ON CLOSED WORLD DATA BASES

by

RAYMOND REITER

TECHNICAL REPORT 77-16

1977 OCTOBER

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

ABSTRACT

Deductive question-answering system generally evaluate queries under one of two possible assumptions which we in this paper refer to as the open and closed world assumptions. The open world assumption corresponds to the usual first order approach to query evaluation: Given a data base DB and a query Q, the only answers to Q are those which obtain from proofs of Q given DB as hypotheses. Under the closed world assumption, certain answers are admitted as a result of failure to find a proof. More specifically, if no proof of a positive ground literal exists, then the negation of that literal is assumed true.

In this paper, we show that closed world evaluation of an arbitrary query may be reduced to open world evaluation of so-called atomic queries. We then show that the closed world assumption can lead to inconsistencies, but for Horn data bases no such inconsistencies can arise.

Presented at the Workshop on Logic and Data Bases, Toulouse, France, November 16-18, 1977.

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

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

ON CLOSED WORLD DATA BASES

by

Raymond Reiter
Department of Computer Science
University of British Columbia
Vancouver, B.C. V6T 1W5
Canada

1. INTRODUCTION

Deductive question-answering systems generally evaluate queries under one of two possible assumptions which we in this paper refer to as the open and closed world assumptions. The open world assumption corresponds to the usual first order approach to query evaluation: Given a data base DB and a query Q, the only answers to Q are those which obtain from proofs of Q given DB as hypotheses. Under the closed world assumption, certain answers are admitted as a result of failure to find a proof. More specifically, if no proof of a positive ground literal exists, then the negation of that literal is assumed true. This can be viewed as equivalent to implicitly augmenting the given data base with all such negated literals.

For many domains of application, closed world query evaluation is appropriate since, in such domains, it is natural to explicitly represent only positive knowledge and to assume the truth of negative facts by default. For example, in an airline data base, all flights and the cities which they connect will be explicitly represented. If I fail to find an entry indicating that Air Canada flight 103 connects Vancouver with Toulouse I will conclude that it does not.

This paper is concerned with closed world query evaluation and its relationship

to open world evaluation. In Section 2 we define a query language and the notion of an open world answer to a query. Section 3 formally defines the closed world assumption and the notion of a closed world answer. Section 4 shows how closed world query evaluation may be decomposed into open world evaluation of so-called "atomic queries" in conjuction with the set operations of intersection, union and difference, and the relational algebra operation of projection. In Section 5 we show that the closed world assumption can lead to inconsistencies. We prove, moreover, that for Horn data bases no such inconsistencies can arise. Also, for Horn data bases, the occurrence of purely negative clauses is irrelevant to closed world query evaluation. By removing such negative clauses one is left with so-called definite data bases which are then consistent under both the open and closed world assumptions.

In the interest of brevity, we have ommitted all proofs of the results in this paper. The dedicated reader is referred to [Reiter 1977a] for the details.

2. DATA BASES AND QUERIES

The query language of this paper is set oriented i.e. we seek all objects (or tuples of objects) having a given property. For example, in an airline data base the request "Give all flights and their carriers which fly from Boston to England" might be represented in our query language by:

<x/Flight, y/Airline|(Ez/City)Connect x,Boston,z ∧ Owns y,x ∧ City-of z,England>

which denotes the set of all ordered pairs (x,y) such that x is a flight, y is an airline and

(Ez/City)Connect x,Boston,z ∧ Owns y,x ∧ City-of z,England

is true. The syntactic objects Flight, Airline and City are called types and serve to restrict the variables associated with them to range over objects of that type. Thus, (Ez/City) may be read as "There is a z which is a city".

Formally, all queries have the form

$<x_1/\tau_1,\ldots,x_n/\tau_n|(Ey_1/\theta_1)\ldots(Ey_m/\theta_m)W(x_1,\ldots,x_n,y_1,\ldots,y_m)>$

where $W(x_1,\ldots,x_n,y_1,\ldots,y_m)$ is a quantifier-free formula with free variables $x_1,\ldots,x_n,y_1,\ldots,y_m$ and moreover W contains no function signs. For brevity we shall often denote a typical such query by $<\vec{x}/\vec{\tau}|(E\vec{y}/\vec{\theta})W>$. The τ's and θ's are called types. We assume that with each type τ is associated a set of constant signs which we denote by $|\tau|$. For example, in an airline data base, |City| might be {Toronto, Boston, Paris,...,}. If $\vec{\tau} = \tau_1,\ldots,\tau_n$ is a sequence of types we denote by $|\vec{\tau}|$ the set $|\tau_1| \times\ldots\times |\tau_n|$.

A data base (DB) is a set of clauses containing no function signs. For an airline data base, DB might contain such information as:

"Air Canada flight 203 connects Toronto and Vancouver."

Connect AC203, Toronto, Vancouver

"All flights from Boston to Los Angeles serve meals."

(x/Flight)Connect x,Boston,LA $\supset$ Meal-serve x

The restriction that neither queries nor clauses of a data base contain function signs may be removed without significantly affecting the results of this paper. Because of space limitation, we have chosen to present a simplified theory in which the only possible objects are constants.

Let $Q = \langle \vec{x}/\vec{\tau} | (E\vec{y}/\vec{\theta})W(\vec{x},\vec{y}) \rangle$ and let DB be a data base. A set of n-tuples of constant signs $\{\vec{c}^{(1)},\ldots,\vec{c}^{(r)}\}$ is an answer to Q (with respect to DB) iff

1. $\vec{c}^{(i)} \in |\vec{\tau}|$ $i = 1,\ldots,r$ and
2. $DB \vdash \bigvee_{i \leq r} (E\vec{y}/\vec{\theta})W(\vec{c}^{(i)},\vec{y})$

Notice that if $\{\vec{c}^{(1)},\ldots,\vec{c}^{(r)}\}$ is an answer to Q, and $\vec{c}$ is any n-tuple of constant signs satisfying 1. then so also is $\{\vec{c}^{(1)},\ldots,\vec{c}^{(r)},\vec{c}\}$ an answer to Q. This suggests the need for the following definitions:

An answer A to Q is minimal iff no proper subset of A is an answer to Q. If A is a minimal answer to Q, then if A consists of a single n-tuple, A is a definite answer to Q. Otherwise, A is an indefinite answer to Q. Finally define $\|Q\|_{OWA}$ to be the set of minimal answers to Q. (For reasons which will become apparent later, the subscript OWA stands for "Open World Assumption".) Notice the interpretation assigned to an indefinite answer $\{\vec{c}^{(1)},\ldots,\vec{c}^{(r)}\}$ to Q: $\vec{x}$ is either $\vec{c}^{(1)}$ or $\vec{c}^{(2)}$ or...or $\vec{c}^{(r)}$ but there is no way, given the information in DB, of determining which. Instead of denoting an answer as a set of tuples $\{\vec{c}^{(1)},\ldots,\vec{c}^{(r)}\}$ we prefer the more suggestive notation $\vec{c}^{(1)} +\ldots+ \vec{c}^{(r)}$, a notation we shall use in the remainder of this paper.

Example 2.1

Suppose DB knows of 4 humans and 2 cities:

$|Human| = \{a,b,c,d\}$ $|City| = \{B,V\}$

Suppose further that everyone is either in B or in V:

(x/Human)Loc x,B V Loc x,V

and moreover, a is in B and b is in V:

Loc a,B Loc b,V

Then for the query "Where is everybody?"

$Q = \langle x/Human, y/City | Loc\ x,y \rangle$

we have

$\|Q\|_{OWA}$ = {(a,B), (b,V), (c,B) + (c,V), (d,B) + (d,V)}

i.e. a is in B, b is in V, c is either in B or V, and d is either in B or V.

Since it is beyond the scope of this paper, the reader is referred to [Reiter 1977a] or [Reiter 1977b] for an approach to query evaluation which returns $\|Q\|_{OWA}$ given any query Q.

3. THE CLOSED WORLD ASSUMPTION

In order to illustrate the central concept of this paper, we consider the following purely extensional data base (i.e. a data base consisting of ground literals only):

|Teacher| = {a,b,c,d}

|Student| = {A,B,C}

Teach|

a	A
b	B
c	C
a	B

Now consider the query: Who does not teach B?

Q = <x/Teacher|$\overline{\text{Teach}}$ x,B>

By the definition of the previous section, we conclude, counterintuitively, that $\|Q\|_{OWA} = \phi$.

Intuitively, we want {c,d} i.e. |Teacher| - $\|$<x/Teacher|Teach x,B>$\|_{OWA}$.

The reason for the counterintuitive result is that first order logic interprets the DB literally; all the logic knows for certain is what is explicitly represented in the DB. Just because $\overline{\text{Teach}}$ c,B is not present in the DB is no reason to conclude that $\overline{\text{Teach}}$ c,B is true. Rather, as far as the logic is concerned, the truth of Teach c,B is unknown! Thus, we would also have to include the following facts about Teach:

Teach	
a	C
b	A
b	C
c	A
c	B
d	A
d	B
d	C

Unfortunately, the number of negative facts about a given domain will, in general, far exceed the number of positive ones so that the requirement that all facts, both positive and negative, be explicitly represented may well be unfeasible. In the case of purely extensional data bases there is a ready solution to this problem. Merely explicitly represent positive facts. A negative fact is implicitly present provided its positive counterpart is not explicitly present. Notice, however, that by adopting this convention, we are making an assumption about our knowledge about the domain, namely, that we know everything about each predicate of the domain. There are no gaps in our knowledge. For example, if we were ignorant as to whether or not a teaches C, we could not permit the above implicit representation of negative facts. This is an important point. The implicit representation of negative facts presumes total knowledge about the domain being represented. Fortunately, in most applications, such an assumption is warranted. We shall refer to this as the closed world assumption (CWA). Its opposite, the open world assumption (OWA), assumes only the information given in the data base and hence requires all facts, both positive and negative, to be explicitly represented. Under the OWA, "gaps" in one's knowledge about the domain are permitted.

Formally, we can define the notion of an answer to a query under the CWA as follows:

Let DB be an extensional data base and let

$\overline{EDB}$ = $\{\bar{P}\vec{c} \mid$ P is a predicate sign, $\vec{c}$ a tuple of constant signs and $P\vec{c} \notin DB\}$

Then $\vec{c}$ is a CWA answer to $\langle \vec{x}/\vec{\tau} \mid (E\vec{y}/\vec{\theta})W(\vec{x},\vec{y})\rangle$ (with respect to DB) iff

1. $\vec{c} \in |\vec{\tau}|$ and
2. $DB \cup \overline{EDB} \vdash (E\vec{y}/\vec{\theta})W(\vec{c},\vec{y})$

For purely extensional data bases, the CWA poses no difficulties. One merely imagines the DB to contain all negative facts each of which has no positive version in the DB. This conceptual view of the DB fails in the presence of non ground clauses. For if $P\vec{c} \notin DB$, it may nevertheless be possible to infer $P\vec{c}$ from the DB, so that we cannot, with impunity, imagine $\bar{P}\vec{c} \in DB$. The obvious generalization is to assume that the DB implicitly contains $\bar{P}\vec{c}$ whenever it is not the case that $DB \vdash P\vec{c}$.

Formally, we can define the notion of an answer to a query under the CWA for an arbitrary data base DB as follows:

Let

$\overline{EDB}$ = $\{\bar{P}\vec{c} \mid$ P is a predicate sign, $\vec{c}$ a tuple of constant signs and $DB \nvdash P\vec{c}\}$

Then $\vec{c}^{(1)} + \ldots + \vec{c}^{(r)}$ is a CWA answer to

$\langle \vec{x}/\vec{\tau} \mid (E\vec{y}/\vec{\theta})W(\vec{x},\vec{y})\rangle$ (with respect to DB) iff

1. $\vec{c}^{(i)} \in |\vec{\tau}|$ $\quad i = 1,\ldots,r$ and
2. $DB \cup \overline{EDB} \vdash \bigvee_{i \leq r} (E\vec{y}/\vec{\theta})W(\vec{c}^{(i)},\vec{y})$

This definition should be compared with the definition of an answer in Section 2. We shall refer to this latter notion as an OWA answer. As under the OWA, we shall require the notions of minimal, indefinite and definite CWA answers. If Q is a query, we shall denote the set of minimal CWA answers to Q by $\|Q\|_{CWA}$.

Example 3.1

We consider a fragment of an inventory data base.

1. Every supplier of a part supplies all its subparts.

   (x/Supplier)(yz/Part)Supplies x,y ∧ Subpart z,y ⊃ Supplies x,z

2. Foobar Inc. supplies all widgets.

   (x/Widget)Supplies Foobar,x

3. The subpart relation is transitive.

   (xyz/Part)Subpart z,y ∧ Subpart y,x ⊃ Subpart z,x

Assume the following type extensions:

|Supplier| = {Acme, Foobar, AAA}

|Widget| = $\{w_1, w_2, w_3, w_4\}$

|Part| = $\{p_1, p_2, p_3, w_1, w_2, w_3, w_4\}$

Finally, assume the following extensional data base:

Supplies	x	y
	Acme	p_1
	AAA	w_3
	AAA	w_4

Subpart	x	y
	p_2	p_1
	p_3	p_2
	w_1	p_1
	w_2	w_1

Then $\overline{EDB}$ is:

$\overline{\text{Supplies}}$	x	y
	Acme	w_3
	Acme	w_4
	AAA	p_1
	AAA	p_2
	AAA	p_3
	AAA	w_1
	AAA	w_2
	Foobar	p_1
	Foobar	p_2
	Foobar	p_3
	p_1	Acme
	p_1	AAA
	p_1	Foobar
	p_1	p_1
	p_1	p_2
	p_1	p_3
	p_1	w_1
	etc.	

$\overline{\text{Subpart}}$	x	y
	p_1	p_1
	p_1	p_2
	p_1	p_3
	p_1	w_1
	p_1	w_2
	p_1	w_3
	p_1	w_4
	p_2	p_2
	p_2	p_3
	p_2	w_1
	p_2	w_2
	p_2	w_3
	p_2	w_4
	p_3	p_3
	p_3	w_1
	p_3	w_2
	p_3	w_3
	p_3	w_4
	etc.	

The notion of a CWA answer is obviously intimately related to the negation operators of PLANNER [Hewitt 1972] and PROLOG [van Emden and Kowalski 1976] since in these languages, negation means "not provable" and the definition of $\overline{EDB}$ critically depends upon this notion.

Notice that under the CWA, there can be no "gaps" in our knowledge about the domain. More formally, for each predicate sign P and each tuple of constant signs $\vec{c}$, either $DB \vdash P\vec{c}$ or $\overline{EDB} \vdash \overline{P}\vec{c}$ and since, under the CWA the data base is taken to be $DB \cup \overline{EDB}$, we can always infer either $P\vec{c}$ or $\overline{P}\vec{c}$ from $DB \cup \overline{EDB}$. Since there are no "knowledge gaps" under the CWA, it should be intuitively clear that indefinite CWA answers cannot arise, i.e. each minimal CWA answer to a query is of the form $\vec{c}$. The following result confirms this intuition.

Theorem 3.1

Let $Q = \langle \vec{x}/\vec{\tau} \mid (E\vec{y}/\vec{\theta})W(\vec{x},\vec{y}) \rangle$. Then every minimal CWA answer to Q is definite.

There is one obvious difficulty in directly applying the definition of a CWA answer to the evaluation of queries. The definition requires that we explicitly know $\overline{EDB}$ and, as Example 3.1 demonstrates, the determination of $\overline{EDB}$ is generally non trivial. In any event, for non toy domains, $\overline{EDB}$ would be so large that its explicit representation would be totally unfeasible. Fortunately, as we shall see in the next section, there is no need to know the elements of $\overline{EDB}$ i.e. it is possible to determine the set of closed world answers to an arbitrary query Q by appealing only to the given data base DB.

4. QUERY EVALUATION UNDER THE CWA

It turns out that the CWA admits a number of significant simplifications in the query evaluation process. The simplest of these permits the elimination of the logical connectives $\wedge$ and $\vee$ in favour of set intersection and union respectively, as follows:

Theorem 4.1

1. $\|\langle \vec{x}/\vec{\tau} | (E\vec{y}/\vec{\theta})W_1 \vee W_2 \rangle\|_{CWA} = \|\langle \vec{x}/\vec{\tau} | (E\vec{y}/\vec{\theta})W_1 \rangle\|_{CWA} \cup \|\langle \vec{x}/\vec{\tau} | (E\vec{y}/\vec{\theta})W_2 \rangle\|_{CWA}$
2. $\|\langle \vec{x}/\vec{\tau} | W_1 \wedge W_2 \rangle\|_{CWA} = \|\langle \vec{x}/\vec{\tau} | W_1 \rangle\|_{CWA} \cap \|\langle \vec{x}/\vec{\tau} | W_2 \rangle\|_{CWA}$

Notice that in the identity 2. the query must be quantifier free.

Notice also that the identities of Theorem 4.1 fail under the OWA.

To see why, consider the following:

Example 4.1

$|\tau| = \{a\}$

DB: $Pa \vee Ra$

$Q = \langle x/\tau | Px \vee Rx \rangle$

$\|Q\|_{OWA} = \{a\}$

but

$\|\langle x/\tau | Px \rangle\|_{OWA} = \|\langle x/\tau | Rx \rangle\|_{OWA} = \phi$

Example 4.2

$|\tau| = \{a,b\}$

DB: $Pa \vee Pb$, Ra, Rb

$Q = \langle x/\tau | Px \wedge Rx \rangle$

$\|Q\|_{OWA} = \{a+b\}$

but

$\|\langle x/\tau | Px \rangle\|_{OWA} = \{a+b\}$

$\|\langle x/\tau | Rx \rangle\|_{OWA} = \{a,b\}$

One might also expect that all occurrences of negation can be eliminated in favour of set difference for CWA query evaluation. This is indeed the case, but only for quantifier free queries and then only when DB ∪ $\overline{EDB}$ is consistent.

Theorem 4.2

If W, W_1 and W_2 are quantifier free, and DB ∪ $\overline{EDB}$ is consistent, then

1. $\|\langle\vec{x}/\vec{\tau}|\overline{W}\rangle\|_{CWA} = |\vec{\tau}| - \|\langle\vec{x}/\vec{\tau}|W\rangle\|_{CWA}$
2. $\|\langle\vec{x}/\vec{\tau}|W_1 \wedge \overline{W}_2\rangle\|_{CWA} = \|\langle\vec{x}/\vec{\tau}|W_1\rangle\|_{CWA} - \|\langle\vec{x}/\vec{\tau}|W_2\rangle\|_{CWA}$

To see why Theorem 4.2 fails for quantified queries, consider the following:

Example 4.3

$|\tau| = \{a,b\}$

DB: Pa,a

Then $\overline{EDB} = \{\overline{P}a,b,\ \overline{P}b,a,\ \overline{P}b,b\}$

Let $Q(P) = \langle x/\tau|(Ey/\tau)Px,y\rangle$

$Q(\overline{P}) = \langle x/\tau|(Ey/\tau)\overline{P}x,y\rangle$

Then $\|Q(P)\|_{CWA} = \{a\}$

$\|Q(\overline{P})\|_{CWA} = \{a,b\} \neq |\tau| - \|Q(P)\|_{CWA}$

Notice also that Theorem 4.2 fails under the OWA.

By an atomic query we mean any query of the form $\langle\vec{x}/\vec{\tau}|(E\vec{y}/\vec{\theta})Pt_1,\ldots,t_n\rangle$ where P is a predicate sign and each t is a constant sign, an x, or a y.

Theorems 4.1 and 4.2 assure us that for quantifier free queries, CWA query evaluation can be reduced to the Boolean operations of set intersection union and difference applied to atomic queries. However, we can deal with quantified queries by introducing the following projection operator [Codd 1972]:

Let $Q = \langle \vec{x}/\vec{\tau}, z/\psi | W \rangle$ where W is a possibly existentially quantified formula, and $\vec{x}$ is the n-tuple $x_1,\ldots,x_n$. Then $||Q||_{CWA}$ is a set of (n+1)-tuples, and the projection of $||Q||_{CWA}$ with respect to z, $\pi_z||Q||_{CWA}$, is that set of n-tuples obtained from $||Q||_{CWA}$ by deleting the (n+1)st component from each (n+1)-tuple of $||Q||_{CWA}$.

For example, if $Q = \langle x_1/\tau_1, x_2/\tau_2, z/\psi | W \rangle$

and if

$||Q||_{CWA} = \{(a,b,c),(a,b,d),(c,a,b)\}$

then

$\pi_z||Q||_{CWA} = \{(a,b),(c,a)\}$

Theorem 4.3

$||\langle \vec{x}\ \vec{\tau} | (E\vec{y}/\vec{\theta})W \rangle||_{CWA} = \pi_{\vec{y}}||\langle \vec{x}/\vec{\tau}, \vec{y}/\vec{\theta} | W \rangle||_{CWA}$

where $\pi_{\vec{y}}$ denotes $\pi_{y_1}\pi_{y_2}\ldots\pi_{y_m}$

Corollary 4.4

1. $||\langle \vec{x}/\vec{\tau} | (E\vec{y}/\vec{\theta})\overline{W} \rangle||_{CWA} = \pi_{\vec{y}}||\langle \vec{x}/\vec{\tau}, \vec{y}/\vec{\theta} | \overline{W} \rangle||_{CWA}$
$= \pi_{\vec{y}}(|\vec{\tau}| \times |\vec{\theta}| - ||\langle \vec{x}/\vec{\tau}, \vec{y}/\vec{\theta} | W \rangle||_{CWA})$

2. $||\langle \vec{x}/\vec{\tau} | (E\vec{y}/\vec{\theta})W_1 \wedge W_2 \rangle||_{CWA} = \pi_{\vec{y}}(||\langle \vec{x}/\vec{\tau}, \vec{y}/\vec{\theta} | W_1 \rangle||_{CWA} \cap ||\langle \vec{x}/\vec{\tau}, \vec{y}/\vec{\theta} | W_2 \rangle||_{CWA})$

Thus, in all cases, an existentially quantified query may be decomposed into atomic queries each of which is evaluated under the CWA. The resulting sets of answers are combined under set union, intersection and difference, but only after the projection operator is applied, if necessary.

Example 4.4

$||\langle x/\tau | (Ey/\theta)Px,y \vee Qx,y\ Rx,y \rangle||_{CWA}$

$= ||\langle x/\tau | (Ey/\theta)Px,y||_{CWA} \cup \pi_y(||\langle x/\tau, y/\theta | Qx,y \rangle||_{CWA} \cap ||\langle x/\tau, y/\theta | Rx,y \rangle||_{CWA})$

$||\langle x/\tau | PxQx \vee \overline{Rx} \rangle||_{CWA}$

$= ||\langle x/\tau | Px \rangle||_{CWA} \cap ||\langle x/\tau | Qx \rangle||_{CWA} \cup [|\tau| - ||\langle x/\tau | Rx \rangle||_{CWA}]$

$$\|\langle x/\tau \mid (Ey/\theta)Px,y \vee Qx,y\ \bar{R}x,y\rangle\|_{CWA}$$

$$= \|\langle x/\tau \mid (Ey/\theta)Px,y\rangle\|_{CWA} \cup \pi_y(\|\langle x/\tau, y/\theta \mid Qx,y\rangle\|_{CWA} - \|\langle x/\tau, y/\theta \mid Rx,y\rangle\|_{CWA})$$

In view of the above results, we need consider CWA query evaluation only for atomic queries.

We shall say that DB is consistent with the CWA iff DB $\cup$ $\overline{EDB}$ is consistent.

Theorem 4.5

Let Q be an atomic query. Then if DB is consistent with the CWA, $\|Q\|_{CWA} = \|Q\|_{OWA}$

Theorem 4.5 is the principal result of this section. When coupled with Theorems 4.1 and 4.2 and the remarks following Corollary 4.4 it provides us with a complete characterization of the CWA answers to an arbitrary existential query Q in terms of the application of the operations of projection, set union, intersection and difference as applied to the OWA answers to atomic queries. In other words, CWA query evaluation has been reduced to OWA atomic query evaluation. A consequence of this result is that we need never known the elements of $\overline{EDB}$. CWA query evaluation appeals only to the given data base DB.

Example 4.5

We consider the inventory data base of Example 3.1. Suppose the following query:

$$Q = \langle x/Supplier \mid (Ey/Widget)Supplies\ x,y \wedge Subpart\ y,p_1 \wedge \overline{Supplies}\ x,p_3\rangle$$

Then

$$\|Q\|_{CWA} = \pi_y(\|Q_1\|_{OWA} \cap \|Q_2\|_{OWA}) \cap (|Supplier| - \|Q_3\|_{OWA})$$

where

$$Q_1 = \langle x/Supplier,\ y/Widget \mid Supplies\ x,y\rangle$$

Q_2 = <x/Supplier, y/Widget|Subpart y,p_1>

Q_3 = <x/Supplier|Supplies x,p_3>

It is easy to see that

$||Q_1||_{OWA}$ = {(Foobar,w_1), (Foobar,w_2), (Foobar,w_3),
(Foobar,w_4), (AAA,w_3) (AAA,w_4), (Acme,w_1), (Acme,w_2)}

$||Q_2||_{OWA}$ = {(Acme,w_1), (Acme,w_2), (AAA,w_1), (AAA,w_2), (Foobar,w_1), (Foobar,w_2)}

$||Q_3||_{OWA}$ = {Acme}

whence

$\pi_y(||Q_1||_{OWA} \cap ||Q_2||_{OWA})$ = {Foobar,Acme}

and

|Supplier| - $||Q_3||_{OWA}$ = {Foobar,AAA}

Hence

$||Q||_{CWA}$ = {Foobar}.

5. ON DATA BASES CONSISTENT WITH THE CWA

Not every consistent data base remains consistent under the CWA.

Example 5.1

DB: Pa v Pb

Then, since DB $\not\vdash$ Pa and DB $\not\vdash$ Pb, $\overline{EDB}$ = {$\overline{Pa}$, $\overline{Pb}$} so that DB $\cup$ $\overline{EDB}$ is inconsistent.

Given this observation, it is natural to seek a characterization of those data bases which remain consistent under the CWA. Although we know of no such characterization, it is possible to give a sufficient condition for CWA consistency which encompases a large natural class of data bases, namely the Horn data bases. (A data base is Horn iff every clause is Horn i.e. contains at most one positive literal. The data base of Example 3.1 is Horn.)

Theorem 5.1

Suppose DB is Horn, and consistent. Then DB $\cup$ $\overline{EDB}$ is consistent i.e. DB is consistent with the CWA.

Following [van Emden 1977] we shall refer to a Horn clause with exactly one positive literal as a definite clause. If DB is Horn, let Δ(DB) be obtained from DB by removing all non definite clauses i.e. all negative clauses. The following Theorem demonstrates the central importance of these concepts:

Theorem 5.2

If $Q = \langle \vec{x}/\vec{\tau} \mid (E\vec{y}/\vec{\theta})W \rangle$ and DB is Horn and consistent, then $\|Q\|_{CWA}$ when evaluated with respect to DB yields the same set of answers as when evaluated with respect to Δ(DB). In other words, negative clauses in DB have no influence on CWA query evaluation.

Theorem 5.2 allows us, when given a consistent Horn DB, to discard all its negative clauses without affecting CWA query evaluation. Theorem 5.2 fails for non Horn DBs, as the following example demonstrates:

Example 5.2

DB: $\overline{P}a \vee \overline{R}a$, $Ra \vee Sa$, Pa

Then DB $\vdash$ Sa

But $\Delta(DB) = \{Ra \vee Sa, Pa\}$ and $\Delta(DB) \nvdash Sa$.

Let us call a data base for which all clauses are definite a definite data base.

Theorem 5.3

If DB is definite then DB is consistent.

Corollary 5.4

If DB is definite then

(i) DB is consistent

(II) DB is consistent with the CWA.

Corollary 5.4 is a central result. It guarantees data base and CWA consistency for a large and natural class of data bases. Since the data base of Example 3.1 is definite we are assured that it is consistent with the CWA.

In [van Emden 1977], van Emden addresses, from a semantic point of view, the issues of data base consistency under the CWA. He defines the notion of a "minimal model" for a data base as the intersection of all its models. If this minimal model is itself a model of the data base, then the data base is consistent with the CWA. Van Emden goes on to point out some intriguing connections between minimal models and Scott's minimal fixpoint approach to the theory of computation, results which are elaborated in [van Emden and Kowalski 1976].

6. SUMMARY

We have introduced the notion of the closed world assumption for deductive question-answering. This says, in effect, "everything that you don't know to be true may be assumed false". We have then shown how query evaluation under the closed world assumption reduces to the usual first order proof theoretic approach to query evaluation as applied to atomic queries. Finally, we have shown that consistent Horn data bases remain consistent under the closed world assumption and that definite data bases are consistent with the closed world assumption.

ACKNOWLEDGEMENT

This paper was written with the financial support of the National Research Council of Canada under grant A7642. I wish to thank Craig Bishop for his careful criticism of an earlier draft of this paper.

REFERENCES

Codd, E.F. (1972). "Relational Completeness of Data Base Sublanguages," in Data Base Systems, R. Rustin (Ed.), Prentice-Hall, Englewood Cliffs, N.J., 1972, 65-98.

van Emden, M.H. (1977). "Computation and Deductive Information Retrieval," Dept. of Computer Science, University of Waterloo, Waterloo, Ont., Research Report CS-77-16, May 1977.

van Emden, M.H., and Kowalski, R.A. (1976). "The Semantics of Predicate Logic as a Programming Language," J.ACM, 23 (Oct. 1976), 733-742.

Hewitt, C. (1972). Description and Theoretical Analysis (Using Schemata) of PLANNER: A Language for Proving Theorems and Manipulating Models in a Robot, AI Memo No. 251, MIT Project MAC, Cambridge, Mass., April 1972.

Reiter, R. (1977a). An Approach to Deductive Question-Answering, Technical report, Bolt, Beranek and Newman Inc., Cambridge, Mass., September 1977, 161 pp.

Reiter, R. (1977b). Deductive Question-Answering on Relational Data Bases, Technical Report 77-15, Department of Computer Science, Univ. of British Columbia Vancouver, B.C., Oct. 1977.