Default Databases and Incomplete Information

P. KING AND C. SMALL†
Department of Computer Studies, Birkbeck College, University of London, Malet Street, London WC1E 7HX

We present the concept of a default database which comprises a set of facts, a set of deduction rules, and a set of defaults. Defaults define assumptions to be made about information not derivable from the rules and facts. Defaults, by augmenting the information which is a consequence solely of the rules and facts, enable definite responses to queries on the basis of 'common sense' assumptions rather than responses of the form 'unknown'. The augmented information is termed an extension. Although such an extension is self-consistent, in general two or more mutually inconsistent extensions can arise. We characterise the rules and defaults of a database as safe if only one extension can arise for any given set of facts. We give conditions which are necessary and sufficient to ensure safety.

Received July 1989, revised November 1990

INTRODUCTION

We take the view that a database comprises information about various entities, their attributes, relationships among them, and constraints and rules reflecting the real world situation of which the database is, in some sense, a model. Such a database can be viewed or formulated in the conventions and notations of logic where it is useful. We adopt this approach in the present work, regarding a database as comprising a collection of clausal form formulae, and note that the conventional database models can be so regarded.[12]

With the relational model, for example, the intension of each relation is seen as an n-ary predicate with the extension being a set of n-tuples for each of which the predicate is true. For each such relation either the Closed World Assumption or the Open World Assumption can be made.* The former assumes that all n-tuples for which the predicate is true are present in the extension, and thus for any other n-tuple the predicate is false. The Open World Assumption allows there to be further n-tuples not in the extension satisfying the predicate, and thus for any other n-tuple the truth or falsity of the predicate is unknown. These two assumptions are commonly made in database practice, although often implicitly in terms of the context and embedded in the functionality of the application programs.

We can adapt the default reasoning discussed by Reiter[10] to the database environment in order to provide a mechanism for making assumptions about data which is neither recorded nor inferable using the rules, based upon what is usual, or on statistical considerations, weighted according to the penalties incurred should the assumptions be wrong. The defaults will be explicit and specified at schema or subschema level and not embedded in program code as hitherto. Adapting Reiter's notation we express an assumption as a default in the form:

$$A(X) \leftarrow P(X) \,\&\, MC(X)$$

where $A(X)$ is the assumption to be made about X, $P(X)$ is the prerequisite which must be true for the assumption to be made, and $MC(X)$ expresses the consistency requirement. The notation may colloquially be read: 'if $P(X)$ is true then in the absence of information to the contrary assume $A(X)$', the information to the contrary being that $C(X)$ is false. For example, we can express the assumptions that soldiers are normally male, and persons aged over 65 are retired, with the defaults:

$$Male(X) \leftarrow Soldier(X) \;\&\; M\ Male(X)$$

$$Retired(X) \leftarrow AgeOver65(X) \;\&\; M\ Retired(X)$$

The remainder of this paper is concerned only with cases like these where $A(X)$ and $C(X)$ are identical, which is thought to cover the majority of practical cases; consequently we omit the consistency component when writing defaults. Persuasive arguments that most defaults are of this form are given by Reiter,[10] although the need for the more general form has been demonstrated by several authors.[1, 11, 14]

It is possible to write defaults which, in some circumstances, are mutually inconsistent. Thus, if we allow more than one profession for an individual the two defaults:

$$Male(X) \leftarrow Soldier(X)$$

$$Female(X) \leftarrow Nurse(X)$$

are clearly inconsistent for a person who is both a soldier and a nurse (because $Female(X) \equiv \neg Male(X)$). If the defaults are activated sequentially rather than simultaneously then the activation of the first would cause Male to be stored which would then prevent the activation of the second; and vice versa.

This paper addresses the problem of such inconsistencies and gives necessary and sufficient conditions which the schema must satisfy to avoid such conflicts whatever the set of stored facts. A schema satisfying these conditions is termed safe. Safety in this sense is a property of the schema alone, which is important since the set of stored facts is normally much larger and more volatile than the schema information. A detailed description of an earlier version of an implemented system is given by Small,[13] which included the detection of conflicting defaults. The method used in discussed briefly in Section 4 below, and is currently the subject of further investigation.

In Section 1 we give a definition of a default database using function free clausal form formulae. In Section 2 we define the mechanism by which the defaults are used to give less incomplete extensions, and in Section 3 we formalise the concept of a safe default database. We conclude by discussing some related work in Section 5.

* Note that our use of these terms does not follow Reiter's definitions.[9]

† Author for correspondence.

1. THE CLAUSAL FORM REPRESENTATION OF A DEFAULT DATABASE

In this representation a term is either a constant or a variable. An atom consists of an n-ary predicate followed by n terms. We use upper case letters for variables, strings starting with lower case letters as constants, and place the terms of an atom within round brackets. A literal is either an atom or its negation. A clause is a disjunction of literals; the variables of a clause are implicitly universally quantified. A clause consisting of a single literal is termed a unit clause.

A schema consists of a set of rules and a set of defaults. A default database consists of a schema and a set of facts. A fact is a ground unit clause. A rule is a non-ground or non-unit clause. Until Section 4 we assume the set of rules to be deductively closed. A default is a formula of the form:

$$A \leftarrow P$$

where A is a single literal and P is a conjunction of literals. P are termed the prerequisite literals and A the assumption literal. We read such a default as stating that if, for some instantiation of the variables of the default, P is true and A is not inconsistent with the database then assume A.

A set of substitutions is a set of ordered pairs:

$$\{V_1/T_1, V_2/T_2, \ldots, V_n/T_n\}$$

where the V_i are distinct variables and the T_i are terms. The result of applying a set of substitutions to a clause or default is a new clause (default) in which each V_i is replaced by the corresponding T_i. We denote the result of the application of a set of substitutions, S, to a clause or default C as $C^{(S)}$. An expression (i.e. a term, atom, literal, clause or default) is said to be ground if it contains no variables. Under the assumption that there are only a finite number of facts, rules and defaults, each of which contains only finitely many constants, we observe that only a finite number of ground expressions can be formed.

Throughout the paper we assume that all variables given in the rules and defaults are distinctly named; i.e. 'standardised apart' in the terminology of resolution theorem proving.[8]

2. A COMPLETED EXTENSION OF A DEFAULT DATABASE

Consider a default database with a consistent set of facts and rules. Let:

$$E_0, E_1, \ldots$$

be a series of sets of clauses such that:

1. E_0 consists of the facts and rules;
2. For all i, $i > 0$, let D_i be the set of candidate defaults which are eligible to be 'fired' given E_{i-1}. That is D_i consists of all ground defaults:

$$(A \leftarrow P)^{(S)}$$

such that:

- $P^{(S)}$ is implied by E_{i-1}; and
- neither $A^{(S)}$ nor $\neg A^{(S)}$ is implied by E_{i-1};

If D_i is empty then E_i consists of E_{i-1}, otherwise E_i consists of E_{i-1} together with a ground unit clause comprising the assumption literal of a non-deterministically selected default of D_i;

E_j is a completed extension of the default database if D_j is empty. Two completed extensions, E_j and $E'_{j'}$, are said to be equivalent if they have the same deductive closure, and are said to be distinct otherwise.

THEOREM 1. *A completed extension, E_j, of a default database is consistent if and only if E_0, i.e. the facts and rules, is consistent.*

Proof. The proof is in two parts.

Firstly, if E_0 is inconsistent then E_j is inconsistent, since by construction it contains E_0 as a subset.

Secondly we show that if E_0 is consistent then E_j is consistent. Proof is by induction. For the base case we have that E_0 is consistent. Assume that for all i, $0 \leqslant i < k$, E_i is consistent. Let the unit clause added to E_{k-1} to form E_k be A. $E_{k-1} \cup A$ is inconsistent if and only if $\neg A$ is implied by E_{k-1}. However, by definition E_{k-1} does not imply $\neg A$. Hence $E_{k-1} \cup A$, i.e. E_k, is consistent. □

Example 1. Consider the following default database:

F1 *Reads(mikhail,pravda)*
R1 $\neg$*Capitalist*(*X*) v$\neg$*Socialist*(*X*)
D1 *Capitalist*(*X*) $\leftarrow$ *Reads*(*X*,*wall_street_journal*)
D2 *Socialist*(*X*) $\leftarrow$ *Reads*(*X*,*pravda*)

In this case there is only one completed extension which contains the following two additional clauses:

Socialist(mikhail)
$\neg$*Capitalist(mikhail)*

3. SAFE DEFAULT DATABASES

A completed extension is a consistent set of clauses derived from the facts, rules and defaults in the way described above. Note that the definition allows a default database to have more than one distinct completed extension. For example, consider the addition of the following fact to the default database of Example 1:

F2 *Reads(mikhail,wall_street_journal)*

It is now possible to generate a second completed extension containing:

$\neg$*Socialist(mikhail)*
Capitalist(mikhail)

If there are two distinct completed extensions of a default database then their union is inconsistent:

LEMMA 1. *If the facts and rules of a default database are inconsistent, then the database has only one completed extension.*

Proof. Every clause is implied by an inconsistent set of clauses.[5] □

LEMMA 2. *If a default database has two distinct completed extensions, E and E', then $E \not\subset E'$ and $E' \not\subset E$.*

Proof. Firstly we observe that if a default database has two distinct completed extensions then the facts and rules are consistent (by lemma 1), and hence both extensions are consistent (by Theorem 1). Assume a default database has two distinct completed extensions E_j and $E'_{j'}$:

$$E_j: E_0 \cup A_1 \cup \ldots \cup A_{j-1}$$
$$E'_{j'}: E_0 \cup A'_1 \cup \ldots \cup A'_{j'-1}$$

where A_i (respectively, A'_i) is the assumption added at step i in the construction of E_j (respectively, $E'_{j'}$). Suppose, without loss of generality, that $E_j \subset E'_{j'}$, and let i be the least i such that A'_i is not implied by E_j. Then A'_i results from a default:

$$A'_i \leftarrow P'_i$$

Since P'_i is implied by E_j (by the minimality of i) it must be the case that $\neg A'_i$ is implied by E_j (for otherwise E_j is not a completed extension). However, $E'_{j'}$ is consistent and so does not imply $\neg A'_i$, contradicting the assumption that $E_j \subset E'_{j'}$. □

THEOREM 2. *If a default database has two distinct completed extensions, E and E', then $E \cup E'$ is inconsistent.*

Proof. Assume a default database has two distinct completed extensions, E_j and $E'_{j'}$:

$$E_j : E_0 \cup A_1 \cup \ldots \cup A_{j-1}$$

$$E'_{j'} : E_0 \cup A'_1 \cup \ldots \cup A'_{j'-1}$$

Let i, $0 < i < j'$, be the least i such that E_j does not imply A'_i (the existence of such an A'_i is guaranteed by Lemma 2). Then A'_i results from a default:

$$A'_i \leftarrow P'_i$$

E_j implies P'_i (by the minimality of i), and hence implies $\neg A'_i$ (for otherwise E_j is not a completed extension). Thus $E_j \cup E'_{j'}$ implies both A'_i and $\neg A'_i$, and hence is inconsistent. □

In many practical cases the user will require a default database to give rise to only one completed extension, and will believe the database to be in error if it supports contradictory extensions. Consequently we characterise a schema as safe if the rules and defaults give only one completed extension for any given set of facts.

Our examination of the database proceeds in two phases. In the first phase 'connections' are identified: a connection may be thought of as a path through the rules from the assumption literal of one default to the assumption literal of another. Then, in the second phase, the connections are tested against the rules to ascertain whether the firing of one default may prevent the firing of another.

Definition. A connection exists between the assumption literals, A and A', of two defaults if there exists a set of substitutions, S, (called a connecting set of substitutions) such that:

1. $A^{(S)}$ *is* $\neg A'^{(S)}$ (in this case A and A' are said to be directly connected); or
2. there is a rule (called a connecting rule) which, after making the substitutions S, contains $\neg A^{(S)}$ and $\neg A'^{(S)}$ (in this case A and A' are said to be indirectly connected).

A pair of defaults is said to be connected if there is a connection between their assumption literals. A default is said to be self-connected if a copy can be made of that rule (with the variables differently named), and the default and the copy are indirectly connected as defined above.

Example 2.1. An example of an indirectly connected pair of defaults:

R1 $\neg$*Reads*(W,*wall_st_journal*) v $\neg$*Reads*(W,*pravda*)
R2 $\neg$*Socialist*(X) v $\neg$*Capitalist*(X)
D1 *Capitalist*(Y) $\leftarrow$ *Reads*(Y,*wall_street_journal*)
D2 *Socialist*(Z) $\leftarrow$ *Reads*(Z,*pravda*)

In this instance D1 and D2 are connected via R2 with the connecting set of substitutions $\{Y/X, Z/X\}$. (The purpose of R1 will become clear in the sequel.)

Example 2.2. An example of a self-connected default (D1 is indirectly connected to a copy of itself, D1′):

R1 $\neg$*Citizen*(P,$C1$) v $\neg$*Citizen*(P,$C2$) v $=$($C1$,$C2$)
D1 *Citizen*(Q,*usa*) $\leftarrow$ *Resident*(Q,*usa*)
D1′ *Citizen*(R,*usa*) $\leftarrow$ *Resident*(R,*usa*)

In this instance the connecting set of substitutions is $\{Q/P, R/P, C1/usa, C2/usa\}$.

Example 2.3. An example of a self-connected default (D1 is indirectly connected to a copy of itself, D1′):

R1 $\neg$*Citizen*(P,$C1$) v $\neg$*Citizen*(P,$C2$) v $=$($C1$,$C2$)
D1 *Citizen*(Q,$C1$) $\leftarrow$ *Resident*(Q,$C1$)
D1′ *Citizen*(R,$C2$) $\leftarrow$ *Resident*(R,$C2$)

In this instance the connecting set of substitutions is $\{Q/P, R/P\}$.

Definition. A directly connected pair of defaults:

$$A \leftarrow P$$

$$\neg A \leftarrow P'$$

is said to be safe if for all connecting sets of substitutions, S, one of the following formulae (in which $\hat{x}$ denote the free variables of the defaults after applying S):

$$\forall \hat{x} \neg (A \,\&\, P \,\&\, P')$$

$$\forall \hat{x} \neg (\neg A \,\&\, P \,\&\, P')$$

is implied by the set of rules. In other words, the defaults are safe if the rules prevent both from being considered simultaneously for firing.

Definition. An indirectly connected pair of defaults:

$$A \leftarrow P$$

$$A' \leftarrow P'$$

is said to be safe if for every connecting rule:

$$\neg A \vee \neg A' \vee L_0 \vee \ldots \vee L_n$$

and connecting set of substitutions, S, one of the following formulae (in which $\hat{x}$ are the free variables of the defaults and the connecting rule, after applying S):

$$\forall \hat{x} \neg (A \,\&\, P \,\&\, P') \vee L_0 \vee \ldots \vee L_n$$

$$\forall \hat{x} \neg (A' \,\&\, P \,\&\, P') \vee L_0 \vee \ldots \vee L_n$$

is implied by the set of rules. In other words, the defaults are safe if one of the following conditions is satisfied:

(i) both defaults cannot be considered simultaneously for firing; or
(ii) the connecting rule does not enable the inference of $\neg A'$ from A, or does not enable the inference of $\neg A$ from A'.

Example 3.1. Consider again the schema of Example 2.1. Defaults D1 and D2 are indirectly connected via R2. However, they are safe because R1 implies:

$$\forall W \neg (\textit{Capitalist}(W) \,\&\, \textit{Reads}(W,\textit{wall\_street\_journal}) \,\&\, \textit{Reads}(W,\textit{pravda}))$$

Example 3.2. Consider again the scheme of Example 2.2, in which D1 is indirectly connected to a copy of D1. Given that the rules also include, implicitly, the usual definitions for the equality predicate, the formula:

$$\forall P \neg(Citizen(P,usa) \,\&\, Resident(P,usa) \,\&\, Resident(P,usa)) \vee = (usa,usa)$$

is implied by the rules, and hence the database is safe.

Example 3.3. Consider again the schema of Example 2.3, in which D1 is indirectly connected to a copy of D1. Since neither of the following formulae are implied by the rules the schema is unsafe:

$$\forall P \forall C1 \forall C2 \neg(Citizen(P,C1) \,\&\, Resident(P,C1) \,\&\, Resident(P,C2)) \vee = (C1,C2)$$

$$\forall P \forall C1 \forall C2 \neg(Citizen(P,C2) \,\&\, Resident(P,C1) \,\&\, Resident(P,C2)) \vee = (C1,C2)$$

Definition. A schema is safe if all connected pairs of defaults are safe.

We are now in a position to state that our criteria are both sufficient and necessary to guarantee that a safe schema supports only one completed extension regardless of the set of facts:

LEMMA 3. *A default database has two or more distinct extensions if and only if at some step, i, in the construction of each extension E_j there is a default in the set of candidate defaults, D_i:*

$$A \leftarrow P$$

such that E_i implies $\neg A$.

Proof. Let Δ be a default database, and let E_0 be the facts and rules of Δ. If E_0 is inconsistent then any clause is provable from E_0,[5] therefore D_1 contains no defaults, and hence Δ has only one extension. Thus our lemma is true for the case in which E_0 is inconsistent. We may therefore assume that E_0 is consistent, and by Theorem 1 that any extension of Δ is consistent.

The proof of our lemma for the case in which E_0 is consistent is in two parts.

Part 1: If. Let E_j be a completed extension of a default database Δ constructed in terms of the series:

$$E_0, E_1, \ldots, E_{i-1}, E_i, \ldots, E_j$$

Let A_i be the unit clause added to E_{i-1} to give E_i and let D_i contain a default:

$$A_i' \leftarrow P_i'$$

such that E_i implies $\neg A_i'$. Then an extension $E_{j'}'$ can be constructed in terms of the series:

$$E_0, E_1, \ldots, E_{i-1}, E_i', \ldots, E_{j'}'$$

such that E_i and E_i' differ only with respect to A_i and A_i'. $E_{j'}'$ does not imply A_i (for otherwise it is inconsistent), and hence E_j and $E_{j'}'$ are distinct.

Part 2: Only if. Let E_j be an extension of a default database Δ. At each step, i, in the construction of E_j, if there is a default:

$$(1)\ A \leftarrow P$$

in D_i and E_i does not imply $\neg A$, then either E_i implies A or (1) is also in D_{i+1}. Thus if there is no step i such that D_i contains:

$$A \leftarrow P$$

and E_i implies $\neg A$, then E_j implies the assumption literal of every default considered for firing at each stage in its construction. Consequently Δ does not have a second extension, $E_{j'}'$, for otherwise $E_{j'}'$ is properly contained in E_j, contradicting Lemma 2. □

THEOREM 3. *A default database with a safe schema has only one distinct completed extension.*

Proof. Let Δ be a default database with a safe schema and with completed extension E_j. By Lemma 3 we need only show that at each step i, $0 < i < j$, there is no default:

D1 $A' \leftarrow P'$

in D_i such that $\neg A'$ is implied by E_i. Proof is by contradiction. Let i be the least i such that for some such default $\neg A'$ is implied by E_i, and let the default fired at step i be:

D2 $A \leftarrow P$

Note that E_{i-1} must imply $(P \,\&\, P')$. We consider two cases:

Case 1. D1 is directly connected to D2. In this case D1 is:

$$\neg A \leftarrow P'$$

As the schema of Δ is safe the rules imply one of:

$$\neg(A \,\&\, P \,\&\, P')$$

$$\neg(\neg A \,\&\, P \,\&\, P')$$

Thus, E_{i-1} must imply either A or $\neg A$, contradicting the assumption that both D1 and D2 are in D_i.

Case 2. D1 is indirectly connected to D2. In this case there is a rule:

$$(1)\ \neg A \vee \neg A' \vee L_0 \vee \ldots \vee L_n$$

such that $(\neg L_0 \,\&\, \ldots \,\&\, \neg L_n)$ is implied by E_i. Since the scheme of Δ is safe the rules imply one of:

$$\neg(A \,\&\, P \,\&\, P') \vee L_0 \vee \ldots \vee L_n$$

$$\neg(A' \,\&\, P \,\&\, P') \vee L_0 \vee \ldots \vee L_n$$

and hence E_{i-1} cannot imply $(\neg L_0 \,\&\, \ldots \,\&\, \neg L_n)$ for otherwise it implies either $\neg A$ or $\neg A'$, contradicting the assumption that both D1 and D2 are in D_i. Hence, let E_{i-1} imply $(\neg L_0 \,\&\, \ldots \,\&\, \neg L_k)$, $k < n$, but neither $\neg L_{k+1}$ nor $\ldots$ nor $\neg L_n$. Since E_{i-1} and E_i differ only with respect to A, there must be a rule:

$$(2)\ \neg A \vee (\neg L_{k+1} \,\&\, \ldots \,\&\, \neg L_n)$$

and since the set of rules is deductively closed it contains (from (1) and (2)):

$$\neg A \vee \neg A' \vee L_0 \vee \ldots \vee L_k$$

indirectly connecting D1 and D2. Furthermore, since the schema of Δ is safe the rules must also imply one of:

$$\neg(A \,\&\, P \,\&\, P') \vee L_0 \vee \ldots \vee L_k$$

$$\neg(A' \,\&\, P \,\&\, P') \vee L_0 \vee \ldots \vee L_k$$

But E_{i-1} implies $(P \,\&\, P' \,\&\, \neg L_0 \,\&\, \ldots \,\&\, \neg L_k)$ and hence either $\neg A$ or $\neg A'$, contradicting the assumption that both D1 and D2 are in D_i. □

THEOREM 4. *For every unsafe schema there is a default database with that schema having two or more distinct completed extensions.*

Proof. Let S be an unsafe schema. We prove, by construction of a set of facts, F, that the database with

schema S and facts F has two extensions. Let D1 and D2 be a pair of unsafe defaults of S. We must consider two cases:

Case 1. D1 and D2 are directly connected. In this case the defaults are of the form:

D1 $A \leftarrow P$
D2 $\neg A \leftarrow P'$

Since the defaults are unsafe the rules of S imply neither of:

$$(1)\ \neg(A \,\&\, P \,\&\, P')$$

$$(2)\ \neg(\neg A \,\&\, P \,\&\, P')$$

Consider now the database comprising schema, S, and facts:

F: $\{P, P'\}$

Then the facts and rules (i.e. E_0) imply neither A nor $\neg A$. Moreover, since E_0 implies P and does not imply $\neg A$, D1 enters A into some extension, say E. For similar reasons there is also an extension, say E', containing $\neg A$. Hence the database with schema S and facts F has two distinct extensions, E and E'.

Case 2. D1 and D2 are indirectly connected. In this case the defaults are of the form:

D1 $A \leftarrow P$
D2 $A' \leftarrow P'$

and there is a connecting rule of the form:

$$\neg A \vee \neg A' \vee L_0 \vee \ldots \vee L_n$$

such that the rules imply neither of the following formulae:

$$(1)\ \neg(A \,\&\, P \,\&\, P' \,\&\, \neg L_0 \,\&\, \ldots \,\&\, \neg L_n)^*$$

$$(2)\ \neg(A' \,\&\, P \,\&\, P' \,\&\, \neg L_0 \,\&\, \ldots \,\&\, \neg L_n)$$

Consider now the database comprising schema S and facts:

F: $\{P, P', \neg L_0, \ldots, \neg L_n\}$

Then the facts and rules (i.e. E_0) imply neither $\neg A$ nor $\neg A'$. Moreover, since E_0 implies P and does not imply $\neg A$, some extension, say E, contains A and (as a consequence of the connecting rule) also $\neg A'$. For similar reasons there is also an extension, say E', containing A' and $\neg A$. Hence the database with schema S and facts F has two distinct extensions, E and E'. □

Example 3.4. We saw, in Example 3.3, that the schema of Example 2.3 is unsafe. Consider the facts:

$$Resident(\alpha, usa)$$

$$Resident(\alpha, ussr)$$

A database with the schema of Example 2.3 and the above facts gives rise to two distinct completed extensions, one of which contains $Citizen(\alpha, usa)$ and the other $\neg Citizen(\alpha, ussr)$.

4. DERIVATION OF CONNECTIONS

So far we have assumed, for the purpose of finding connections between pairs of defaults, that the set of rules is deductively closed. Clearly, however, this is impractical since the deductive closure will in general be infinite. We now relax this assumption and use linear resolution[4] to determine the connections between defaults. We henceforward regard a set of literals as synonymous with a clause. For example:

$$\{\neg Socialist(john), \neg Capitalist(john)\} = \neg Socialist(john) \vee \neg Capitalist(john)$$

If C_1 and C_2 are two clauses, $L_1 \in C_1, L_2 \in C_2$ and $L_1 = \neg L_2$ then the clause:

$$(C_1 - \{L_1\}) \cup (C_2 - \{L_2\})$$

is a resolvent of C_1 and C_2. Given a set R of clauses and a clause $C_0 \in$ R, a series of clauses:

$$C_0, C_1, \ldots, C_n$$

is a linear deduction of C_n from R with top clause C_0 if for all i, $0 < i \leqslant n$, C_i is a resolvent of C_{i-1} and either a clause from R or a clause $C_j, 0 \leqslant j < i$. A refutation of a set of clauses R is a deduction of the empty clause, { }, from R. We can now reformulate our definition of a connection to obviate our requirement that the set of rules be deductively closed. Given a default database with rules R, two defaults:

$$A \leftarrow P$$

$$A' \leftarrow P'$$

are connected if there is a deduction of $\{\neg A', L_0, \ldots, L_m\}$ from R $\cup \{A\}$, in which case the connecting rule is $\neg A \vee \neg A' \vee L_0 \vee \ldots \vee L_m$. Finally, to test the connection for safety we must show that one of the following formulae is implied by R:

$$\neg(A \,\&\, P \,\&\, P') \vee L_0 \vee \ldots \vee L_m$$

$$\neg(A' \,\&\, P \,\&\, P') \vee L_0 \vee \ldots \vee L_m$$

This may be undertaken simply by taking each formula in turn, converting the negation of the formula into a set of clauses, say C, and showing that there is a refutation of R $\cup$ C. If there is such a refutation, the original formula is implied by R; otherwise the defaults are unsafe. A more efficient connection-finding procedure will be the subject of a forthcoming paper.

5. RELATED WORK

Research into the interaction of defaults has also been undertaken by Reiter and Criscuolo,[11] who consider several cases in which defaults give rise to different extensions, whereas intuitively one would expect one default to override the others. The authors show how this objective can be achieved, either by introducing additional defaults, or by re-expressing the assumptions with 'semi-normal' defaults, and present a classification of such interacting defaults. However, the classification is incomplete, in the sense that the interaction of some defaults may not be detected, and some defaults may be classified as interacting when this is not the case.

Throughout the paper we have assumed that unsafe defaults are to be rejected, since we believe that for many large practical database systems the user will consider the database to be in error if more than one extension arises. There are, however, two alternative approaches. The first, proposed by Reiter,[10] allows the more than one extension to arise and treats a query as a request to

* Note that this is logically equivalent to $\neg(A \,\&\, P \,\&\, P') \vee L_0 \vee \ldots \vee L_n$.

ascertain whether there exists an extension implies that query. The second approach is to 'prioritise' the defaults; for example, Touretzky,[14] within the framework of inheritance networks, uses a topological relation called 'inferential distance' to prefer defaults associated with sub-classes to those associated with super-classes, and Etherington[1] shows how semi-normal defaults may be used to give some defaults priority over others. There would appear to be no reason why one or other of these approaches could not be adopted within the framework outlined above; nevertheless, we would argue that even should such an approach be adopted it is still desirable that potential conflicts be drawn to the users attention, since an unsafe pair of defaults would tend to suggest that conclusions may be drawn too hastily.

Minker,[6] and more recently Gelfond and Pryzmusinska,[2] have developed work on the Generalised Closed World Assumption. There are two major differences between our work and the GCWA. Firstly, the GCWA affects closure by reference to the predicate symbols only, whilst our work allows the DBA to select subsets of the ground atoms with a particular predicate to be closed by attaching conditions to the relevant defaults. Secondly, since the rules of a default database can be used to derive negative information (such as the citizenship of a person in rule 1, Example 2.2), our method allows defaults to derive positive information as well as negative information (see again Example 2.2).

Finally, in a recent paper Motro[7] uses 'completeness assertions' to define subsets of the database to which the Closed World Assumption is to be applied. It is of particular interest to note that Motro's work is subsumed by our approach, since each completeness assertion can be expressed as a default with positive prerequisite literals and a negative assumption literal.

Acknowledgements

This work was carried out with the support of IBM (UK) Research Laboratories Ltd, the second author being in receipt of an SERC(CASE) award. The final stages of the work were financed under SERC (ALVEY) project number GR/0/7306.5-IKBS/140. The authors gratefully acknowledge the help and advice of Mir Derakhshan, Alexandra Poulovassilis, Geoff Sharman and Norman Winterbottom.

REFERENCES

1. D. W. Etherington, Formalizing non-monotonic reasoning systems. *Artificial Intelligence* **31**, 1 (1987).
2. M. Gelfond and H. Przymusinska, Negation as failure: careful closure procedure. *Artificial Intelligence* **30**, 273–287 (1986).
3. P. M. D. Gray and R. J. Lucas, *Prolog and Databases: Implementations and New Directions*. Ellis Horwood (1988).
4. D. W. Loveland, A linear format for resolution. Symposium on Automatic Demonstration, Springer Lecture Notes on Mathematics, no. 125, pp. 147–162.
5. E. Mendelsson, *Introduction to Mathematical Logic*, 3rd edition. Wadsworth Inc. (1987).
6. J. Minker, On indefinite databases and the closed world assumption. In *Proceedings of the 6th Conference on Automated Deduction*. Springer Lecture Notes in Computer Science no. 138, pp. 292–308 (1982).
7. A. Motro, Completeness information and its application to query processing. *Proceedings of the 12th International Conference On Very Large Databases* (1986).
8. N. J. Nilsson, Principles Of Artificial Intelligence. Springer-Verlag (1980).
9. R. Reiter. *On Closed World Databases, in Logic and Databases*. Plenum Press (1978).
10. R. Reiter, A logic for default reasoning. *Artificial Intelligence* **13**, 1, 2 (1980).
11. R. Reiter and G. Criscuolo, On interacting defaults. International Joint Conference On Artificial Intelligence (1981).
12. H. Rybinski, On first-order-logic databases. *ACM Transactions On Database Systems* **12**, 3 (1987).
13. C. Small, Guarded default databases: a prototype implementation, in ref. 3.
14. D. S. Touretzky, *The Mathematics Of Inheritance Systems*. Morgan Kaufmann (1986).

Announcements

28–31 OCTOBER 1991

Eurographics Workshop on Computer Graphics and Mathematics, Genova, Italy

Aims and scope

The aims of the workshop are (1) to provide a forum for the exchange of research results in the application of mathematics to computer graphics, i.e. a technology transfer; (2) to encourage mathematicians to attack computer graphics problems; and (3) to promote the use of all relevant mathematical techniques and methods in computer graphics. A secondary aim is to discover the basic mathematical tools every computer graphics expert should have.

Topics include

- *Techniques from geometry*
  - Projective geometry
  - Computational geometry
  - Differential geometry
  - Fractal geometry
- *Probability*
  - Stochastic functions
  - Monte Carlo methods
  - Chaos theory
- *Techniques from topology*
  - Topological modelling
  - Non-manifold modelling
- *Logic and reasoning*
  - Theorem proving
  - Symbolic reasoning
- *Analytical and algebraic tools*
  - Algebraic geometry
  - Complex analytical tools
  - Quaternions

Venue and fee

The workshop will be held in S. Margherita (Genova), Italy. The fee will be about 630,000 Liras, including accommodation and meals.

Organisation

The workshop is organised by the I.M.A. (Institute for Applied Mathematics) of the C.N.R. for Eurographics.

Workshop format

The workshop is limited to about 50 participants. Papers will be presented to the plenum while specialised groups will be formed for discussion sessions.

Information

Please write for further details to: Bianca Falcidieno *or* Caterina Pienovi, Istituto per la Matematica Applicata, Via L. B. Alberti 4, 16132 Genova, Italy. Email: FALCIDIENO@IMAGE.GE.CRN.IT *or* PIENOVI@IMAGE.GE.CNR.IT. Tel.: +39 10-517639. Fax: +39 10-517801.