THE CONSISTENCY OF NEGATION AS FAILURE

TIM FLANNAGAN

▷ Clark's attempt [1] to validate negation as failure in first order logic is shown to contain some fundamental errors. In particular, we show that the motivation for the completed database, the definition of the completed database, and the attempt to validate negation as failure in terms of it are illogical, that the completed database cannot be regarded as the intended meaning of the database, and that the closed world assumption is generally absurd and, in any case, irrelevant. A validation is given using a consistent first order extension of the database and hence in the only terms which appear to make any sense, namely, consistency with the database. However, it seems that the query evaluation process, with negation interpreted as failure, is of no practical use as a theorem prover. ◁

INTRODUCTION

The paper is in two parts. Section I discusses some fundamental errors in Clark's pioneering attempt [1] to validate negation as failure (NAF) in terms of first order deduction from the completed database (CDB). Section II presents a validation of NAF using a consistent first order extension (DB*) of the database (DB). The query evaluation process (QEP) is shown to be a theorem prover for DB* and hence a consistency prover for database queries or their negations. Like the CDB, DB* cannot, however, be regarded as the intended meaning of DB. Its role in the argument is therefore catalytic, and the essential function of the QEP, with negation interpreted as failure, is as a consistency prover. Thus, NAF is validated in first order logic without assigning to the database any meaning beyond itself. In particular, it is unnecessary to assume the free interpretation of database terms, as in the CDB.

Address correspondence to Tim Flannagan, Logica Cambridge Ltd, Betjeman House, 104 Hills Road, Cambridge CB2 1LQ, England.
Received 18 September 1985; accepted 23 September 1985.

THE JOURNAL OF LOGIC PROGRAMMING

52 Vanderbilt Ave., New York, NY 10017 0743-1066/86/$03.50

To assist readability, proofs of theorems in Section II are given in the appendix.

The reader is assumed to be familiar with [1] and [6], and familiarity with [7] would be useful.

I. CLARK'S ATTEMPT TO VALIDATE NAF

We begin with notation. DB will denote an arbitrary database of the type specified in [1] (pp. 297–298), i.e., a database consisting of formulae of the form

$$R(t_1,\ldots,t_n) \leftarrow L_1 \wedge \cdots \wedge L_m, \qquad n, m \geq 0,$$

where the L_i are atoms or the negations of atoms (literals). CWA will denote the formal theory of Reiter's closed world assumption [2], as defined in [7]. Note that there is a distinction between the informal assumption of the closed world and the formal theory representing it. $\mathscr{L}$(DB) will denote the first order language of DB; Q an arbitrary query in $\mathscr{L}$(DB), i.e., a formula of the form

$$L_1 \wedge \cdots \wedge L_k,$$

where the L_i are literals; and if A is a first order formula with free variables $x_1,\ldots,x_n$, then $\forall[A]$ will denote the universal closure of A, i.e., $\forall x_1 \cdots \forall x_n A$.

NAF is an ad hoc rule, and in order to imbue it with logical significance and thereby justify its application to any DB, Clark proved the following metatheorem.

Theorem 1. For any DB *and any query* Q,

if Q *fails in the QEP, then* $\mathrm{CDB} \vdash \forall[\neg Q]$, *and*

if Q *succeeds with answer* θ, *then* $\mathrm{CDB} \vdash \forall[Q\theta]$.

He called this result the "soundness" of NAF.

Theorem 1 is not strictly correct, for the CDB is not well-defined (pp. 303–304 of [1]). Every theory has a particular vocabulary, i.e., a set of function and predicate symbols, and theories with different vocabularies are different theories. Clark's definition of the CDB assumes the existence of a universal vocabulary, and a basic result of set theory shows that there is no such thing. Thus, if DB is any database and P is a 0-place predicate not in the vocabulary of CDB, then P fails in the QEP and, contrary to Theorem 1, P is not a theorem of CDB, since no sentence can be a theorem of a theory unless it is in the language of the theory.

Thus, in order for NAF to be interpreted in terms of the CDB, as in Theorem 1, the QEP should first check that queries are in the language of CDB, whatever that language is. Since the CDB is not explicitly defined by DB, this is only possible if the vocabulary of CDB is finite and identical to that of DB. In this case, the check is a terminating, recursive procedure, provided that formulae in DB are well formed in the standard way. This is not the case if, for example, the expressions t_i in the formula

$$R(t_1,\ldots,t_n)$$

above are allowed to be formulae, as with some forms of PROLOG. Nevertheless, Theorem 1 cannot validate NAF for an arbitrary DB, for if CDB is inconsistent, then the theorem is trivial because every sentence in the language of an inconsistent

theory T is a theorem of T. In particular, if CDB is inconsistent, then, whether Q fails, succeeds, flounders, or has an infinite evaluation tree, both Q and $\neg Q$ are theorems of CDB. At best, therefore, Theorem 1 can only justify the application of NAF to those DBs for which the CDB is consistent, and then only when the definition of the CDB is modified as above.

Thus, Clark overlooked the fundamental logical principle that, roughly speaking, no proposition can be justified on the basis of derivation from an inconsistent theory. Nowhere in his attempt to justify NAF is there mention of the consistency of CDB. It is mentioned on p. 317 in relation to the question of the "completeness" of NAF (the converse of Theorem 1), and quite rightly, for, if NAF is "complete", in the sense of [1], and CDB is inconsistent, then every query Q both fails and succeeds, which is impossible [7, Theorem 4]. Thus, the consistency of CDB is necessary both to validation by "soundness" and to "completeness", and we have established the following.

Theorem 2. NAF is "incomplete" for DB*s with inconsistent* CDB*s.*

Now, it is easy to find DBs whose CDBs are inconsistent. Take, for instance, the following.

DB_1: $P \leftarrow \neg P$

The CDB is

CDB_1: $P \leftrightarrow \neg P$,

which is a contradiction. However, the only atomic formula in the language of DB_1 is P, and the query P neither succeeds nor fails in the QEP. In this case, it goes into an infinite loop. Hence, DB_1 does not satisfy the assumption of Theorem 1, namely, that its language supports at least one query, which either succeeds or fails.

DB_1 therefore does not invalidate Clark's claim that Theorem 1 validates NAF.

Notice that an inconsistent CDB can always be obtained simply by adding

$P \leftarrow \neg P$

to DB, where P is a new 0-place predicate, but again, this isn't a counterexample.

Nevertheless, it is easy to find DBs and queries which clearly satisfy the assumption of Theorem 1, and for which the CDB is inconsistent. For example, let DB_2 be the following database.

DB_2: $P(a, b)$,

$P(x, y) \leftarrow \neg P(y, x)$.

The query $P(b, a)$ is "allowed" [1, p. 317], since the negative literal $\neg P(a, b)$ is closed; i.e., the variables y, x in $\neg P(y, x)$ are instantiated to constants (in this case, by the query). The query clearly fails. However, the CDB is

CDB_2: $\forall x, y[P(x, y) \leftrightarrow (x = a \wedge y = b) \vee \neg P(y, x)]$,

$a \neq b$.

This is inconsistent, for, putting $x = y = a$, it follows that

$P(a, a) \leftrightarrow \neg P(a, a)$,

which is a contradiction.

Incidentally, we abjure Clark's use of the word "ground" as in "ground literal" since it duplicates the word "closed", which is standard in mathematical logic and applies not just to literals but to any well-formed expression; a closed expression being one with no free variables.

Considering the simplicity of DB_2, we are bound to conclude that, if DB contains negative literals, and particularly if it is at all complex, then, even if its language contains queries which succeed or fail, the CDB is highly likely to be inconsistent. Theorem 1 is therefore useless as a justification of NAF for DBs with negative literals.

Now, to say that NAF is valid is to say that it is valid for all DBs. Contrary to Clark's claim, we therefore have the following.

Theorem 3. The "soundness" of NAF does not validate it.

Likewise, to say that NAF is "complete" is to say that it is "complete" for all DBs. From Theorem 2 and from what we have just shown, it follows that NAF is "incomplete" for DBs with negative literals. Thus, we also have

Theorem 4. NAF is "incomplete".

It is not only inconsistency of the CDB which renders NAF "incomplete". Clark [1, p. 316] gives an example of a DB and an atom Q such that $DB \vdash Q$ although Q neither succeeds nor fails but has an infinite evaluation tree for every selection rule and, indeed, every ordering of the clauses. Moreover, it is easily seen that the CDB is consistent. Thus, the "incompleteness" of NAF is implied by its incompleteness in the more usual sense that queries don't always succeed or fail.

Since $DB \vdash Q$ in this example, we have the following result.

Theorem 5. There is no first order extension of DB *for which NAF is "complete".*

Shepherdson [8, Appendix, p. 3] gave a stronger result about weak completeness.

Now, Jaffar et al. showed in [4] (see also [5]) that NAF is "complete" for DBs with no negated literals, i.e., for definite Horn formula DBs. Their result is therefore as strong as possible. It implies incidentally that the CDB is consistent for positive Horn formula DBs. A direct proof of this is given in the appendix, but it therefore follows from Theorem 1 (with the extra condition on the QEP mentioned above) that NAF is valid for definite Horn formula DBs. On account of Theorem 3, this result is also optimal. However, as Shepherdson [8] pointed out, it isn't very useful.

Shepherdson [7] also showed that NAF is "sound" in terms of the CWA as well as the CDB. That is, he proved a result like Theorem 1 with CWA in place of CDB. However, in the database

DB_3: $P(x) \leftarrow \neg R(x)$,
$R(x) \leftarrow \neg S(x)$,
$T(a)$

the queries $\neg P(a)$ and $R(a)$ are both "allowed" by the QEP, and both fail. Since neither $P(a)$ nor $R(a)$ is a theorem of DB_3, the CWA includes the sentences $\neg P(a)$ and $\neg R(a)$ as well as the axioms of DB_3. Thus the CWA is inconsistent.

Shepherdson did not claim, of course, that his theorem validates NAF, but it is worth noting that the above example shows that NAF cannot be validated by the CWA any more than it can be validated by the CDB.

It seems that Theorem 1 was called the "soundness" of NAF because it was thought to validate it. Now, both words "sound" and "complete" have been taken from mathematical logic, where neither concept depends on the notion of consistency, since they refer to logical calculi and their rules of inference rather than particular sets of formulae. We say that a calculus is sound if, for any set of sentences $X \cup \{A\}$,

$$X \vdash A \quad \text{implies} \quad X \models A,$$

and complete if the converse holds, where $X \vdash A$ means that there is a deduction of A from X, and $X \models A$ means that every model of X is a model of A.

Thus, the word "sound" is used in logic because it is regarded as validating the rules of inference of the logic in question, i.e., validating their application to any set X of assumptions, whether X is consistent or not. Thus, since Theorem 1 does not validate NAF, it is inappropriate to call it the "soundness" of NAF and its converse the "completion".

The proof of Theorem 3 also raises some fundamental questions about the notion of the CDB.

The Questionable Value of CDB

Since DB is always consistent and CDB is highly likely to be inconsistent and in some cases is, it is illogical to suppose, as in [1] (p. 303), that CDB is the intended or implicit meaning of DB. Note that an inconsistency in CDB means that it contains a statement which directly contradicts a theorem of DB. For example, DB_2 is equivalent to

$$P(x, y) \leftarrow (x = a \wedge y = b) \vee \neg P(y, x),$$

which implies $P(a, a)$, and CDB_2 is got from DB_2 by adding to it, *inter alia*,

$$P(x, y) \rightarrow (x = a \wedge y = b) \vee \neg P(y, x),$$

which implies $\neg P(a, a)$.

Thus, the value of even the concept of CDB is suspect, and it is questionable whether any meaning should be given to DB beyond itself.

Notice that we are not saying that the value of a theory (in this case CDB) diminishes just because it might be inconsistent. Arithmetic might be inconsistent. We are saying that the value of a theory is seriously diminished if it is very likely to be inconsistent. The same goes for a class of theories, such as the class of CDBs, if instances of it are very likely to be inconsistent.

The point is that, if a theory is to be at all useful, it is necessary to at least be able to believe with good reason that it is consistent, even if one cannot prove its consistency. As Gödel showed, it is impossible to prove that arithmetic is consistent without invoking a stronger theory, but arithmetic has so far withstood all attempts to prove it inconsistent, so the world, give or take the odd logician, takes it on trust. (In fact, there are proofs of the consistency of strong and useful fragments of it.) In contrast, the simplicity of DB_2 shows that, in general, we are bound to disbelieve the consistency of CDB.

Thus, the notion of CDB is not a useful one. Nor, for the same reason, is CWA.

Even when CDB is consistent, it is an improbable theory because its equality axioms [1, p. 304] formally impose on DB an unnatural free interpretation of its terms, which trivializes CDB models. Shepherdson [8] illustrates this with a good example.

It is therefore worth looking at Clark's reason for considering the notion of CDB in the first place.

The Motivation for CDB

On page 294 of [1] there is the following sentence.

S: "Note that to assume that a relation instance is false if it is not implied, is to assume that the data base gives *complete* information about the true instances of its relations."

This latter assumption is Reiter's closed world assumption that, for atomic sentences, DB is implicitly complete. To be precise, it is the assumption that, for every atomic sentence A, if $\mathrm{DB} \nvdash A$, then DB implicitly implies $\neg A$ [6, p. 60]. The completion of a database is then defined accordingly.

S, however, is incorrect, for, if a sentence A is not a theorem of a theory T, one can assume the negation of A anyway, since it is consistent with T. Doing so implies nothing whatsoever about the completeness or otherwise of T. For example, consider the database on p. 294 of [1]:

$$\text{Maths-course}(\text{C101}) \cdot$$

$$\text{Maths-course}(\text{C301}),$$

which is equivalent to

$$\forall x\,[\text{Maths-course}(x) \leftarrow x = \text{C101} \vee x = \text{C301}].$$

Clark [1, p. 295] asserts that, in order to logically infer

$$\neg\text{Maths-course}(C)$$

for some constant C different from C101 and C301, it is necessary to assume the stronger statement

$$\forall x\,[\text{Maths-course}(x) \leftrightarrow x = \text{C101} \vee x = \text{C301}].$$

This is not true. In order to infer

$$\neg\text{Maths-course}(C)$$

it suffices to assume it, and one can assume it, since it is consistent with the database. If one wants the stronger statement, one can assume that as well, in this case, since it too is consistent with the database. However, as we saw with DB_2, the converse of an implication is not always consistent with it.

Moreover, in formal logic, it is meaningless to say that a theory is implicitly complete if it isn't actually complete. If it is actually complete and $T \nvdash A$, then it is unnecessary to assume $\neg A$, since it is implied.

Let us look more closely, then, at the closed world assumption.

The Closed World Assumption

Reiter formalized the closed world assumption by defining the theory DB + $\overline{\text{EDB}}$, and Clark formalized it as CDB.

Given the definitions of these theories, there is a deeper, more serious error in [6], and hence in [1], than the falsity of S, for the notions of truth and falsity are not definable in logic except in terms of satisfiability in models. Thus, "true" and "false" respectively mean true and false in some model.

Consider the definition of DB + $\overline{\text{EDB}}$ [6, p. 60], which, apart from some additional equality and domain closure axioms, is what Shepherdson [7] denotes by CWA. $\overline{\text{EDB}}$ is the set of all negative closed literals $\neg L$ such that $DB \nvdash L$, i.e., such that $DB + \neg L$ is consistent. DB + $\overline{\text{EDB}}$ is obtained from DB by simultaneously adding to it all the members of $\overline{\text{EDB}}$.

Now the separate consistency of two sentences A_1 and A_2 with a theory T does not imply their joint consistency with T. That is, the consistency of both $T + A_1$ and $T + A_2$ does not imply the consistency of $T + A_1 + A_2$, for, if A_1 is added to T, a new theory T' is formed, and the consistency of A_2 with T does not imply its consistency with T'. For example, let T be the database

DB: $P \leftarrow \neg R$.

Then $\overline{\text{EDB}}$ is the set $\{\neg P, \neg R\}$, since DB $\nvdash P$, DB $\nvdash R$. Although DB + $\neg P$ and DB + $\neg R$ are therefore both consistent, DB + $\neg P$ + $\neg R$ is inconsistent, since DB + $\neg R \vdash P$.

The point is that A_1 being true in one model and A_2 being true in another doesn't mean there is a model in which they are both true. In other words, one simply cannot say, as Reiter does, that if $T \nvdash A_1$ and $T \nvdash A_2$, then A_1 and A_2 can both be assumed to be false and that $\neg A_1$ and $\neg A_2$ are therefore both true, and can be added to T together.

Thus, in general, to say that a DB with negative literals is implicitly complete is to say that it is implicitly inconsistent and, hence, worthless as a set of assumptions. Generally, then, the assumption of implicit completeness is both unnecessary and absurd.

There is therefore no logical justification for Reiter's definition of DB + $\overline{\text{EDB}}$. It is not surprising that, even for simple DBs, DB + $\overline{\text{EDB}}$ can be inconsistent.

Certainly, if DB is a definite Horn formula database, then DB + $\overline{\text{EDB}}$ is consistent [7]. Perhaps the closed world assumption is even natural for some such databases, such as airline timetables, which are extensional [6, p. 60], but Reiter claimed [6, p. 60] that the closed world assumption is warranted for "most databases generally". This implies that, in most cases, DB + $\overline{\text{EDB}}$ is consistent, which is not true.

Now, first order theories are usually incomplete, and for such a theory T, there are often true statements in the language of T which are not theorems of T; i.e., there are sentences A such that $T + A$ and $T + \neg A$ are both consistent. It would be perverse to automatically assume such sentences to be false, even though it is consistent to do so.

To take a simple example, suppose the axioms of T are the order axioms

$$x < y \leftarrow \neg(y < x) \wedge y \neq x$$

$$x < z \leftarrow x < y \wedge y < z$$

Add a 1-place function symbol g and a constant c to the vocabulary of T, and let $\forall x(x < g(x))$ be the Skolem resolution of $\forall x \exists y(x < y)$. Then $T \nvdash c < g(c)$. Nevertheless, since $c < g(c)$ is implied by $\forall x \exists y(x < y)$, which, like the axioms of T, is true in the standard model consisting of the set of natural numbers with the usual ordering relation, it would be absurd to regard $c < g(c)$ as necessarily false. Thus, even if DB + $\overline{\text{EDB}}$ is consistent, Reiter's claim that the closed world assumption is "usually" warranted is extremely unlikely. In fact, like the assertion that $P(n)$ is true for most natural numbers n, where P is some predicate, the claim is both unprovable and irrefutable, and hence worthless.

Debate about whether or not the closed world assumption is warranted for a given DB is futile. It is warranted only if it is legitimate, and it is legitimate if and only if DB + $\overline{\text{EDB}}$ is consistent. Even if this is consistent, the assumption is unlikely if not perverse, and, as explained earlier, it is in any case irrelevant and based on the false assumption that separate consistency implies joint consistency.

The Completed Database

As we have seen, Clark assumed that DB is implicitly complete and hence that separate consistency implies joint consistency. This alone seriously weakens the credibility of CDB, but the same sort of error persists in the definition of CDB.

Apart from the equality axioms obtained from the free interpretation of terms in DB, CDB is essentially obtained from DB by doing two things: first, by replacing "if" throughout DB by "iff" ("if and only if"), i.e., $\leftarrow$ by $\leftrightarrow$, and second, by simultaneously adding the universal closure of the negations of all positive literals L for which there is no clause

$$L \leftarrow G_1 \wedge \cdots \wedge G_m, \qquad m \geq 0,$$

"about" L in DB [1, p. 303].

Now "if" is replaced by "iff", i.e., augmented with "only if", because implicit completeness says that atomic sentences are true "only if" they are implied by DB. This formal interpretation of implicit completeness is clearly ad hoc. The correct one is DB + $\overline{\text{EDB}}$, which, as we have seen, is generally inconsistent.

In fact, as with DB_2, one cannot expect to maintain consistency (recall that DB is always consistent) replacing even a single "if" statement with the corresponding "iff" statement. As in the definition of DB + $\overline{\text{EDB}}$, the situation is made even more precarious by making many such replacements at once, as well as by simultaneously adding the negations of all those literals for which there is no clause in the database. Since these replacements and additions are made only in order to make implicit completeness explicit [1, p. 295], and since the notion is unnecessary and generally invalid, there is no sound reason to make them anyway.

Thus, the definition of CDB is as illogical as that of the theory DB + $\overline{\text{EDB}}$.

The real illogicality in the definition of CDB is not that the universal closures of the negative literals are added to DB, but that all "if" statements are replaced by "iff" statements, for the theory obtained from DB by only adding the closures of the negative literals is consistent.

Let us call this theory the "extended database" (nothing to do with Reiter) and denote it by EDB.

Theorem 6. EDB *is consistent.*

The proof is given in the appendix.

Both Reiter and Clark, then, attempted to make explicit an assumption which is unnecessary, formally meaningless, and absurd, and did so by introducing fundamental illogicalities.

Now the obvious logical analogy of the NAF rule, which is

if L fails in the QEP, then $\neg L$ *succeeds,*

is that, if a sentence A is not a theorem of a theory T, then $\neg A$ can be assumed, i.e., $T + \neg A$ is consistent, which is inference at the meta-level of T. It would therefore have been more natural to try to justify NAF by showing that

if A fails, $\neg A$ *can be assumed*;

i.e.,

if A fails and DB *is consistent, then* DB + $\neg A$ *is consistent.*

Since DB is always consistent, this would amount to showing that

if A fails, then DB + $\neg A$ *is consistent,*

which would justify NAF in terms of meta-level inference, instead of in terms of object-level inference à la Theorem 1. It would also satisfy the minimum requirement of logic, namely, that assumptions be consistent with the database. Amongst other things, this is proved in Section II.

Given the illogical motivation for constructing CDB and its illogical definition, it is not surprising that there is confusion about the nature of NAF and that it is described rather curiously.

The Nature of NAF

NAF is commonly called, after Clark, a "meta-rule of inference" on account of Theorem 1, but the expression is a bad one, for its meaning is unclear. This, in turn, has fostered an unfortunate and somewhat pretentious mystique in logic programming circles, which is that a "meta-rule" is somehow extralogical and special. The expression "meta-rule" certainly has no standard logical meaning. The only meaning it can have in the context of [1] is that NAF or, more precisely, Theorem 1 is a meta-level statement (of inference) about object-level inference, which, trivially, it is, since CDB is a first order extension of DB. Clearly, there is a distinction between NAF and Theorem 1, which was supposed to validate it. NAF itself is nothing but an operational rule, whereas Theorem 1 is a statement about logical inference. The prefix "meta" is therefore redundant, for every statement about object-level inference is a meta-level statement and cannot be anything else. It is also pedantic, because meta-level inference is the common mode of inference in mathematics. There is nothing either extralogical or special about it at all.

In fact, as we've seen, NAF is not valid in terms of inference from CDB, and it is clearly not valid in terms of inference from DB. For example, if DB is

DB_5: $P \leftarrow \neg R$,

then P succeeds but $DB_5 \nvdash P$.

Thus, it is not immediately clear how NAF could be interpreted at all in terms of object-level inference.

NAF is also described [1, p. 296] as a "derived inference rule for deductions from the CDB". The only logical meaning this seems to have is that Theorem 1 is a derived rule of inference from CDB, which is false. The only inference rules for CDB are those of some formal logical calculus, in this case the predicate calculus and, hence, those of any other first order theory.

Thus, Clark's original question remains open. Can NAF, in any sense, be regarded as a valid logical inference? In fact, is there any logical justification for it at all? The answer is yes, as we show below.

II. THE CONSISTENCY OF NAF

In 1949 Kurt Gödel showed that the axiom of choice (AC) is consistent with the other axioms of Zermelo-Fraenkel set theory (ZF), i.e., if ZF is consistent, so is ZF + AC; and in 1963 Paul Cohen showed that the negation of AC is also consistent with ZF: if ZF is consistent, so is ZF + ¬AC. The profound, revolutionary effect that these two results have had on our understanding of the nature of mathematics and deductive reasoning is to show that logic is not only about proving theorems of formal theories, i.e., about proving the existence of formal proofs of given assertions, but it is more fundamentally about proving consistency or, more precisely, relative consistency, i.e., about proving the nonexistence of formal proofs of the negations of given assertions. Whichever way one looks at it, therefore, logic is about exhibiting meta-level proofs: on the one hand, proofs of the existence of object-level proofs, and, on the other hand, proofs of their nonexistence. Roughly speaking, therefore, logic is as much concerned with meta-level inference as it is with object-level inference, and the two are inseparable.

It is therefore not unreasonable to contend, as we do, that, in its broadest sense, logic programming should be concerned with effectively establishing the existence of both proofs of consistency and proofs of theorems of formal theories. It seems, however, that it is commonly conceived only in terms of the latter. The concept of automatic theorem proving thus seems to be heavily overplayed at the expense of the more fundamental notion of consistency.

We now show that NAF can be logically validated by both derivability and consistency—derivability from a consistent extension DB* of the database and, hence, consistency with the database.

For the sake of continuity, omitted proofs are given in the appendix.

Definition. DB* = DB + E, where E is

$$\{\neg L : L \text{ is a closed, positive literal in } \mathscr{L}(\mathrm{DB}) \text{ and } L \text{ fails in QEP}\}.$$

Theorem 7. DB* *is consistent.*

Corollary 7.1. DB *is consistent.*

Corollary 7.2. EDB *is consistent.*

PROOF (Sketch). If L is a predicate symbol of $\mathscr{L}$(DB) such that there is no clause about L in DB, then, for any closed substitution instance L' of L, L' cannot unify with the head of any DB clause. It therefore fails (at height 1 of the failure tree). Thus, DB* is an extension of EDB and EDB is consistent. □

Direct proofs of Corollaries 7.1, 7.2 are given in the appendix, since the methods of proof are of interest.

Theorem 8. If a query Q fails in QEP, *then* $\text{DB}^* \vdash \forall[\neg Q]$.

Hence, by Theorem 7, we have

Corollary 8.1. If Q fails in QEP, *then* $\text{DB} + \forall[\neg Q]$ *is consistent.*

Note that this result was proved by Shepherdson [7, Theorem 9]. A direct, alternative proof to Shepherdson's is given in the appendix.

Theorem 9. If Q succeeds in QEP *with an answer θ, then* $\text{DB}^* \vdash \forall[Q\theta]$.

Again, by Theorem 7, we immediately have

Corollary 9.1. If Q succeeds in QEP *with an answer θ, then* $\text{DB} + \forall[Q\theta]$ *is consistent.*

For example, in DB_5, P succeeds, $\neg P$ fails, and both $\text{DB}_5 + P$ and $\text{DB}_5 + \neg P$ are clearly consistent.

Theorem 10. If L is a closed, positive literal in $\mathscr{L}$(DB), *and if* $\text{DB}^* \vdash \neg L$, *then L fails in* QEP.

Now, recall that different selection rules determine different QEPs [1, p. 301], and that when we say that a query fails (succeeds) in QEP, we mean that it fails (succeeds) under some selection rule. Moreover, a query which fails (succeeds) under one selection rule need not fail (succeed) under another selection rule, but it cannot succeed (fail) under another one [7, Theorem 4]. We therefore introduce the following definitions.

Definitions. Let DB be a database, **R** be a selection rule, and Q be an arbitrary query in $\mathscr{L}$(DB).

(1) Q is **R**-*decidable* if it either fails or succeeds under **R**.

(2) **R** *decides* Q if Q is **R**-decidable.

(3) Q is QEP-*decidable* if it is **R**-decidable for some **R**.

(4) QEP *decides* Q if Q is QEP-decidable.

(5) DB is **R**-*complete*, or **R** is *complete for* DB, if every query in $\mathscr{L}$(DB) is **R**-decidable.

(6) DB is QEP-*complete*, or QEP is *complete for* DB, if every query in $\mathscr{L}$(DB) is QEP-decidable.

REMARK. Instead of simply saying that Q is decidable or that DB is complete, the qualifications "**R**-decidable" etc. are used to prevent confusion with the standard notions of decidability and completeness in mathematical logic. A sentence S is decidable in a theory T if $T \vdash S$ or $T \vdash \neg S$, and T is complete if every sentence S in $\mathscr{L}(T)$ is decidable in T. Here, of course, we are talking about logical derivability, not query evaluation.

The proof of the next theorem is obvious.

Theorem 11. If Q is QEP-*decidable* (**R**-*decidable*), *then, for any closed substitution instance Q' of Q in* $\mathscr{L}$(DB),

(1) DB* $\vdash \neg Q'$ *implies Q fails in* QEP (*under* **R**), *and*

(2) DB* $\vdash Q'$ *implies Q succeeds in* QEP (*under* **R**).

Clearly, then, the following weaker result holds. This is analogous to Clark's remark [1, p. 317] about the "completeness" of NAF.

Corollary 11.1. If DB *is* QEP-*complete* (**R**-*complete*), *then, for every closed substitution instance Q' of Q in* $\mathscr{L}$(DB),

(1) DB* $\vdash \neg Q'$ *implies Q fails in* QEP (*under* **R**), *and*

(2) DB* $\vdash Q'$ *implies Q succeeds in* QEP (*under* **R**).

Notice that, whereas Clark supposed [1, p. 317]—rightly, as it turned out (see [7, Theorem 3])—that QEP-completeness would yield the consistency of CDB and hence what he called the "completeness" of NAF, the consistency of DB* does not depend on the QEP being complete for DB. It is consistent anyway.

Theorems 8 and 9 and their corollaries provide a clear validation of NAF as an effective (and efficient) means of proving theorems of DB* and consistency with DB. Considering the earlier remarks about the nature of logic and the original motivation for CDB, the validation in terms of consistency is fundamental, probably the most natural, and certainly a minimum logical requirement. Notice that it does not presuppose the need to assign any meaning to DB outside itself. In particular, unlike the equality axioms of CDB, the question of the free interpretation of terms in DB simply doesn't arise. Putting this simplistically, then, NAF is a theorem prover for DB* and a consistency prover for DB.

QEP *as a Theorem Prover*

Although DB* is a first order extension of DB, it cannot plausibly be considered a natural interpretation of DB, despite its consistency. We are not assuming that such an interpretation is desirable or even useful, for, as we have seen, it is unnecessary.

However, the reason for the implausibility is that DB* is defined with respect to the infinite totality of all possible selection rules and hence independently of any one of them. One clearly cannot program according to all possible rules, for the simple reason that a query can fail (succeed) under one selection rule but not under another.

Incidentally, since DB* is independent of any one selection rule, one might be led to suppose that it is independent of the logical formulation of DB, but this is not so. It is possible for two logically equivalent DBs to have incompatible DB*s. For example, let P and L be two 0-place predicate symbols, and consider

DB_6: $\quad P \leftarrow \neg L,$

which is equivalent to

DB_7: $\quad L \leftarrow \neg P.$

For any selection rule **R**, L fails in DB_6 and P fails in DB_7. Hence, if $\{P, L\}$ is the vocabulary of both databases, then DB_6^* is $\mathrm{DB}_6 + \{\neg L\}$ and DB_7^* is $\mathrm{DB}_7 + \{\neg P\}$. Thus, DB_6^* and DB_7^* are different theories. Moreover, $\mathrm{DB}_6^* \vdash \neg L$ and $\mathrm{DB}_7^* \vdash L$.

Now, if a database DB is constructed on the basis of a given selection rule **R**, and if an extension DB^+ of DB is the intended meaning of DB, it is reasonable to suppose that the definition of DB^+ should, in some way, depend upon **R**, and hence on the formulation of DB. Thus, logically equivalent DBs could be expected to yield even incompatible interpretations, although these must, of course, be compatible with their respective databases. For example, since only one selection rule is possible for databases DB_6 and DB_7, DB_6^* and DB_7^* can reasonably be considered to be interpretations of DB_6 and DB_7 respectively.

It is therefore only because DB* is defined with respect to all possible selection rules that we hold it to be an unacceptable interpretation of DB.

This does not mean that the logical validation of NAF is weakened in any way. It simply means that QEP cannot yet be regarded as a theorem prover for any useful theory whatsoever, and that the usefulness of QEP is only as a consistency prover for DB.

QEP *as a Consistency Prover*

Roughly speaking, if Q succeeds, we are entitled by Corollaries 8.1 and 9.1 to assume Q and, if it fails, to assume $\neg Q$, which is very different from inferring that Q holds or that $\neg Q$ holds. Thus, the word "infer" in the NAF rule [1, p. 294] should be replaced by "assume", and the rule should be

if Q succeeds with answer θ, assume $\forall[Q\theta]$, and

if Q fails, assume $\forall[\neg Q]$.

Now, for any sentence Q, it is often the case that both $\mathrm{DB} + Q$ and $\mathrm{DB} + \neg Q$ are consistent, so NAF is also a decision rule, viz., a rule for deciding which of Q and $\neg Q$ to assume. This, of course, is useful, but it has certain practical and theoretical limitations, the most obvious of which are as follows.

Logical consistency of the database and with the database are altogether different from the semantic consistency of a given situation. The intended meanings of the various predicates in a database could well, and often do, give rise to contradictions

at the level of intended meaning. Likewise, answers to queries may be incorrect. In particular, just as implicit completeness blindly assumes that unprovable atoms are always false, the decision which NAF makes about which of Q and $\neg Q$ to assume is inflexible and not always the most desirable. This was illustrated above with the database for the ordering relation.

Corollaries 8.1 and 9.1 do not mean that, if $\mathcal{M}$ is a model of DB and Q fails, then $\mathcal{M}$ is a model of $\neg Q$, or that, if Q succeeds, $\mathcal{M}$ is a model of Q. Otherwise, $DB \vdash \neg Q$ or $DB \vdash Q$, respectively. All that is meant is that if Q fails, there is a model $\mathcal{M}'$ of $DB + \neg Q$, and that, if it succeeds, there is a model $\mathcal{M}''$ of $DB + Q$. Thus, a new model is invoked every time a closed negative literal is evaluated by QEP, and the new model is not determined by the old one.

Thus, NAF gives no information about what is logically derivable from DB, only about what is not derivable; for if Q succeeds, then $DB \nvdash \neg Q$ although perhaps $DB \vdash Q$. Likewise, if Q fails, then $DB \nvdash Q$ although perhaps $DB \vdash \neg Q$. In general, therefore, if we want to prove theorems of DB, then DB must not contain negative literals, NAF serves no purpose, and we are obliged to use some other mechanism, such as SLD resolution, which is actually a theorem prover for definite Horn databases.

The main limitation of NAF as a consistency prover is that it is one-sided, for, since a query Q cannot both fail and succeed (even under different selection rules), QEP cannot show that both $DB + Q$ and $DB + \neg Q$ are consistent, when they might well be, and if one of them is consistent, one usually needs to know if the other is too.

Nor is NAF always helpful. If it is regarded as a consistency prover, then the query Q can only mean "Is Q consistent with DB?". If the query succeeds, the answer is yes. If it doesn't, then, even if it fails, we are no better off. All we can then know is that $DB + \neg Q$ is consistent, and nothing about the consistency of $DB + Q$. In this case, success is helpful and failure isn't.

The opposite is also true. Q might be regarded as true in some context, so we might want to include it in the database if it isn't already a theorem. Failure of Q would then be useful, for it would mean that Q isn't a theorem. Success would simply mean that $DB \nvdash \neg Q$, which, if Q is an atom, we know anyway (see appendix).

We might also wish to assume that a sentence Q is true if $DB + Q$ is consistent and, as in the closed world assumption, false if $DB + \neg Q$ is consistent. By Corollaries 8.1 and 9.1, Q could then be assumed true if it succeeds and false if it fails. Of course, the difference between this assumption and the closed world assumption, quite apart from the fact that ours is legitimate and the closed world assumption generally isn't, is that firstly, ours can be made for an arbitrary query, as opposed to only positive closed literals, and secondly, it applies only to one query at a time.

FINAL REMARK

The version of NAF which is described in [1], whereby negative literals must be closed in order to be evaluated, is implemented in MU-PROLOG (Melbourne University PROLOG). Thus, negation in MU-PROLOG is logical negation, and QEP is a theorem prover for DB* and a consistency prover for DB. However, other forms of PROLOG allow open negative literals to be evaluated, and since the proofs

of the theorems in this paper, in particular Theorems 8, 9, all depend heavily on negative literals being closed, it seems that it would be difficult to show that negation in these forms of PROLOG has any logical meaning at all, even the meaning which is commonly ascribed to it, viz., that if Q fails, then Q is not a theorem of DB. This is effectively Corollary 8.1, which has only been proven for the version of NAF in [1].

APPENDIX

This contains the proofs of Theorems 6–10.

First, we need to be clear about notation and the various formal systems of first order logic in which we shall be working. These are denoted by $\varepsilon(\mathscr{V})$, $\mathrm{PC}(\mathscr{V})$, and $\mathrm{EC}(\mathscr{V})$, where, for a given vocabulary $\mathscr{V}$ of function and predicate symbols, $\varepsilon(\mathscr{V})$ is the ε-calculus for $\mathscr{V}$, $\mathrm{PC}(\mathscr{V})$ is the predicate calculus for $\mathscr{V}$, and $\mathrm{EC}(\mathscr{V})$ is the elementary calculus for $\mathscr{V}$. The reader is referred to [2] or [3] for background details, which are necessarily omitted.

The axioms of $\varepsilon(\mathscr{V})$ are all those sentences of $\mathscr{L}(\mathscr{V})$, the first order language of $\mathscr{V}$, which are instances of the following axiom schemata:

P1. $A \to B \to A$.

P2. $(A \to B \to C) \to (A \to B) \to A \to C$.

P3. $(\neg A \to \neg B) \to (B \to A)$.

P4. $(A \to \not{f}) \to \neg A$.

P5. $(A \wedge B) \to A$.

P6. $(A \wedge B) \to B$.

P7. $A \to B \to (A \wedge B)$.

P8. $\neg(A \vee B) \to \neg A$.

P9. $\neg(A \vee B) \to \neg B$.

P10. $\neg A \to \neg B \to \neg(A \vee B)$.

Q1. $\forall x A \to \neg \exists x \neg A$.

Q2. $\neg \forall x A \to \exists x \neg A$.

Q3. $\neg \exists x A \to \neg A(t)$.

Q4. $\exists x A \to A(\varepsilon x A)$.

E1. $s = t \wedge A(x/s) \to A(x/t)$, where A is an atom and x does not occur free in A within the scope of an ε-symbol.

E2. $\forall z(A(x/z) \leftrightarrow B(y/z)) \to \varepsilon x A = \varepsilon y B$.

E3. $t = t$.

The sentence $\not{f}$ is not a predicate symbol of $\mathscr{L}(\mathscr{V})$, but a logical symbol denoting falsity.

The single inference rule of $\varepsilon(\mathscr{V})$ is modus ponens:

$$\frac{A,\ A \to B}{B}.$$

The axioms of PC($\mathscr{V}$) are the ε-free axioms of $\varepsilon(\mathscr{V})$. In addition to modus ponens, PC($\mathscr{V}$) has the following inference rule, called the $\exists$-rule.

$$\frac{A(x/a) \to B}{\exists x A \to B},$$

where a is an individual symbol not appearing in A or B.

The axioms of EC($\mathscr{V}$) are the elementary axioms of $\varepsilon(\mathscr{V})$, i.e., those which are quantifier-free and ε-free. Modus ponens is the single inference rule of EC($\mathscr{V}$).

NOTE. As is usual in logic, we do not assume that the vocabulary of a database DB only consists of the function and predicate symbols which appear in the axioms of DB.

Notation.

(1) If $X \cup \{A\}$ is any set of sentences in $\mathscr{L}(\mathscr{V})$ (sentences have no free variables), we write $X \vdash_{\varepsilon} A$, $X \vdash_{\mathrm{PC}} A$, and $X \vdash_{\mathrm{EC}} A$ to denote that there is a deduction of A from X in $\varepsilon(\mathscr{V})$, PC($\mathscr{V}$), and EC($\mathscr{V}$) respectively. If it is clear which logical calculus is referred to, we simply write $X \vdash A$.

(2) $\mathrm{con}_{\varepsilon}(X)$ means that X is consistent in $\varepsilon(\mathscr{V})$, i.e., that $\not$ is not a theorem of X in $\varepsilon(\mathscr{V})$. Likewise, we write $\mathrm{con}_{\mathrm{PC}}(X)$, $\mathrm{con}_{\mathrm{EC}}(X)$, or just $\mathrm{con}(X)$.

(3) By a *literal* we shall mean an atom $P(t_1, \ldots, t_n)$, where P is a predicate symbol of $\mathscr{V}$, or the negation of such an atom. Note that, like the symbol $\not$, the identity symbol $=$ is not a predicate symbol of $\mathscr{V}$, although every formula $s = t$ is an atom.

Before proving Theorem 6, we need some lemmas, and we begin with a rigorous proof of the well-known result that DB is consistent. In fact, we show that this would hold even if DB were allowed to contain ε-terms and formulae of the form

$$s = t \leftarrow L_1 \wedge \cdots \wedge L_m, \qquad m \geq 0$$

and even if the L_i were allowed to be atomic formulae of the form $s = t$. We shall denote databases in this wider sense by DB^+.

Lemma 1. $\mathrm{con}_{\varepsilon}(\mathrm{DB}^+)$.

PROOF. For any formula A of $\mathscr{L}(\mathscr{V})$, define the formula $g(A)$ by induction on the complexity of A, as follows:

(1) If A is an atom $P(t_1, \ldots, t_n)$, then $g(A)$ is $\neg\not$.

(2) If A has the form $s = t$, then $g(A)$ is $\neg\not$.

(3) If A is $\not$, then $g(A)$ is $\not$.

(4) If A has the form $\neg B$ or $B * C$, where $*$ is $\wedge$, $\vee$, or $\to$, then $g(A)$ is $\neg g(B)$ or $g(B) * g(C)$ respectively.

(5) If A has the form $\forall x B$ or $\exists x B$, then $g(A)$ is $g(B)$.

Thus, $g(A)$ is obtained from A by erasing all terms appearing in A, erasing all quantifiers, and replacing every occurrence of $=$ by $\neg\not$.

It is easy to see that if A is a logical axiom of $\varepsilon(\mathscr{V})$, then $g(A)$ is a tautology, and that if $g(A)$ and $g(A \rightarrow B)$ are tautologies, then $g(B)$ is a tautology. Moreover, if A is (the universal closure of) an axiom

$$R \leftarrow L_1 \wedge \cdots \wedge L_m$$

of DB^+, then $g(A)$ is a tautology of the form

$$\neg f \leftarrow g(L_1) \wedge \cdots \wedge g(L_m).$$

Hence, if $\mathrm{DB}^+ \vdash A$, then $g(A)$ is a tautology. However, $g(f)$ is f, which is not a tautology, so $\mathrm{DB}^+ \nvdash f$. □

NOTE. If DB^+ is ε-free and contains no axioms of the form

$$s = t \quad \leftarrow \quad L_1 \wedge \cdots \wedge L_m,$$

then DB^+ is just DB and, by Hilbert's second ε-theorem (see [2] or [3]), $\mathrm{DB} \vdash_{\varepsilon} f$ implies $\mathrm{DB} \vdash_{\mathrm{PC}} f$. In this case, $g(s = t)$ could also have been defined as follows:

$$g(s = t) \text{ is } \begin{cases} f & \text{if } s \text{ is not } t, \\ \neg f & \text{otherwise.} \end{cases}$$

Corollary 1. By the proof of Lemma 1, *every* DB *and* DB^+ *has a model in which every atom* $P(t_1, \ldots, t_n)$ *and* $s = t$ *is true for all terms* $t_1, \ldots, t_n, s, t$.

Corollary 2. If $L_1, \ldots, L_n$ *are any positive literals, then* $\mathrm{con}(\mathrm{DB} + \forall[L_1 \wedge \cdots \wedge L_n])$, *whether the* L_1 *belong to* $\mathscr{L}(\mathscr{V})$ *or not.*

PROOF. If an L_i is not in $\mathscr{L}(\mathscr{V})$, extend $\mathscr{V}$ by adding the corresponding predicate symbol and augment DB with the universal closures of the L_i to get a new database. □

Corollary 3. If A *is a closed atom other than* f, *then* $\neg A$ *is not a theorem of* DB.

PROOF. If A is a literal, the result is immediate. The only other atoms have the form $s = t$. The proof of Lemma 1 converts these to $\neg f$, so if $\mathrm{DB} \vdash s \neq t$, then $\mathrm{DB} \vdash f$, which is a contradiction. □

Definition 1. A substitution is any function from a finite set of variables into the set $T_{\mathscr{L}}$ of terms of $\mathscr{L}(\mathscr{V})$, which may contain free variables.

Definition 2. Let $\mathscr{V}$ be the vocabulary of DB, and N be the set of predicate symbols P in $\mathscr{V}$ for which there is no axiom in DB "about" P, i.e., no axiom of the form

$$P(t_1, \ldots, t_n) \leftarrow L_1 \wedge \cdots \wedge L_m.$$

Let E be the set

$$\{\forall[\neg P] : P \text{ is in } N\}.$$

Then EDB, as defined above, is $\mathrm{DB} + E$.

PROOF OF THEOREM 6. As in Lemma 1, we show in fact that if EDB^+ is $\mathrm{DB}^+ + E$, then $\mathrm{con}_{\varepsilon}(\mathrm{EDB}^+)$.

Suppose the contrary and let $\mathscr{R} = \langle A_1, \ldots, A_n \rangle$ be a refutation of EDB^+ in $\varepsilon(\mathscr{V})$. Let $'$ denote the operation of replacing $P\theta$ by f throughout $\mathscr{R}$ for every P in N and

every θ. It is easy to see that if A_i is a logical axiom of $\varepsilon(\mathscr{Y})$, then A_i' is either a tautology or a logical axiom of the same form, and that, if A_i follows by modus ponens from A_j and A_k, then A_i' follows by modus ponens from A_j' and A_k'. If A_i is in E, then A_i' is the tautology

$$\neg \not f.$$

If A_i is a nonlogical axiom of DB^+, e.g., the universal closure of

$$R(t_1,\ldots,t_r) \leftarrow L_1 \wedge \cdots \wedge L_m,$$

then, since $R(t_1,\ldots,t_r)$ is not $P\theta$ for any P in N (by the definition of N), A_i' is

$$R(t_1',\ldots,t_r') \leftarrow L_1' \wedge \cdots \wedge L_m'.$$

Consider the L_j, $j = 1,\ldots,m$.

Case (i): L_j is a positive literal. If L_j is P for some P in N, then L_j' is $\not f$ and A_i' is the tautology

$$R(t_1',\ldots,t_r') \leftarrow L_1' \wedge \cdots \wedge L_{j-1}' \wedge \not f \wedge L_{j+1}' \wedge \cdots \wedge L_m'.$$

Otherwise, L_j' is L_j and A_i' is

$$R(t_1',\ldots,t_r') \leftarrow L_1' \wedge \cdots \wedge L_{j-1}' \wedge L_j \wedge L_{j+1}' \wedge \cdots \wedge L_m'.$$

Case (ii): L_j is a negative literal. In this case, L_j' is similarly either $\neg \not f$ or L_j. If L_j' is $\neg \not f$, then A_i' is equivalent in $\varepsilon(\mathscr{Y})$ to

$$R(t_1',\ldots,t_r') \leftarrow L_1' \wedge \cdots \wedge L_{j-1}' \wedge L_{j+1}' \wedge \cdots \wedge L_m'.$$

Case (iii): L_j has the form $s = t$ or $s \neq t$. L_j' is then L_j, since $=$ is not a predicate symbol, and A_i' is

$$R(t_1',\ldots,t_r') \leftarrow L_1' \wedge \cdots \wedge L_{j-1}' \wedge L_j \wedge L_{j+1}' \wedge \cdots \wedge L_m'.$$

Hence, if A_i is an axiom of DB^+, then A_i' is the universal closure of either a tautology [and so equivalent in $\varepsilon(\mathscr{Y})$ to a tautology], or of a formula of the form

$$S \leftarrow G_1 \wedge \cdots \wedge G_r.$$

Let DB_1^+ be the set of formulae so formed by the operation $'$. By augmenting $\mathscr{R}'$ with proofs of the tautologies introduced by $'$, we thus find that

$$\mathrm{DB}_1^+ \vdash_{\varepsilon} \not f,$$

which, by Lemma 1, is impossible. □

Corollary 4. It follows from the above proof that EDB^+, *and hence* DB^+, *has a model in which every atom of the form $s = t$ is true for all s and t, and in which every positive literal $P(t_1,\ldots,t_n)$ is false for all $t_1,\ldots,t_n$ if P is in N, and true for all $t_1,\ldots,t_n$ otherwise.*

The next three lemmas were proved by Shepherdson [7].

Lemma 2. If a query Q fails in QEP *under some selection rule* **R**, *then, for each substitution θ, there is a selection rule $\mathbf{R}_\theta$ under which $Q\theta$ fails.*

Lemma 3. If a query succeeds in QEP (*under some selection rule*), *then it cannot fail under any selection rule.*

Lemma 4. If Q succeeds in QEP *with an answer θ under a selection rule* **R**, *then Qθ succeeds under a rule* $\mathbf{R}_\theta$.

The next lemma follows by an easy induction from Lemma 6 in [7].

Lemma 5. Let Q be a closed query $L_1 \wedge \cdots \wedge L_m$.

(1) *If Q fails under a selection rule* **R**, *then, for some* $i = 1, \ldots, m$, L_i *fails under a selection rule* **R**′.
(2) *If Q succeeds under a selection rule* **R**, *then for each* $i = 1, \ldots, m$, L_i *succeeds under a selection rule* **R**′.

PROOF OF THEOREM 7. Note that Theorem 7 follows trivially from Lemma 8 of [7], where it is shown that the model $\mathcal{M}$, in which closed positive literals are true unless they fail in QEP, is a model of DB. Since [7] omits details of the proof that $\mathcal{M}$ satisfies the axioms of PC($\mathcal{V}$), and since it is instructive to do so, we give a constructive proof here of Theorem 7. Note also that, as it stands in [1], the QEP cannot handle ε-terms or the identity symbol, so if DB contains either of these, then DB* is undefined.

Suppose Theorem 7 is false. Then

$$\mathrm{DB}^* \vdash_{\mathrm{PC}} \not\!f,$$

so by Hilbert's first ε-theorem ([2] or [3]),

$$Y + \{\neg L_i\}_{i \in I} \vdash_{\mathrm{EC}} \not\!f,$$

where $\{\neg L_i\}_{i \in I}$ is some finite subset of E, and each member of Y is a substitution instance of the matrix of some axiom of DB*.

Let $\mathcal{R} = \langle A_1, \ldots, A_n \rangle$ be a refutation in EC($\mathcal{V}$) of

$$Y + \{\neg L_i\}_{i \in I},$$

and let $'$ denote the operation of

(1) replacing by $\not\!f$ every positive closed literal L in $\mathcal{R}$ such that $\neg L$ is in E,
(2) replacing by $\neg\not\!f$ every other positive closed literal in $\mathcal{R}$,
(3) replacing $s = t$ by $\not\!f$ if s is not t and by $\neg\not\!f$ if s is t.

We show by induction on i that, for each $i = 1, \ldots, n$, A_i' is a tautology. We then argue as at the end of the proof of Lemma 1.

If A_i is an axiom of the form P1, ..., P10, then A_i' is clearly an axiom of the same form and hence a tautology. If A_i is an E1-axiom

$$s = t \wedge A(x/s) \rightarrow A(x/t),$$

then A_i' is the tautology

$$\not\!f \wedge A(x/s)' \rightarrow A(x/t)'$$

if s is not t. If s is t, A_i' is the tautology

$$\neg\not\!f \wedge A(x/s)' \rightarrow A(x/s)'.$$

If A_i is an E3-axiom $t = t$, then A_i' is the tautology $\neg\not\!f$. Thus, if A_i is a logical axiom of EC($\mathcal{V}$), then A_i' is a tautology. If A_i is a member of E, then A_i' is $\neg\not\!f$. If

A_i is a member of Y, then it has the form

$$R \leftarrow L_1 \wedge \cdots \wedge L_m,$$

where R, $L_1, \ldots, L_m$ are closed literals. A_i is then a closed substitution instance A of a DB formula

$$A: \qquad R^* \leftarrow L_1^* \wedge \cdots \wedge L_m^*.$$

Suppose A_i' is not a tautology. It must then be

$$\not\leftarrow L_1' \wedge \cdots \wedge L_m', \qquad m \geq 0,$$

where L_j' is equivalent to $\neg\not$ for each $j = 1, \ldots, m$, and R must fail in QEP under some selection rule **R**. Since R is closed, it unifies with R^* under the restriction θ_1 of θ to the set of free variables of R^*. That is, R is $R^*\theta_1$, which is $R^*\theta$. Hence, $R^*\theta_1$ fails under **R**, $m > 0$, and the query

$$Q: \qquad \{L_1^* \wedge \cdots \wedge L_m^*\}\theta_1$$

fails under **R**. Let $\theta = \theta_1\theta_2$. Then, by Lemma 2, $Q\theta_2$ fails under some selection rule **R'**. By Lemma 5, $L_j^*\theta$ fails under some selection rule **R''** for some $j \leq m$. That is, L_j fails for some $j \leq m$. If L_j is a positive literal, then L_j' is $\not$ and A_i' is a tautology, contrary to supposition. L_j must therefore be a negative literal, say $\neg L_j^\circ$. Since L_j fails, L_j° is closed and it succeeds. Hence, by Lemma 3, L_j° cannot fail (under any selection rule). Since L_j° is closed, $(L_j^\circ)'$ is $\neg\not$ and L_j' is $\neg\neg\not$, which is a contradiction. Thus, if A_i is a member of Y, then A_i' is a tautology.

If A_i follows by modus ponens from A_j and A_k, then $j, k < i$, so, by the induction hypothesis, A_j' and A_k' are tautologies. Clearly, A_i' follows by modus ponens from A_j' and A_k', so A_i' is also a tautology. Hence, for each $i = 1, \ldots, m$, A_i' is a tautology. However, $\not'$ is $\not$, which is not a tautology, so A_n is not $\not$, which contradicts the original supposition. □

NOTE. The above proof permits formulae L_i in the antecedents of database axioms to be atoms of the form $s = t$ so long as QEP is extended in such a way that, for any terms s, t (not necessarily closed), the following conditions hold:

(1) If $s = t$ fails, then $\{s = t\}\theta$ fails for every θ.

(2) If s, t are different constant terms (with no free variables), then $s = t$ fails.

(3) $t = t$ succeeds.

PROOF OF THEOREM 8. Suppose Q is

$$L_1 \wedge \cdots \wedge L_n$$

and that it fails in QEP. Let $Q\theta$ be an arbitrary substitution instance of Q. By Lemma 2, $Q\theta$ fails. By Lemma 5, $L_i\theta$ fails for some $i \leq n$, so, by the definition of DB*, $\mathrm{DB}^* \vdash \neg L_i\theta$. Hence, $\mathrm{DB}^* \vdash \neg Q\theta$ for every θ, and $\mathrm{DB}^* \vdash \forall[\neg Q]$, as required. □

PROOF OF THEOREM 9. Suppose Q succeeds with answer θ. We show by induction on the finite height $h \geq 0$ of the success tree of Q that $\mathrm{DB}^* \vdash_{\mathrm{PC}} \forall[Q\theta]$.

If $h = 0$, there is nothing to prove, since Q is empty and the result is tautologous. Suppose that $h > 0$, that Q is

$$L_1 \wedge \cdots \wedge L_m, \qquad m \geq 1,$$

and that the selection rule **R**, say, first selects L_i.

Case (i): L_i is a positive literal. Then L_i is not necessarily closed, and the first step in the evaluation of L_i finds a unifying substitution θ_1, which unifies L_i with the head of a DB clause

$$R(t_1, \ldots, t_n) \leftarrow L'_1 \wedge \cdots \wedge L'_k$$

and replaces Q with $Q'\theta_1$, where $Q'\theta_1$ is

$$\{L_1 \wedge \cdots \wedge L_{i-1} \wedge L'_1 \wedge \cdots \wedge L'_k \wedge L_{i+1} \wedge \cdots \wedge L_m\}\theta_1.$$

Since $Q'\theta_1$ succeeds, there is a sequence of unifying substitutions $\theta_1, \ldots, \theta_r$ such that $\theta = \theta_1 \cdots \theta_r$. Since the height of the success tree of $Q'\theta_1$ is less than h and $Q'\theta_1$ succeeds with answer $\theta' = \theta_2 \cdots \theta_r$, it follows by the induction hypothesis that

$$\mathrm{DB}^* \vdash_{\mathrm{PC}} \forall[Q'\theta_1\theta'],$$

that is,

$$\mathrm{DB}^* \vdash_{\mathrm{PC}} \forall[Q'\theta].$$

Now clearly

$$\mathrm{DB} \vdash_{\mathrm{PC}} \forall[Q'\theta_1 \rightarrow R\theta_1].$$

Since $R\theta_1$ is $L_i\theta_1$, it follows that

$$\mathrm{DB} \vdash_{\mathrm{PC}} \forall[Q'\theta_1 \rightarrow L_i\theta_1];$$

so

$$\mathrm{DB} \vdash_{\mathrm{PC}} \forall[Q'\theta \rightarrow L_i\theta]$$

and

$$\mathrm{DB}^* \vdash_{\mathrm{PC}} \forall[Q\theta].$$

Case (ii): L_i is a negative literal, say $\neg P$. Since Q succeeds, P is closed and fails. Incidentally, its closed terms are not inherited under unification from any other literals in Q, since L_i is the first literal in Q to be selected by **R**. Since P fails, Q is replaced by

$$Q'': \quad L_1 \wedge \cdots \wedge L_{i-1} \wedge L_{i+1} \wedge \cdots \wedge L_m.$$

Since Q'' succeeds with answer θ, and the height of its success tree is less than h, it follows by the induction hypothesis that

$$\mathrm{DB}^* \vdash_{\mathrm{PC}} \forall[Q''\theta].$$

Now, by the definition of DB*, $\mathrm{DB}^* \vdash_{\mathrm{PC}} L_i$, since P fails, and

$$\varnothing \vdash_{\mathrm{PC}} \forall[Q\theta \leftrightarrow L_i \wedge Q''\theta],$$

since L_i is closed; so

$$\mathrm{DB}^* \vdash_{\mathrm{PC}} \forall[Q\theta],$$

as required. □

PROOF OF THEOREM 10. Let $\mathcal{M}$ be a structure for DB in which a positive closed literal L holds unless it fails under some selection rule. By Lemma 8 of [7], $\mathcal{M}$ is a model of DB, so it is clearly a model of DB*. Hence, if $\mathrm{DB}^* \vdash \neg L$, then $\mathcal{M} \models \neg L$, so L fails in QEP. □

REFERENCES

1. Clark, K. L., Negation as Failure, in: H. Gallaire and J. Minker (eds.), *Logic and Data Bases*, Plenum, New York, 1978.
2. Flannagan, T. B., On an Extension of Hilbert's Second ε-Theorem, *J. Symbolic Logic* 40(3): 393–397 (1975).
3. Leisenring, A. C., *Mathematical Logic and Hilbert's ε-Symbol*, MacDonald, London, 1969.
4. Jaffar, J., Lassez, J.-L., and Lloyd, J. W., Completeness of the Negation as Failure Rule, IJCAI-83, Karlsruhe, 1983, pp. 500–506.
5. Lloyd, J. W., *Foundations of Logic Programming*, Symbolic Computation Series, Springer, New York, 1984.
6. Reiter, R., On Closed World Data Bases, in: H. Gallaire and J. Minker (eds.), *Logic and Data Bases*, Plenum, New York, 1978, pp. 55–76.
7. Shepherdson, J. C., Negation as Failure: A Comparison of Clark's Completed Data Base and Reiter's Closed World Assumption, *J. Logic Programming* (1984).
8. Shepherdson, J. C., Negation as Failure, presented at Alvey Inference Workshop, Imperial College of Science and Technology, Univ. of London, Sept. 1984.