y)$ exists, so that C is not an answer. The reason is clear enough. Without domain closure there are models of the database containing additional individuals which are history courses but which are not taught by C. The difficulty can be traced to the semantics of "for all." If the intended interpretation of "(x)" is "for all x *that you know about,*" then the domain closure axiom is required. On the other hand, if we mean "for all x, even those whose possible existence you may not be aware of," then no such axiom is in force.

Whether or not a given database is to be treated as closed is clearly a pragmatic question. Examples of "naturally closed" databases are inventories, the education domain of Example 2.1, blocks worlds, and flight schedules. It is of some interest to note that relational database theory [6] implicitly makes a domain closure assumption by virtue of the way in which the relational calculus interprets universal quantifiers.

The purpose of the next two sections is to demonstrate how the domain closure axiom can be "factored out" of a closed database.

4. *Existential Queries and the Domain Closure Axiom*

In general, the evaluation of queries requires the use of a theorem prover. For databases with large numbers of individual constants, we can therefore expect the domain closure axiom to lead to unfeasible computations. Fortunately, as the next theorem shows, this axiom is irrelevant in the case of *existential queries*, i.e., queries of the form $\langle \mathbf{x}/\tau \mid (E\mathbf{y}/\boldsymbol{\theta})W(\mathbf{x}, \mathbf{y})\rangle$.

THEOREM 4.1. *Let DC be the domain closure axiom of a closed database DB. Then*

$$DB \vdash (E\mathbf{y}/\boldsymbol{\theta})W(\mathbf{y}) \quad \textit{iff} \quad DB - \{DC\} \vdash (E\mathbf{y}/\boldsymbol{\theta})W(\mathbf{y}),$$

where $W(\mathbf{y})$ is a quantifier-free formula with free variables y *whose only constant signs are among those occurring in DB.*

PROOF. Each formula of $D = \text{DB} \cup \{\sim(E\mathbf{y}/\boldsymbol{\theta})W(\mathbf{y})\}$ has the property that when transformed to prenex normal form, it has no existential quantifier. This means that in converting such a formula to quantifier-free form (see, e.g., [14]), no Skolem constant or function is introduced. Hence the Herbrand Universe of D, $H_G(D)$, is the set of constant signs $\{c_1, \ldots, c_p\}$ appearing in D and hence in DB. Suppose D is unsatisfiable. Then by Herbrand's theorem [4], the set S of ground instances over $H_G(D)$ of formulas of D is unsatisfiable. Suppose S contains a ground instance I of the domain closure axiom which is subsumed by a ground unit U of S. Then delete I from S. Let Σ be the set resulting from S by so deleting all such I's whenever a U subsumes I. Σ is unsatisfiable.

CLAIM. Σ *contains no ground instance of the domain closure axiom.*

Assuming the truth of this claim, it follows, again by Herbrand's theorem, that $(\text{DB} - \{\text{DC}\}) \cup \{\sim(E\mathbf{y}/\boldsymbol{\theta})W(\mathbf{y})\}$ is unsatisfiable, whence $\text{DB} - \{\text{DC}\} \vdash (E\mathbf{y}/\boldsymbol{\theta})W(\mathbf{y})$.

PROOF OF CLAIM. Every ground instance of DC is of the form $c = c_1 \vee \cdots \vee c = c_p$ where c is one of $c_1, \ldots, c_p$, so that such a ground instance will be subsumed by a ground instance of the equality axiom $(x)x = x$. Q.E.D.

5. *Reduction of Arbitrary Queries to Existential Queries for Closed Databases*

The previous section showed that the presence of the domain closure axiom in a database is irrelevant in the case of existential queries. As we saw in Section 3 above, this is not true for arbitrary queries.

It follows that there remains a problem in dealing with queries whose matrix contains one or more universal quantifiers. Our approach will be to "strip off" leading quantifiers in the matrix until a purely existential query remains. To that end, we require the following.

LEMMA 5.1. *Let $W(y)$ be a (not necessarily quantifier-free) twff with free variable y, and let DB be a closed database. Then $DB \vdash (y/\theta)W(y)$ iff for all constant signs $c \in |\theta|$, $DB \vdash W(c)$.*

PROOF. The $\Rightarrow$ half is obvious. To establish the $\Leftarrow$ half we shall argue somewhat informally. For an arbitrary individual Y, the domain closure axiom guarantees that Y is equal to one of the constant signs $c_1, \ldots, c_p$ occurring in DB. Hence either $Y \in |\theta|$ or $Y \notin |\theta|$. If $Y \in |\theta|$, then DB $\vdash W(Y)$ by hypothesis, whence DB $\vdash \theta(Y) \supset W(Y)$. If $Y \notin |\theta|$, then by the τ-completeness of the TDB, TDB $\vdash \sim\theta(Y)$. Since TDB $\subseteq$ DB, we have DB $\vdash \theta(Y) \supset W(Y)$. Thus in either case DB $\vdash \theta(Y) \supset W(Y)$ for an arbitrary individual Y, so that DB $\vdash (y)\theta(y) \supset W(y)$; i.e., DB $\vdash (y/\theta)W(y)$. Q.E.D.

Notice that the proof of Lemma 5.1 invokes the τ-completeness assumption for the TDB.

LEMMA 5.2. *Let $W(\mathbf{x}, y)$ be a (not necessarily quantifier-free) twff with free variables $\mathbf{x}$ and y. Then for a closed database, $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)}$ is an answer to $\langle \mathbf{x}/\tau \mid (y/\theta)W(\mathbf{x}, y)\rangle$ iff for all $a_1 \in |\theta|, \ldots, a_m \in |\theta|$, $(\mathbf{c}^{(1)}, a_1) + \cdots + (\mathbf{c}^{(m)}, a_m)$[1] is an answer to $\langle \mathbf{x}/\tau, y/\theta \mid W(\mathbf{x}, y)\rangle$.*

PROOF. $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)}$ is an answer to $\langle \mathbf{x}/\tau \mid (y/\theta)W(\mathbf{x}, y)\rangle$ iff $\mathbf{c}^{(i)} \in |\tau|$, $i = 1, \ldots, m$ and DB $\vdash \bigvee_{i \le m}(y/\theta)W(\mathbf{c}^{(i)}, y)$; iff $\mathbf{c}^{(i)} \in |\tau|$, $i = 1, \ldots, m$ and DB $\vdash (y_1/\theta)\cdots(y_m/\theta)\bigvee_{i \le m} W(\mathbf{c}^{(i)}, y_i)$; iff, by Lemma 5.1, $\mathbf{c}^{(i)} \in |\tau|$, $i = 1, \ldots, m$ and DB $\vdash \bigvee_{i \le m} W(\mathbf{c}^{(i)}, a_i)$ for all $a_1 \in |\theta|, \ldots, a_m \in |\theta|$; iff for all $a_1 \in |\theta|, \ldots, a_m \in |\theta|$, $(\mathbf{c}^{(1)}, a_1) + \cdots + (\mathbf{c}^{(m)}, a_m)$ is an answer to $\langle \mathbf{x}/\tau, y/\theta \mid W(\mathbf{x}, y)\rangle$. Q.E.D.

Definition. Let $Q = \langle \mathbf{x}/\tau, z/\psi \mid (qy/\theta)W(\mathbf{x}, \mathbf{y}, z)\rangle$. The *quotient of* $\| Q \|$ *by* ψ, $\Delta_\psi \| Q \|$, is a set of disjunctive tuples defined as follows: $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)} \in \Delta_\psi \| Q \|$ iff

(1) for all $a_1 \in |\psi|, \ldots, a_m \in |\psi|$, $(\mathbf{c}^{(1)}, a_1) + \cdots + (\mathbf{c}^{(m)}, a_m)$ is an answer (not necessarily minimal) to Q (and hence some subdisjunctive tuple of $(\mathbf{c}^{(1)}, a_1) + \cdots + (\mathbf{c}^{(m)}, a_m)$ is an element of $\| Q \|$), and
(2) for no i, $1 \le i \le m$, does $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(i-1)} + \mathbf{c}^{(i+1)} + \cdots + \mathbf{c}^{(m)}$ have property (1).

The operator Δ_ψ is called the *division operator with respect to* ψ and is an appropriate generalization of the division operator of relational algebra [5].

Evidently the calculation of $\Delta_\psi \| Q \|$ when given $\| Q \|$ is in general a nontrivial undertaking. Since the specification of a division algorithm would take us too far afield from the purposes of this paper, we omit any details. Notice, however, that in the event that $\| Q \|$ consists only of definite answers, the calculation of $\Delta_\psi \| Q \|$ is straightforward; $\Delta_\psi \| Q \|$ contains only definite answers and $\mathbf{c} \in \Delta_\psi \| Q \|$ iff for all $a \in |\psi|$, $(\mathbf{c}, a) \in \| Q \|$.

Example 5.1. Suppose $\| Q \| = \{(A, a) + (B, b), (A, b), (B, a), (C, a), (C, b), (D, a), (E, a) + (F, a), (E, b)\}$ and $|\psi| = \{a, b\}$. Then $\Delta_\psi \| Q \| = \{A + B, C\}$.

[1] There is a slight abuse of notation here. If $\mathbf{c} = (c_1, \ldots, c_n)$, then $(\mathbf{c}, a)$ is intended to denote $(c_1, \ldots, c_n, a)$.

THEOREM 5.3. *For a closed database,*

$$\| \langle \mathbf{x}/\tau | (y/\theta) W(\mathbf{x}, y) \rangle \| = \Delta_\theta \| \langle \mathbf{x}/\tau, y/\theta | W(\mathbf{x}, y) \rangle \|,$$

where $W(\mathbf{x}, y)$ is a (not necessarily quantifier-free) twff with free variables $\mathbf{x}$ *and* y.

PROOF. Suppose $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)} \in \| \langle \mathbf{x}/\tau | (y/\theta) W(\mathbf{x}, y) \rangle \|$. Then $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)}$ is a minimal answer, and $\mathrm{DB} \vdash \mathsf{V}_{i \le m}(y/\theta) W(\mathbf{c}^{(i)}, y)$. By Lemma 5.2, for all $a_1 \in |\theta|, \ldots, a_m \in |\theta|$, $(\mathbf{c}^{(1)}, a_1) + \cdots + (\mathbf{c}^{(m)}, a_m)$ is an answer to $\langle \mathbf{x}/\tau, y/\theta | W(\mathbf{x}, y) \rangle$. We must prove that for no i, $1 \le i \le m$, does $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(i-1)} + \mathbf{c}^{(i+1)} + \cdots + \mathbf{c}^{(m)}$ have this property.

Suppose, on the contrary, that for all $a_1 \in |\theta|, \ldots, a_{m-1} \in |\theta|$, $(\mathbf{c}^{(1)}, a_1) + \cdots + (\mathbf{c}^{(m-1)}, a_{m-1})$ is an answer to $\langle \mathbf{x}/\tau, y/\theta | W(\mathbf{x}, y) \rangle$. Again, by Lemma 5.2, it must be that $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m-1)}$ is an answer to $\langle \mathbf{x}/\tau | (y/\theta) W(\mathbf{x}, y) \rangle$, contradicting the minimality of $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)}$. Now suppose that

$$\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)} \in \Delta_\theta \| \langle \mathbf{x}/\tau, y/\theta | W(\mathbf{x}, y) \rangle \|. \tag{5.1}$$

Then for all $a_1 \in |\theta|, \ldots, a_m \in |\theta|$, $(\mathbf{c}^{(1)}, a_1) + \cdots + (\mathbf{c}^{(m)}, a_m)$ is an answer to $\langle \mathbf{x}/\tau, y/\theta | W(\mathbf{x}, y) \rangle$, whence by Lemma 5.2, $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)}$ is an answer to $\langle \mathbf{x}/\tau | (y/\theta) W(\mathbf{x}, y) \rangle$.

We must prove that $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)}$ is a minimal answer to $\langle \mathbf{x}/\tau | (y/\theta) W(\mathbf{x}, y) \rangle$. To that end, assume it is not—say $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m-1)}$ is an answer. Then by Lemma 5.2, for all $a_1 \in |\theta|, \ldots, a_{m-1} \in |\theta|$, $(\mathbf{c}^{(1)}, a_1) + \cdots + (\mathbf{c}^{(m-1)}, a_{m-1})$ is an answer to $\langle \mathbf{x}/\tau, y/\theta | W(\mathbf{x}, y) \rangle$, which contradicts (5.1) and the definition of Δ_θ. Q.E.D.

Recall that Theorem 4.1 allows us to dispense with the domain closure axiom for existential queries. Our objective is to reduce all queries to this case. Theorem 5.3 provides a mechanism for "stripping off" leading universal quantifiers. Thus

$$\| \langle \mathbf{x}/\tau | (\mathbf{y}/\psi)(E\mathbf{z}/\theta) W(\mathbf{x}, \mathbf{y}, \mathbf{z}) \rangle \| = \Delta_{\psi_1} \cdots \Delta_{\psi_m} \| \langle \mathbf{x}/\tau, \mathbf{y}/\psi | (E\mathbf{z}/\theta) W(\mathbf{x}, \mathbf{y}, \mathbf{z}) \rangle \|,$$

which reduces the universal query to a sequence of division operators applied to an existential query. We cannot, as yet, reduce a query like $\langle x/\tau_1 | (Ey/\tau_2)(z/\tau_3) W(x, y, z) \rangle$ to an existential query. Some mechanism for "stripping off" existential quantifiers must be provided.

LEMMA 5.4. *Let $W(y)$ be a (not necessarily quantifier-free) twff with free variable y, and let DB be a database. Then $\mathrm{DB} \vdash (Ey/\theta) W(y)$ iff there are constants $a_1, \ldots, a_r \in |\theta|$ such that $\mathrm{DB} \vdash W(a_1) \vee \cdots \vee W(a_r)$.*

PROOF. The $\Leftarrow$ half is immediate. Hence, suppose $\mathrm{DB} \vdash (Ey/\theta) W(y)$. Then there is a minimal set of constants $\{a_1, \ldots, a_r\}$ such that $\mathrm{DB} \vdash \theta(a_1) W(a_1) \vee \cdots \vee \theta(a_r) W(a_r)$. If $\mathrm{TDB} \vdash \theta(a_i)$, $i = 1, \ldots, r$, the result follows. Otherwise, by the τ-completeness of the TDB, for some a_i, say a_1, we have $\mathrm{TDB} \vdash \sim\theta(a_1)$. But then $\mathrm{DB} \vdash \theta(a_2) W(a_2) \vee \cdots \vee \theta(a_r) W(a_r)$, contradicting the minimality of the set $\{a_1, \ldots, a_r\}$. Q.E.D.

Notice that the proof of Lemma 5.4 requires the τ-completeness assumption for the TDB.

LEMMA 5.5. *Let $W(\mathbf{x}, y)$ be a (not necessarily quantifier-free) twff with free variables* $\mathbf{x}$ *and* y. *Then $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)}$ is an answer to $\langle \mathbf{x}/\tau | (Ey/\theta) W(\mathbf{x}, y) \rangle$ iff there exist constants $a_1, \ldots, a_r \in |\theta|$ such that $+_{j \le r, i \le m} (\mathbf{c}^{(i)}, a_j)$ is an answer to $\langle \mathbf{x}/\tau, y/\theta | W(\mathbf{x}, y) \rangle$.*

PROOF. $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)}$ is an answer to $\langle \mathbf{x}/\tau | (Ey/\theta) W(\mathbf{x}, y) \rangle$ iff $\mathbf{c}^{(i)} \in |\tau|$, $i = 1, \ldots, m$, and $\mathrm{DB} \vdash \mathsf{V}_{i \le m} (Ey/\theta) W(\mathbf{c}^{(i)}, y)$; iff $\mathbf{c}^{(i)} \in |\tau|$, $i = 1, \ldots, m$, and $\mathrm{DB} \vdash (Ey/\theta) \mathsf{V}_{i \le m} W(\mathbf{c}^{(i)}, y)$; iff $\mathbf{c}^{(i)} \in |\tau|$, $i = 1, \ldots, m$, and, by Lemma 5.4, there exist constants $a_1, \ldots, a_r \in |\theta|$ such that $\mathrm{DB} \vdash \mathsf{V}_{j \le r} \mathsf{V}_{i \le m} W(\mathbf{c}^{(i)}, a_j)$; iff there exist constants $a_1, \ldots, a_r \in |\theta|$ such that $+_{j \le r, i \le m} (\mathbf{c}^{(i)}, a_j)$ is an answer to $\langle \mathbf{x}/\tau, y/\theta | W(\mathbf{x}, y) \rangle$. Q.E.D.

Definitions. Let $Q = \langle \mathbf{x}/\tau, z/\psi \mid (qy/\theta)W(\mathbf{x}, y, z)\rangle$. The *projection of* $\|Q\|$ *with respect to* ψ, $\pi_\psi\|Q\|$, is a set of disjunctive tuples defined as follows: $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)} \in \pi_\psi\|Q\|$ iff

(1) there exist constants $a_1, \ldots, a_r \in |\theta|$ such that $+_{j\leq r, i\leq m}(\mathbf{c}^{(i)}, a_j)$ is an answer (not necessarily minimal) to Q, and
(2) for no i, $1 \leq i \leq m$, does $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(i-1)} + \mathbf{c}^{(i+1)} + \cdots + \mathbf{c}^{(m)}$ have property (1).

A disjunctive tuple D_1 *subsumes* disjunctive tuple D_2 iff when viewed as sets, $D_1 \subseteq D_2$. (Recall that $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)}$ is our preferred notation for $\{\mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(m)}\}$.) For example, (a, b) subsumes $(a, b) + (c, d)$, which subsumes $(a, a) + (a, b) + (b, b) + (c, d)$.

Computing $\pi_\psi\|Q\|$ given $\|Q\|$ is straightforward:

(a) Let the set A be obtained from $\|Q\|$ as follows: For each disjunctive tuple $\mathbf{d}^{(1)} + \cdots + \mathbf{d}^{(m)} \in \|Q\|$, form $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)}$ where $\mathbf{c}^{(i)}$ is obtained from $\mathbf{d}^{(i)}$ by deleting the last component of $\mathbf{d}^{(i)}$. (Recall that $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)}$ is an alternate representation for $\{\mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(m)}\}$, so repeated tuples are to be deleted from $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)}$.) A is the set of such resulting disjunctive tuples.

(b) Delete from A any disjunctive tuple subsumed by some other disjunctive tuple of A. The resulting set is $\pi_\psi\|Q\|$.

The operator π_ψ is called the *projection operator with respect to* ψ and is an appropriate generalization of the projection operator of relational algebra [5].

Notice that in the event that $\|Q\|$ consists only of definite answers, the calculation of $\pi_\psi\|Q\|$ is straightforward: $\pi_\psi\|Q\|$ contains only definite answers, and $\mathbf{c} \in \pi_\psi\|Q\|$ iff for some $a \in |\psi|$, $(\mathbf{c}, a) \in \|Q\|$. Section 6 describes some conditions on databases under which only definite answers can arise.

Example 5.2. Suppose $Q = \langle x_1/\theta_1, x_2/\theta_2, z/\psi \mid W(x, y, z)\rangle$, and that

$$\|Q\| = \{(a, a, A) + (a, a, B) + (a, d, C), (a, b, A) + (a, c, D), (a, b, C)\}.$$

Then

$$\pi_\psi\|Q\| = \{(a, a) + (a, d), (a, b)\}.$$

THEOREM 5.6. $\|\langle \mathbf{x}/\tau \mid (Ey/\theta)W(\mathbf{x}, y)\rangle\| = \pi_\theta\|\langle \mathbf{x}/\tau, y/\theta \mid W(\mathbf{x}, y)\rangle\|$, *where* $W(\mathbf{x}, y)$ *is a (not necessarily quantifier-free) twff with free variables* $\mathbf{x}$ *and* y.

PROOF. $\Rightarrow$. Suppose $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)} \in \|\langle \mathbf{x}/\tau \mid (Ey/\theta)W(\mathbf{x}, y)\rangle\|$. Then by Lemma 5.5, there exist constants $a_1, \ldots, a_r \in |\theta|$ such that $+_{j\leq r, i\leq m}(\mathbf{c}^{(i)}, a_j)$ is an answer to $Q = \langle \mathbf{x}/\tau, y/\theta \mid W(\mathbf{x}, y)\rangle$. Hence, $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)} \in \pi_\theta\|\langle \mathbf{x}/\tau, y/\theta \mid W(\mathbf{x}, y)\rangle\|$ provided we can prove that for no $b_1, \ldots, b_k \in |\theta|$ is $+_{j\leq k, i\leq m-1}(\mathbf{c}^{(i)}, b_j)$ an answer to Q. Suppose the contrary. Then by Lemma 5.5, $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m-1)}$ is an answer to $\langle \mathbf{x}/\tau \mid (Ey/\theta)W(\mathbf{x}, y)\rangle$, contradicting the fact that $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)}$ is a minimal such answer.

$\Leftarrow$. Suppose

$$\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)} \in \pi_\theta\|\langle \mathbf{x}/\tau, y/\theta \mid W(\mathbf{x}, y)\rangle\|. \tag{5.2}$$

Then there exist constants $a_1, \ldots, a_r \in |\theta|$ such that $+_{j\leq r, i\leq m}(\mathbf{c}^{(i)}, a_j)$ is an answer to $Q = \langle \mathbf{x}/\tau, y/\theta \mid W(\mathbf{x}, y)\rangle$, so by Lemma 5.5, $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)}$ is an answer to $\langle \mathbf{x}/\tau \mid (Ey/\theta)W(\mathbf{x}, y)\rangle$. We prove it is a minimal such answer, in which case the result follows. For if not, then $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m-1)}$, say, is an answer to $\langle \mathbf{x}/\tau \mid (Ey/\theta)W(\mathbf{x}, y)\rangle$, whence by Lemma 5.5, there exist constants $b_1, \ldots, b_k \in |\theta|$ such that $+_{j\leq k, i\leq m-1}(\mathbf{c}^{(i)}, b_j)$ is an answer to Q, contradicting (5.2). Q.E.D.

For closed databases Theorems 5.3 and 5.6 allow us to reduce an arbitrary query to an existential one in the sense that the set of minimal answers to an arbitrary query can be obtained by applying the projection and division operators to the minimal answers of an appropriate existential query. Thus, for example,

$$\|\langle x/\tau \mid (Ey/\theta)(z/\psi)(Ew/\rho)W(x, y, z, w)\rangle\| = \pi_\theta\Delta_\psi\|\langle x/\tau, y/\theta, z/\psi \mid (Ew/\rho)W(x, y, z, w)\rangle\|.$$

Since the domain closure axiom is irrelevant to existential query evaluation (Section 4), we have thus shown that arbitrary queries may be evaluated independently of this axiom, i.e., that the domain closure axiom may be removed from the database. In essence, the effects of this axiom are completely embodied in the division operator.

Alternatively, we can view this result as an appropriate generalization of conventional relational database theory [6] to relational databases with a deductive component. Non-deductive relational databases *implicitly* appeal to a domain closure axiom by virtue of the relational algebra division operator which completely characterizes the effects of this axiom. What we have shown is that this view generalizes to the deductive case provided the relational algebra operators of projection and division are suitably generalized.

6. *On Indefinite Answers*

As we remarked in Section 2.4, indefinite answers arise in incompletely specified worlds. It is just such answers which lead to the computational and conceptual complexities associated with the projection and division operators of the previous section. In this section we provide a sufficient condition that all minimal answers be definite, a condition which encompasses a large and natural class of databases and queries. For this purpose it is assumed that the reader is familiar with resolution theory [4].

Let T be a database twff in prenex normal form, so that T has the form $(\mathbf{x}/\tau)C(\mathbf{x})$ for some quantifier-free formula $C(\mathbf{x})$. (Recall that all twffs of a database have no existential quantifiers in their prenex normal forms.) Without loss of generality, we can assume that $C(\mathbf{x})$ is a *clause*, i.e., a disjunction of literals. Then T is a *Horn twff* iff $C(\mathbf{x})$ is a *Horn clause*, i.e., $C(\mathbf{x})$ has at most one positive literal. A database is *Horn* iff each of its twffs is Horn. The database of Example 2.1 is Horn.

An existential query is *positive* iff it has the form $\langle \mathbf{x}/\tau \mid (E\mathbf{y}/\boldsymbol{\theta})(K_1 \vee \cdots \vee K_n)\rangle$, where each K is a conjunct of positive literals.

THEOREM 6.1. *Suppose that DB is a Horn satisfiable database, and that Q is a positive existential query. Then $\| Q \|$ consists of definite answers only.*

PROOF. Suppose, on the contrary, that $Q = \langle \mathbf{x}/\tau \mid (E\mathbf{y}/\boldsymbol{\theta})(K_1(\mathbf{x}, \mathbf{y}) \vee \cdots \vee K_n(\mathbf{x}, \mathbf{y}))\rangle$ has an indefinite answer $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)}$, $m > 1$. Then

$$\mathrm{DB} \vdash \bigvee_{i \le m} (E\mathbf{y}/\boldsymbol{\theta})(K_1(\mathbf{c}^{(i)}, \mathbf{y}) \vee \cdots \vee K_n(\mathbf{c}^{(i)}, \mathbf{y})),$$

i.e.,

$$\mathrm{DB} \vdash (E\mathbf{y}/\boldsymbol{\theta}) \left[\bigvee_{\substack{i \le m \\ j \le n}} K_j(\mathbf{c}^{(i)}, \mathbf{y}) \right],$$

so by Lemma 5.4, there exist $\mathbf{a}^{(1)}, \ldots, \mathbf{a}^{(r)} \in |\boldsymbol{\theta}|$ such that

$$\mathrm{DB} \vdash \bigvee_{\substack{i \le m \\ j \le n \\ k \le r}} K_j(\mathbf{c}^{(i)}, \mathbf{a}^{(k)}).$$

Hence

$$D = \mathrm{DB} \cup \bigcup_{\substack{i \le m \\ j \le n \\ k \le r}} \{\bar{K}_j(\mathbf{c}^{(i)}, \mathbf{a}^{(k)})\}$$

is unsatisfiable, where $\bar{K}_j$ denotes that clause obtained by negating the conjunct K_j. Notice that each literal of $\bar{K}_j$ is negative, so that $\bar{K}_j$ is Horn.

As in the proof of Theorem 4.1, the Herbrand Universe of D is the set of constant signs $\{c_1, \ldots, c_p\}$ of DB. Since D is unsatisfiable, so is

$$\mathrm{DB}_G \cup \bigcup_{\substack{i \le m \\ j \le n \\ k \le r}} \{\bar{K}_j(\mathbf{c}^{(i)}, \mathbf{a}^{(k)})\}, \tag{6.1}$$

where DB_G is the set of all ground instances of DB over the constants $\{c_1, \ldots, c_p\}$.

Now if $(\mathbf{x}/\tau)H(\mathbf{x})$ is a typical twff of DB, then it has, as typical ground instance, the formula

$$\tau_1(a_1) \wedge \cdots \wedge \tau_n(a_n) \supset H(\mathbf{a}) \tag{6.2}$$

for constants $a_1, \ldots, a_n \in \{c_1, \ldots, c_p\}$. Since TDB is τ-complete and consistent, then either $DB_G \vdash \tau_i(c_i)$ for $i = 1, \ldots, n$, or for some i, $DB_G \vdash \sim\tau_i(c_i)$. In the former case, $DB_G \vdash H(\mathbf{a})$ and we can replace formula (6.2) in DB_G by $H(\mathbf{a})$ without affecting the unsatisfiability of (6.1). In the latter case, formula (6.2) is irrelevant to the unsatisfiability of (6.1).

It follows then that

$$\text{Horn}(DB_G) \cup \bigcup_{\substack{i \leq m \\ j \leq n \\ k \leq r}} \{\bar{K}_j(\mathbf{c}^{(i)}, \mathbf{a}^{(k)})\} \tag{6.3}$$

is unsatisfiable, where

$$\text{Horn}(DB_G) = \{H(\mathbf{a}) \mid \tau_1(a_1) \wedge \cdots \wedge \tau_n(a_n) \supset H(\mathbf{a}) \in DB_G \text{ and for } i = 1, \ldots, n, DB_G \vdash \tau_i(a_i)\}.$$

Since DB is a Horn database, each clause of $\text{Horn}(DB_G)$ is a Horn clause, so that each clause of the unsatisfiable set (6.3) is a Horn clause.

Now a theorem of Henschen and Wos [8] assures us that any unsatisfiable set of Horn clauses has a positive unit refutation, i.e., a refutation by binary resolution in which one parent of each resolution operation is a positive unit. Since (6.3) is such a set of Horn clauses, it has a positive unit refutation. Since DB is satisfiable then for some i, j, k, $\bar{K}_j(\mathbf{c}^{(i)}, \mathbf{a}^{(k)})$ enters into this refutation. Moreover, from the positive unit property, no other negative clause can enter into this refutation; i.e.,

$\text{Horn}(DB_G) \cup \{\bar{K}_j(\mathbf{c}^{(i)}, \mathbf{a}^{(k)}\}$ is unsatisfiable; i.e.,
$DB \cup \{\bar{K}_j(\mathbf{c}^{(i)}, \mathbf{a}^{(k)})\}$ is unsatisfiable; i.e.,
$DB \vdash K_j(\mathbf{c}^{(i)}, \mathbf{a}^{(k)})$; i.e.,
$DB \vdash (Ey/\theta)K_j(\mathbf{c}^{(i)}, \mathbf{y})$; i.e.,
$\mathbf{c}^{(i)}$ is an answer to Q contradicting the assumed indefinite answer $\mathbf{c}^{(1)} + \cdots + \mathbf{c}^{(m)}$. Q.E.D.

The importance of this result stems from the fact that databases frequently have as their natural representation a set of Horn clauses [11].

Another class of databases for which only definite answers can arise is the class of so-called closed-world databases [17]. Such databases are characterized by an inference rule which sanctions the derivation of a ground negative literal $\sim P(c_1, \ldots, c_n)$ by failure to derive $P(c_1, \ldots, c_n)$. An important feature of such databases is that only positive information about a domain need be explicitly represented; negative information is inferred by failure to derive the corresponding positive information. For many domains, the number of negative facts can be astronomically large (e.g., an inventory and the set of pairs of parts such that neither is a subpart of the other). In that case their natural representation will be as closed-world databases; the results in [17] will then guarantee definite answers to all queries.

This section has thus shown that under two common and natural circumstances—Horn databases with positive queries, and closed-world databases—only definite answers can arise. When this is the case, the projection and division operators of Section 5 assume particularly simple forms; in fact they essentially reduce to their counterparts in relational algebra.

7. *Equality and E-Saturated Databases*

In previous sections we have shown how the domain closure axiom of a database may be dispensed with. However, the theorem prover must still cope with the equality axioms E1–

E4 of Section 2.2. Despite the fact that a firm theoretical foundation exists for the treatment of equality [18], no feasible computational approach is known. In this section we define a natural and commonly occurring condition on databases which will allow us to dispense with the genuinely troublesome axioms E2–E4 in the case of existential queries.

A database DB is *E-saturated* iff for all distinct constant signs c_i, c_j, $c_i \neq c_j \in$ DB. Thus, as far as DB is concerned, two constants are treated as equal iff they are identical syntactic objects; no individual has an alias. Many databases have this character: inventories, personnel files, flight schedules, etc. The next theorem shows that under this condition the equality axioms E2–E4 are irrelevant to the evaluation of existential queries.

THEOREM 7.1. *If DB is an E-saturated database, then*

$$DB \vdash (E\mathbf{y}/\boldsymbol{\theta})W(\mathbf{y}) \quad \textit{iff} \quad DB - \{E2, E3, E4\} \vdash (E\mathbf{y}/\boldsymbol{\theta})W(\mathbf{y}),$$

where $W(\mathbf{y})$ is a quantifier-free formula with free variables y *whose only constant signs are among those occurring in DB.*

PROOF. The proof has the same character as for Theorem 4.1. Let S, as in that proof, be the unsatisfiable set of ground instances over $H_G(D)$ of formulas of D. Let S' be obtained from S by deleting from S

(i) all tautologies;
(ii) all formulas subsumed by some other formula of S

Then S' is unsatisfiable.

CLAIM. *S' contains no ground instances of the equality axioms E2–E4.*

Assuming the truth of this claim, it follows that (DB − {E2, E3, E4}) ∪ {~$(E\mathbf{y}/\boldsymbol{\theta})W(\mathbf{y})$ is unsatisfiable, whence DB − {E2, E3, E4} ⊢ $(E\mathbf{y}/\boldsymbol{\theta})W(\mathbf{y})$.

PROOF OF CLAIM. We shall show that S' contains no ground instance of equality axiom E4. The proof for E2 and E3 is similar. Hence, suppose S' contains

$$\alpha_1 = \beta_1 \wedge \cdots \wedge \alpha_n = \beta_n \wedge P(\alpha_1, \ldots, \alpha_n) \supset P(\beta_1, \ldots, \beta_n), \tag{7.1}$$

where the α's and β's are constant signs. If α_i is the same constant sign as β_i for $i = 1, \ldots, n$, then (7.1) is a tautology and cannot be a member of S'. Hence at least one pair α_i, β_i are distinct constant signs. But then, since DB is E-saturated, $\alpha_i \neq \beta_i \in$ DB and hence $\alpha_i \neq \beta_i \in S$. Since $\alpha_i \neq \beta_i$ subsumes (the clausal form of) (7.1), S' cannot contain (7.1). Q.E.D.

Notice that the proof of Theorem 4.1 requires only that equality axiom E1 be present in the database. This observation yields

THEOREM 7.2. *If DB is a closed E-saturated database, then neither the domain closure axiom nor equality axioms E2–E4 are required for the evaluation of existential queries.*

For closed E-saturated databases, queries with universal quantifiers may be evaluated by invoking the projection and division operators, as in Section 5.

8. *On Databases with Infinitely Many Constant Signs*

Recall that we have been dealing with databases which have just finitely many constants. It is tempting to pursue what appears to be a mild generalization of our notion of database by admitting countably infinitely many constants $c_1, c_2, \ldots$. This would, for example, admit databases for which arbitrary integers would be included among the individuals. The purpose of this section is to show that this natural generalization leads to certain pathological situations. Specifically, we show for closed E-saturated databases with countably infinitely many constants that infinitely long indefinite answers can arise.

The following example, due to Andrew Adler, is a closed E-saturated database DB_∞

with countably infinitely many constants $c_0, c_1, \ldots$, a single simple type τ, and a single equivalence relation R.

An axiom specifying τ to be the universal type:

$(x)\tau(x)$

Equality axioms E1–E4 of Section 2.2.

Axioms specifying that all constants are pairwise distinct:

$c_i \neq c_j$ for $i \neq j$

Axioms defining R to be an equivalence relation. In addition, the following axiom:

$(x/\tau)(y/\tau)R(x, y) \supset x = c_0 \vee y = c_0 \vee x = y$

An infinite domain closure axiom:

$(x)x = c_0 \vee x = c_1 \vee \cdots$

The result is a theory in an infinitary logic [10]. We shall argue model-theoretically that this theory admits infinitely long indefinite answers.

If M is a model of DB_∞, then in M,

(i) R's equivalence classes are all singletons $\{c_0\}, \{c_1\}, \ldots$, or
(ii) R's equivalence classes are all singletons except for the single doublet $\{c_0, c_i\}$ for some $i \neq 0$.

Conversely, any equivalence relation over $\{c_0, c_1, \ldots\}$ with property (i) or (ii) defines a model for DB_∞. Hence there are countably many distinct models $M_0, M_1, \ldots$ for DB where R in M_0 has equivalence classes which are all singletons; and for $i \geq 1$, R in M_i has equivalence classes which are all singletons except for the doublet $\{c_0, c_i\}$. Now it is easy to see that

$$\mathrm{DB}_\infty \models \bigvee_i (y/\tau)R(y, c_0) \supset y = c_0 \vee y = c_i,$$

where, in general, $T \models W$ denotes that formula W is true in all models of theory T. Hence

$$\mathrm{DB}_\infty \models (Ex/\tau)(y/\tau)R(y, c_0) \supset y = c_0 \vee y = x.$$

This means that

$$c_0 + c_1 + \cdots \in \| \langle x/\tau \mid (y/\tau)R(y, c_0) \supset y = c_0 \vee y = x \rangle \|;$$

i.e., infinitely long indefinite answers arise.

9. *Discussion*

We have, in this paper, adopted a restricted notion of a database; no function signs are permitted so that, in particular, no existentially quantified formulas may occur. There is a sense in which this function-free assumption together with domain closure are both necessary for a computationally feasible treatment of the equality relation. For if we admit arbitrary functions, then any first-order theory can be viewed as a database, in which case the usual problems with equality can arise. On the other hand, without domain closure, these same problems can arise if we admit queries with arbitrary quantifier structures, since then universal quantifiers in a query lead to the introduction of Skolem functions in the query evaluation phase.

Of course, certain functions can be admitted, essentially by treating them relationally. Thus if f is an n-ary function it can be represented by an $(n + 1)$-ary relation F such that $F(\mathbf{x}, y)$ holds iff $f(\mathbf{x}) = y$. What is precluded by this approach to functions is the ability to treat intensional entities, i.e., descriptions of new entities in terms of old. Typically, in a first-order setting, such descriptions are formed by means of functions (usually Skolem

functions) whose extensions are not completely known. For example, in a payroll application, the exact salaries of managers may be confidential. Nevertheless, for some manager Smith, we can form the description salary (Smith), a perfectly respectable, though unknown entity. Now suppose the database is permitted to know that all managers earn more than $20,000. If we treat the function salary relationally, we obtain the representation

$$(x/\text{MANAGER})(y/\$)\text{SALARY}(x, y) \supset y > 20\text{K}$$

But since the database knows of no instance of the SALARY relation for managers, the value of the query "Which managers earn more than 20K?" is

$$\| \langle x/\text{MANAGER} \,|\, (Ey/\$)\text{SALARY}(x, y) \wedge y > 20\text{K} \rangle \| = \phi$$

rather than the intuitively correct set of all managers. The difficulty can clearly be traced to the fact that salary is a partial function.

We conclude that whenever an application requires only total functions, those functions may be represented relationally, in which case the results of this paper continue to apply. The function-free assumption on databases is therefore not as restrictive as it might first have appeared.

REFERENCES

1 BISHOP, C., AND REITER, R. On taxonomies. Tech. Rep., Dept. of Computer Science, U. of British Columbia, Vancouver, British Columbia, 1980. To appear.

2 BLEDSOE, W. W. Splitting and reduction heuristics in automatic theorem proving. *Artif. Intell. 2*, 1 (Spring 1971), 55–77.

3 CHANG, C. L. DEDUCE 2: further investigations on deduction in relational data bases. In *Logic and Data Bases*, H. Gallaire and J. Minker, Eds., Plenum Press, New York, 1978, pp. 201–236.

4 CHANG, C. L., AND LEE, R. C. T. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, New York, 1973.

5 CODD, E. F. Relational completeness of data base sublanguages. In *Data Base Systems*, R. Rustin, Ed., Prentice-Hall, Englewood Cliffs, N. J., 1972, pp. 65–98.

6 DATE, C. J. *An Introduction to Data Base Systems*. Addison-Wesley, Reading, Mass., 1975.

7. GALLAIRE, H., AND MINKER, J. *Logic and Data Bases*. Plenum Press, New York, 1978.

8 HENSCHEN, L. AND WOS, L. Unit refutations and Horn sets. *J. ACM 21*, 4 (Oct. 1974), 590–605.

9 HILBERT, D., AND ACKERMANN, W. *Principles of Mathematical Logic*. Chelsea, New York, 1950.

10 KEISLER, H. J. *Model Theory for Infinitary Logic*. North-Holland, Amsterdam, 1971.

11 KOWALSKI, R. *Logic for Problem Solving*. North-Holland Elsevier, New York, 1979.

12 MCSKIMIN, J. R. The use of semantic information in deductive question-answering systems. Ph. D. Thesis, Dept. of Computer Science, U. of Maryland, College Park, Md., 1976.

13 MCSKIMIN, J. R., AND MINKER, J. The use of a semantic network in a deductive question-answering system. Tech. Rep. TR-506, Dept. of Computer Science, U. of Maryland, 1977.

14 NILSSON, N. J. *Problem Solving Methods in Artificial Intelligence*. McGraw-Hill, New York, 1971.

15 REITER, R. *An Approach to Deductive Question-Answering*. BBN Tech. Rep. 3649, Bolt, Beranek and Newman, Inc., Cambridge, Mass., Sept. 1977, 161 pp.

16 REITER, R. Deductive question-answering on relational data bases. In *Logic and Data Bases*, H. Gallaire and J. Minker, Eds., Plenum Press, New York, 1978, pp. 149–177.

17 REITER, R. On closed world data bases. In *Logic and Data Bases*, H. Gallaire and J. Minker, Eds., Plenum Press, New York, 1978, pp. 55–76.

18. ROBINSON, G. A., AND WOS, L. Paramodulation and theorem proving in first order theories with equality. In *Machine Intelligence, Vol. 4*, B. Meltzer and D. Michie, Eds., American Elsevier, New York, 1969, pp. 135–150.

RECEIVED JANUARY 1979, REVISED APRIL 1979, ACCEPTED APRIL 1979