A Nonmonotonic Typed Multilevel Logic for Multilevel Secure Data / Knowledge Base Management Systems

Bhavani Thuraisingham

The MITRE Corporation, Bedford, MA 01730

ABSTRACT

This paper describes a logic we call Nonmonotonic Typed Multilevel Logic (NTML) for multilevel database applications. It also describes various approaches to viewing multilevel databases through NTML and discusses techniques for query evaluation and integrity checking.

1. INTRODUCTION

1.1 MOTIVATION

Ever since Colmerauer and Kowalski pioneered the use of predicate logic as a programming language (see for example [KOWA74]), Mathematical Logic has been applied to various areas of computer science such as database systems [GALL78, REIT78, CLAR78, KOWA78, NICO78a, NICO78b, LLOY78, MINK88]. It has not only been used as a framework to study their properties, it has also been used as a basis for developing powerful intelligent database systems [BIBE89]. The first workshop on Logic and Databases held in France in 1977 [GALL78] discussed the formalisms of first order logic for database systems which subsequently led to the formalization of relational database concepts using the proof and model theoretic results of first order logic. Further research activities contributed significantly to the development of advanced logical programming languages, inference engines for database systems, treatment of integrity constraints, and in handling negative, partial, and uncertain information. As a result, complex deduction and decision making processes have been incorporated into commercial intelligent data/knowledge base management systems available today [COHE89].

In the meantime, the recommendations of the Air Force Summer Study [AFSB83] led to the design and development of multilevel secure relational database management systems [GRAU84, DENN87, THUR87, STAC90, THUR90]. In such database systems, users cleared at different security levels can access and share a database with data at different sensitivity levels without violating security. Despite these advances, logic programming language research and research activities in multilevel secure database management systems remained largely separate. That is, a logic for reasoning in a multilevel environment or a logic programming system for multilevel environments is not currently available. Thus multilevel secure database management systems lack several important features that have been successfully incorporated into conventional database management systems. They include constraint processing, deductive reasoning, and handling efficient proof procedures.

An early attempt was made to view multilevel databases through first-order logic [THUR88]. Although not entirely successful, this approach helped gain an insight into utilizing formal logic to develop multilevel systems. That is, classical first order logic, being monotonic, was found to be an inappropriate tool for formalizing concepts in multilevel databases. This is because it is possible for users at different security levels to have different views of the same entity. In other words, statements that are assumed to be true at one security level can very well be false at a different security level. Another contention is that first-order logic deals with only one universe (or world). In a multilevel database environment, there is a world corresponding to each security level. In other words, the universe in a multilevel environment is decomposed into multiple-worlds, one for each security level. Considerations such as these have led us to believe that a special logic is needed for reasoning in a multilevel environment. From an examination of the various nonstandard logics described in the literature [TURN84, FROS86], none appeared capable of being used for multilevel systems. Therefore we have developed a logic for not only formalizing multilevel database concepts, but also for developing intelligent multilevel database systems.

The logic that we have developed for multilevel databases is called Nonmonotonic Typed Multilevel Logic (NTML). It extends typed first-order logic to support reasoning in a multilevel environment. We have also formalized multilevel databases using NTML. In particular, the proof theoretic and model theoretic approaches for viewing multilevel databases have been studied. We have regarded security constraints, that are rules which assign security levels to the data, as integrity constraints for multilevel database systems. Techniques for integrity constraint processing have been adapted for security constraint processing. Also, the essential points towards developing a logic programming language based on NTML for developing intelligent multilevel database systems have been investigated. In addition, extensions to NTML for knowledge-based applications have also been proposed.

1.2 SYNOPSIS OF THIS PAPER

We first describe NTML. NTML is a nonmonotonic typed multilevel logic which extends first order logic for reasoning in a multilevel environment. We discuss the syntax,
semantics, and theories associated with NTML. We then show its relationship to multilevel database systems. NTML and its relationship to multilevel databases are two of the important contributions of this paper. We describe three approaches to formalizing multilevel database concepts using NTML. In the first approach, the multilevel database, the schema, and the integrity constraints are expressed as the proper axioms of NTML which has the multilevel real world as its interpretation. In the second approach, the schema and the integrity constraints are expressed as the proper axioms of an NTML theory whose model is the multilevel database. In the third approach, parts of the integrity constraints are treated as integrity rules and the rest as derivation rules. Based on the developments that have resulted in viewing databases through formal logic (see the discussion in [BIBE89]), we believe that the significance of these three approaches is as follows. The proof-theoretic approach would enable the development of multilevel secure logic database systems. The model theoretic approach would enable the development of efficient techniques for query evaluation and integrity checking in multilevel secure relational database management systems. The integrated approach would enable the development of multilevel secure intelligent (or expert) database management systems.

The paper is organized as follows: In section 2 we describe NTML. In particular, NTML syntax, semantics, and theory are described. In section 3, we show how database concepts are formalized using NTML. The paper is concluded in section 4. We assume that the reader is familiar with both logic and database concepts as well as multilevel database concepts. Logic and database concepts are documented in [GALL78, LALL84]. For a discussion on database concepts we refer to [ULLM88]. A useful starting point for multilevel database concepts is the Air Force Summer Study Report [AFSB83]. A knowledge of elementary mathematical logic and logic programming is useful in order to understand the various concepts discussed in this paper. For a discussion on these topics we refer to [MEND79, LLOY87]. An excellent exposition on viewing data/knowledge base management systems through logic is given in [FROS86].

2. LOGIC FOR MULTILEVEL DATABASES

We develop a Nonmonotonic Typed Multilevel Logic, called NTML, to formalize concepts in multilevel databases. We describe NTML language, NTML semantics, and NTML theories in the next three subsections.

2.1 NTML SYNTAX

The syntax of NTML is typed first-order logic with extensions to support multilevel security. We define the primitive symbols, terms, and formulas of the language. Security properties are enforced for each symbol, term, and formula. Any NTML theory based on this syntax must satisfy these properties.2

Primitive Symbols

- A type symbol denoted by a string of one or more letters (each letter could be subscripted) has a security level assigned to it. If $TS$ is a type symbol, then its security level (also called the inherent level of $TS$) is denoted by Level($TS$).3
- A variable symbol denoted by a string of one or more letters (each letter could be subscripted) has a type and a security level assigned to it. If $VS$ is a variable symbol whose type and level (also called the inherent level of $VS$) are $T$ and $L$, respectively, then Type($VS$) = $T$ and Level($VS$) = $L$. The following security property must be satisfied:

$$SP1: Level(VS) \geq Level(\text{Type}(VS))$$

- A constant symbol consists of a type and a security level. If $CS$ is a constant symbol whose type and level (also called the inherent level of $CS$) are $T$ and $L$, respectively, then Type($CS$) = $T$ and Level($CS$) = $L$. The following security property must be satisfied:

$$SP2: Level(CS) \geq Level(\text{Type}(CS))$$

- A function symbol consists of a type and a level. Associated with each function symbol are its specified arguments, the types of its arguments and the type of its value. If a function symbol $FS$ has $n$ arguments, then $FS$ is an $n$-place function symbol. If $T_1, T_2, T_3, \ldots, T_n$ are the types of the arguments and $T$ is the type of the value, then the type of the function symbol $FS$ denoted Type($FS$) is $T_1 \times T_2 \times T_3 \times \ldots \times T_n \rightarrow T$. The level of $FS$ (also called the inherent level of $FS$) is denoted by Level($FS$). The following security property must be satisfied:

$$SP3: Level(FS) \geq \text{l.u.b.}(Level(T_1), Level(T_2), Level(T_3), \ldots, Level(T_n))$$

- A predicate symbol consists of a type and level. Associated with each predicate symbol are the specified arguments and the types of the arguments. If a predicate symbol $PS$ has $n$ arguments, then $PS$ is an $n$-place predicate symbol. If $T_1, T_2, \ldots, T_n$ are the types of the arguments, then the type of the predicate symbol denoted by Type($PS$) is $T_1 \times T_2 \times T_3 \times \ldots \times T_n$. The level of $PS$ (also called the inherent level of $PS$) is denoted by Level($PS$). The following security property must be satisfied:

$$SP4: Level(PS) \geq \text{l.u.b.}(Level(T_1), Level(T_2), Level(T_3), \ldots, Level(T_n))$$

- Logical connectives $\wedge$ (AND), $\vee$ (OR), $\rightarrow$ (NOT), $\Rightarrow$ (IMPLICATION), $\forall$ (FOR EVERY), $\exists$ (THERE EXISTS). Quantification is permitted over variables and types.

Terms

- A variable symbol is a term. The level and type of the term is the same as those of the corresponding variable symbol.

2 In [THUR90b] we have formally specified these security properties.

3 To simplify the syntax, one possibility would be to assign the lowest security level (i.e., system-low) to all the symbols of the language. Truth values could be assigned with respect to the various security levels.
Atomic Formulas

- Any atomic formula is a wff.
- If W is a wff (whose inherent level is Level(W)), then $\forall \bar{x} \infty (W)$ and $\exists \bar{x} \infty (W)$ are also wffs. The following security property must be satisfied:

$$SP_6: \text{Level}(AM) \geq \min \{\text{type} T_1, \text{type} T_2, \text{type} T_3, \ldots, \text{type} T_n\}$$

Well-Formed Formulas

- Any atomic formula is a wff.
- If W is a wff (whose inherent level is Level(W)), then $\forall \bar{x} \infty (W)$ and $\exists \bar{x} \infty (W)$ are also wffs. The following security property must be satisfied:

$$SP_7: \text{If } L_1 \text{ and } L_2 \text{ are the security levels (or inherent levels) of the formulas } \forall \bar{x} \infty (W) \text{ and } \exists \bar{x} \infty (W) \text{ respectively, then Level}(W) = L_1 = L_2.$$  

Type Symbols: Under an interpretation I, associated with each type T with inherent level $L^*$, there is a domain $D(T, L)$ for each security level $L \geq L^*$. Further, $D(T, L)$ consists of all elements of type T which satisfy the following conditions:

- (a) $L^* \leq \text{Level}(x) \leq L$.
- (b) $D(T, L) \subseteq D(T, L') \subseteq D(T, L^*)$ where $L'$ is the security level that is just less than $L$ (that is, there is no security level $L^*$ such that $L' < L^* \leq L$). We denote $D(T, L)$ to be $I(T, L)$.

Constant Symbols: Let $a$ be a constant symbol of type T. Let Level($a$) = $L$ and Level(T) = $L_2$ (note that $L_2 \leq L_1$). Associated with $a$ is an element $a$ in $D(T, L)$ for each $L \geq L_1$ (note also that for such an element $a$ to exist $L_1$ must be $\leq L_2$). We denote a by $I(a, L)$.

Function Symbols: Let $F$ be an n-place function symbol of type $T_1 \times T_2 \times T_3 \times \ldots \times T_n \rightarrow T$. If $L \leq L_1$, then associated with $F$ is a mapping $F^*$:

$$F^*: D(T_1, L) \times D(T_2, L) \times D(T_3, L) \times \cdots \times D(T_n, L) \rightarrow D(T, L).$$

We denote $F^*$ by $I(F, L)$.

Predicate Symbols: Let $P$ be an n-place predicate symbol of type $T_1 \times T_2 \times T_3 \times \ldots \times T_n$. Let Level($P$) = $L$. If $L \leq L_1$, then associated with $P$ is a relation $P^*$ on $D(T_1, L) \times D(T_2, L) \times D(T_3, L) \times \cdots \times D(T_n, L)$ which evaluates to either True or False. We denote $P^*$ by $I(P, L)$.

Interpretations of Terms and Formulas

We first define interpretations of variable-free terms with respect to security levels. Next, we define interpretations of variable-free atomic formulas. Finally, we define interpretations of formulas which are either variable-free or closed (note that a formula is closed if it does not have any free variables; a free variable is a variable which is not within the bounds of a quantifier).

Let $t$ be a term and Level($t$) = $L$. If $L' \leq L$, then the interpretation of $t$ with respect to security level $L$ is denoted by $I(t, L)$. We consider the various possibilities for $t$.

- (a) $t$ is the constant symbol $a$, then $I(t, L) = I(a, L)$.
- (b) $t$ is a term $F(t_1, t_2, \ldots, t_m)$ where $t_1, t_2, \ldots, t_m$ are variable free terms. Then

$$I(t, L) = F^*(t_1^*, t_2^*, \ldots, t_m^*)$$

where $F^* = I(F, L)$, $t_1 = I(t_1, L)$, $t_2 = I(t_2, L)$, $\ldots, t_m = I(t_m, L)$.

Next we define interpretations of variable-free atomic formulas and formulas which are either variable free or closed.

Let $A$ be the atomic formula with specification $P(t_1, t_2, \ldots, t_n)$ where $P$ is a predicate symbol and $t_1, t_2, \ldots, t_n$ are variable free terms. Let Level($A$) = $L$. Then if $L' \leq L$, $I(A, L) = P^*(t_1^*, t_2^*, \ldots, t_n^*)$ where $P^* = I(P, L)$, $t_1^* = I(t_1, L)$, $t_2^* = I(t_2, L)$, $\ldots, t_n^* = I(t_n, L)$.
Let $F$ and $G$ be variable-free formulas. The interpretation of $\neg F$, $F \land G$, $F \lor G$ are defined as follows:

$I(\neg F, L) = \neg I(F, L)$
$I(F \land G, L) = I(F, L) \land I(G, L)$
$I(F \lor G, L) = I(F, L) \lor I(G, L)$

$I(F \rightarrow G, L) = \text{False}$ if $I(F, L) = \text{True}$ and $I(G, L) = \text{False}$
$I(F \rightarrow G, L) = \text{True}$ if otherwise.

Let $F$ be a formula with free variables $x_1, x_2, \ldots, x_n$. Let $T_1, T_2, \ldots, T_n$ be the types of $x_1, x_2, \ldots, x_n$, respectively. Then, the interpretation of $\forall x_1, x_2, \ldots, x_n (F)$ with respect to $L$ is the tuple $(x_1, x_2, \ldots, x_n)$ that is a member of $D(T_1.L) \times D(T_2.L) \times \ldots \times D(T_n.L)$ such that $I(F, x_1, x_2, \ldots, x_n, L)$ evaluates to True. Otherwise $I(\forall x_1, x_2, \ldots, x_n (F), L)$ evaluates to False.

$I(\exists x_1, x_2, \ldots, x_n (F), L) = \text{True}$ if for every element $(a_1, a_2, \ldots, a_n)$ which is a member of $D(T_1.L) \times D(T_2.L) \times \ldots \times D(T_n.L)$ such that $I(F, a_1, a_2, \ldots, a_n, L)$ evaluates to True. Otherwise $I(\exists x_1, x_2, \ldots, x_n (F), L)$ evaluates to False.

### 4.3 NTML Theory

As in any logic theory, an NTML theory has a set of logical axioms, a set of proper axioms, and a set of inference rules. The logical axioms of an NTML theory are analogous to those of first-order logic with equality. These axioms are shown in table 1.

<table>
<thead>
<tr>
<th>Logical Axioms</th>
</tr>
</thead>
</table>
| If $A$, $B$, and $C$ are $\\text{NTML}$ $\wedge{}$-free whose inherent security levels are $L_1$, $L_2$, and $L_3$, let the inherent security levels of the variable $x$ be $L_4$, the term $t$ be $L_5$, the variable $x$ be $L_6$, the tuple $(x, t)$ be $L_7$, and the variable $y$ be $L_8$. Let $L_1 = \text{Inherent}(A, L_1, L_2, L_3)$, $L_2 = \text{Inherent}(B, L_4, L_5, L_6)$, $L_3 = \text{Inherent}(C, L_7, L_8, L_9)$. Then, the following axioms are true in $\text{NTML}$:
| $A \rightarrow (B \rightarrow C)$, $L_7$ |
| $(A \rightarrow (B \rightarrow C)) \rightarrow (A \rightarrow B) \rightarrow (A \rightarrow C)$, $L_7'$ |
| $\neg (A \rightarrow B) \rightarrow (A \rightarrow \neg B)$, $L_7$ |
| $(A \rightarrow B) \rightarrow (A \rightarrow B)$, $L_7$ |
| $\forall x : (A(x) \rightarrow B(x))$, $L_7$ |
| If $A$ is a $\forall x$ containing no free occurrence of $x$, $A \rightarrow B(x)$, $L_7$ |
| $\forall x : (A(x) \rightarrow B(x))$, $L_7$ |
| $\forall x : (A(x) \rightarrow B(x))$, $L_7$ |
| $\forall x : (A(x) \rightarrow B(x))$, $L_7$ |
| $\forall x : (A(x) \rightarrow B(x))$, $L_7$ |

#### Part of the Proper Axioms

| (Dominant, Secret, Unclassified) |
| (Dominant, Top Secret, Secret) |
| (Unclassified) |
| (Dominant, Secret, Top Secret) |
| (Unclassified) |
| (Dominant, Top Secret, Secret) |

#### Rules of Inference

| MP | $(Q, L)$ is a direct consequence of $(P, L)$ and $(P \rightarrow Q, L)$ |
| OBN | $(\neg Q, L)$ is a direct consequence of $(P, L)$ |
| DASL | $(P, L)$ is a direct consequence of $(L, L)$ |

Every NTML theory has a distinguished type symbol SL. Associated with the distinguished type symbol SL under the interpretation $I$ at level $L$ is the domain $D(SL, L)$. $D(SL, L)$ consists of the security levels which are visible to users at level $L$. For each $L$, if $L_1$ and $L_2$ are elements of $D(SL, L)$, then either $L_1 \leq L_2$ or $L_2 \leq L_1$. Furthermore, the level of the type symbol $SL$ is system-low (which is the lowest security level supported by the system under consideration).

Every NTML theory has a set of constant symbols for every security level considered. Each constant is of type $SL$. Two predicate symbols "Dominate" and "Just-less-than" (both having security levels system-low) are defined on the variable and constant symbols of type $SL$. For example, let $D(SL, \text{Unclassified}) = (\text{Unclassified}, \text{Secret}, \text{Top Secret})$. Let the distinguished individual constants be $\text{Unclassified}$, Secret, and Top Secret whose assignments are the levels Unclassified, Secret, and Top Secret respectively. Then, table 1 shows part of the proper axioms of the theory.

#### Note

(a) Next to each formula we state a security level. This is the level at which the formula is true under all interpretations. If we do not specify a security level next to a formula, then the formula is assumed to be true with respect to all security levels under all interpretations (see the ensuing discussion on the proper axioms).

(b) For convenience we refer to the constants $\text{Unclassified}$, $\text{Secret}$, and $\text{Top Secret}$ by their respective assignments Unclassified, Secret, and Top Secret.

The remaining proper axioms of an NTML theory are constructed from a subset of the $\text{NTML}$ formulas, as follows:

Let $\{W_1, W_2, \ldots, W_n\}$ be a subset of $\text{NTML}$ formulas. Let $L_1, L_2, \ldots, L_n$ be the (inherent) security levels of $W_1, W_2, \ldots, W_n$ respectively. $W_i$ is explicitly asserted to be a proper axiom of an NTML theory at level $L_i$, which is denoted by $(W_i, L_i)$, if

(a) $L_i \leq L$.  
(b) $W_i$ is true in all interpretations $I$ with respect to $L_i$. That is, $I(W_i, L_i)$ is true for all interpretations $I$.  
(c) It is not the case that $(W_i, L_i^*)$ is an axiom where $L_i^*$ is just less than $L_i$.  

Note that if $L_i$ is just less than $L_i^*$ and if $W_i$ is not true with respect to $L_i^*$, then $(\neg W_i, L_i^*)$ is a proper axiom of the theory. We assume that a set of proper axioms with respect to any security level is consistent. That is, the formulas $(P, L)$, $(P \rightarrow Q, L)$ and $(Q, L)$ cannot be the proper axioms of a theory for any security level $L$.

Note that in the discussion on the syntax of NTML, each formula was assigned a security level. This level is the inherent security level of the formula. For example, in the database context the inherent security level of a formula is the security level of the schema which is represented by that formula. So, if $(W, L)$ is a proper axiom of an NTML theory and if the inherent security level of $W$ is $L_i^*$, then the following security property must be satisfied:

$SP_9$: $L_i^* \leq L$
Ponens (MP). Generalization (GEN) and Deductions Across other logic. For example, \((F, L)\) is denoted a theorem of the NTML theory \(T\) if either \((F, L)\) is an axiom of \(T\) or \((F, L)\) can be derived from the axioms using the rules of inference. That is, there exists a derivation \((P_1, L_1), (P_2, L_2) \ldots \ldots (P_n, L_n)\) such that \((P_n, L_n) = (F, L)\) and for each \(i (1 \leq i \leq n)\), \((P_i, L_i)\) is either an axiom of \(T\) or it can be derived from \((P_1, L_1), (P_2, L_2) \ldots \ldots (P_{i-1}, L_{i-1})\) from the rules of inference.

Note: (a) Like an axiom, a theorem is also followed by a security level.
(b) In the case of a formula which is either an atom or the negation of an atom, we include the security level as an argument for convenience. For example, \((P(t_1, t_2, \ldots, t_n), L)\) and \((\neg P(t_1, t_2, \ldots, t_n), L)\) are denoted by \((P(t_1, t_2, \ldots, t_n, L))\) and \((\neg P(t_1, t_2, \ldots, t_n, L))\), respectively.

One can define consistency, soundness, and completeness of NTML. NTML is consistent if there is not a \(wff\) \(F\) and a security level \(L\) such that \((F, L)\) and \((\neg F, L)\) are its theorems. NTML is sound if for every theorem \((F, L)\), \(F\) evaluates to true with respect to \(L\). NTML is complete if for every \(wff\) \(F\) and security level \(L\), if \(F\) evaluates to true with respect to \(L\), then \((F, L)\) is a theorem.

We have the following theorems that show the consistency, soundness, and completeness of NTML.

**Theorem 1:**
NTML is Consistent

**Proof of Theorem 1:** The proof of this theorem uses a technique similar to the proof of the consistency of first-order logic. We refer the reader to [MEND79 - see proposition 2.2, page 62] for the proof of the consistency of first order logic.

**Theorem 2:**
NTML is Sound.

---

**Proof of Theorem 23:** This proof follows from the proofs of the following lemma. The proof of this lemma uses techniques similar to the proof of the soundness of first-order logic given in [MEND79 - see propositions 2.1-2.7, pages 61-65].

**Lemma 4:**
(a) If \(P\) and \(P \rightarrow Q\) are NTML \(wffs\) which are true under an interpretation \(I\) with respect to \(L\), then so is \(Q\).
(b) If the NTML formula \(P\) is true under an interpretation \(I\) with respect to \(L\), then so is \(\forall x. I(P)\).
(c) If the NTML formula \(P\) is true under an interpretation \(I\) with respect to \(L\) and just-less-than \((L_1, L_2)\) is true under \(I\) with respect to all security levels, and it is not the case that \((\neg P)\) is true under \(I\) with respect to \(L_2\), then \(P\) is true under \(I\) with respect to \(L_2\).

**Theorem 5:**
NTML is Complete.

**Proof of Theorem 5:** The proof of this theorem uses techniques similar to the proof of the completeness of first order predicate calculus (see [MEND79, see proposition 2.13, page 69]).

### 3. VIEWING MULTILEVEL DATABASES THROUGH NTML

In this section we provide an overview of the perceived multilevel universe and then describe three approaches to represent it, based on NTML. This work follows from the work reported in [NICO78a] where first-order logic is used to formalize nonmultilevel databases.

#### 3.1 MULTILEVEL UNIVERSE

A state of the universe can be regarded as a set of elements linked together by functions or relations. Since functions can be regarded as special kinds of relations, we only consider relations in our discussion. We distinguish between the actual universe and the perceived universe. The actual universe is the real universe. The perceived universe is the part of the universe that is involved in the particular application under consideration. We refer to the perceived universe as the universe.

Since elements and relations are all assigned security levels, the universe is multilevel. One can partition this universe into worlds corresponding to each security level. These worlds are the views of the universe by users at the corresponding levels. For example, a Secret world in the universe is the world perceived by those cleared at the Secret level.

Information in a multilevel universe is the knowledge of the truth value of a statement with respect to some security level. A statement could be an elementary fact such as "John's salary is 30K," it could be a schema such as "The attributes of employee are SS#, name and salary," or it could be a law such as, "All salary values must be positive." The statement,
"John's salary is 30K," could be true with respect to the Unclassified level, and it could be false with respect to the Secret level. Therefore, in the Unclassified world, "John's salary is 30K" represents true information while in the Secret world "John's salary is 30K" represents false information.

The elementary facts, the schema and the laws are only a subset of the information of the perceived universe. This is known as explicit information. The other kind of information is implicit information which is the information derived from the explicit information.

Note: (a) A general law is in fact an integrity constraint. We use the terms general law and integrity constraint interchangeably.

(b) Security constraints that assign security levels to the data can also be regarded as general laws, (i.e., security constraints are also integrity constraints).

(c) A general law can be either an integrity rules or a derivation rule. An integrity rule must be satisfied by the database. A derivation rule is used to derive new information from the database data and any real-world information.

Representing negative information has been a subject of much research. Negative information can also be either explicitly or implicitly specified. For example, the statement, "It is not the case that John's salary is 60K," explicitly states that John's salary is not 60K.

Implicit representation of negative information can be derived by using certain rules. These rules include the rules of inference associated with the theory as well as other rules of inference such as the uniqueness axiom and the completeness axiom. For example, if there is a rule that "John's salary is unique" and a fact that "John's salary is 30K," then one could deduce the negative information that "John's salary is not 60K." Completeness axioms specify all the values that a particular entity can take. For example, from the following axiom, "John's salary is either 30K or 60K," one can deduce that John's salary is not 50K.

A multilevel universe can either be finite or infinite depending on whether the elements associated with it are finite or infinite. Further, one can regard a universe to be either open or closed. If the universe is closed, then any fact that does not evaluate to true in the universe is assumed to be false and its negation is assumed to be true. If it is an open universe, then such an assumption is not made. That is, one cannot assume negative information unless it is explicitly stated or one can deduce it.

In the next three subsections, we describe three approaches to representing the perceived universe. They are the proof-theoretic, model theoretic, and the integrated approaches. All approaches use NTML as a framework. As stated in the introduction, based on the developments that have resulted in viewing databases through formal logic [BIBE89], we believe that the significance of these three approaches is as follows. The proof-theoretic approach would enable the development of multilevel secure logic database systems. The model theoretic approach would enable the development of efficient techniques for query evaluation and integrity checking in multilevel secure relational database management systems. The integrated approach would enable the development of multilevel secure intelligent (or expert) database management systems.

3.2 PERCEIVED MULTILEVEL UNIVERSE AS AN NTML THEORY

In this approach, called the proof theoretic approach, the perceived multilevel universe is represented as a NTML theory. The NTML theory is defined as follows:

1. Its constants and predicates are respectively the elements and relations associated with the universe.
2. Its proper axioms are defined as follows: If an elementary fact, schema, or a general law is true in all interpretations with respect to a security level L, and it is not the case that this fact or law is true in all interpretations with respect to the security level L*, where L* is just less than L, then the fact or law is assigned the security level L and is taken to be a proper axiom of the theory.
3. If it is not the case that an elementary fact, schema, or law is true in all interpretations with respect to a security level L, and this fact or law is true in all interpretations with respect to a security level L*, where L* is just less than L, then the negation of this fact or law is either taken to be a proper axiom of the theory, or one should be able to derive the negation from the proper axioms.

The implicit information constitute the theorems of the NTML theory. The actual multilevel universe is an interpretation of the NTML theory. Whether the actual universe is a model of the theory depends on how accurately the actual universe fits the perceived universe. The approach is illustrated in figure 1.

Query Evaluation

A query posed by a user at security level L is expressed as a wff of the NTML theory. There are two types of queries: closed and open. A closed query is a wff which is closed (i.e., with no free variables), and an open query is one with at least one free variable. Query evaluation amounts to theorem proving. For example, let W be the wff which corresponds to a query Q posed by a user at level L. Then evaluating the query Q amounts to proving that (W,L) is a theorem of the NTML theory.

For example, let A and B be the wffs which correspond to the respective queries Q1 and Q2. Further assume that Q1 is closed and Q2 is open and that the queries are posed by a user at level L. The solution to Q1 is a Yes or No answer. The answer is Yes if (A, L) is a theorem of the NTML theory. For the open query Q2, a tuple in the response is obtained as follows: Substitute appropriate elements associated with the NTML theory for the free variables of B. Let the resulting wff be B*. If (B*, L) is a theorem of the theory, then the tuple which is formed from the elements substituted to form B* is included in the response.

6 It should be noted that proof procedures for NTML are yet to be developed. For the discussion given in this section we assume that the NTML rules of inference are used for deductions.
We illustrate both open and closed query evaluation associated with this first approach with examples. We assume that there are only two security levels: Secret and Unclassified. The Secret level dominates the Unclassified level.

Figure 1. Perceived Multilevel Universe is Represented as an NTML Theory Whose Interpretation is the Actual Multilevel Universe

The proper axioms of the NTML theory are shown in table 2. PA1 and PA2 are schemas for the relations EMP and SEN-EMP, respectively. Both schemas have the Unclassified level associated with them as they are both true at the Unclassified level. PA1 states that if EMP(X,Y,Z) is true in an interpretation with respect to L, then the types of X, Y, and Z are SS#, NAME and SALARY, respectively. Note that since EMP(X,Y,Z) is an atomic formula, we include the level L as one of its arguments. PA2 states that if SEN-EMP(X) is true under an interpretation with respect to L, then the type of X is NAME. PA3, PA4, and PA5 are atomic formulas with no variables. So are the axioms PA11 - PA14. Although it is not explicitly specified, axioms PA6 - PA9 assume universal quantification over all variables X, Y, Z, L (with or without subscripts). PA6 is a law which defines all those who earn more than 30K or more as senior employees. PA7 is a law which states that all salaries must be less than or equal to 40K. This law is true only with respect to the Unclassified level. Note that PA10 negates this law at the Secret level. PA8 is the equivalent to the primary key constraint in relational databases. PA9 states that if two tuples exist with the same SS# at different security levels, then the negation of the tuple at the lower security level is true with respect to the higher level.

Queries are expressed as formulas of NTML theories at the level of the user who posed the queries. Consider the following queries:

Q1: EMP(000, John, 20K), Unclassified
Q2: EMP(111, James, 60K), Unclassified
Q3: EMP(X,Y,Z), Unclassified
Q4: EMP(000, John, 20K), Secret
Q5: EMP(111, James, 35K), Secret

The five queries are expressed as formulas of NTML whose proper axioms are given in table 2. Query evaluation amounts to theorem proving. Note that the queries Q1, Q2, Q4, and Q5 are closed queries while Q3 is open. Also, the queries Q4 and Q5 are posed by Secret users while Q1, Q2, and Q3 are posed by Unclassified users.

An attempt to prove that (EMP(000, John, 20K), Unclassified) is a theorem of NTML will be successful as PA3 is a proper axiom of the theory. Therefore, the answer to query Q1 is "Yes."

For the query Q2, since its negation (i.e., (¬ EMP(111, James, 60K), Unclassified)) can be proved to be a theorem of NTML, the answer is "No."

The answer to query Q3 consists of all the (Y, Z) pairs such that there is some X for which (EMP(X,Y,Z), Unclassified) is a theorem of NTML. From the axioms it can be seen that the answer to Q3 is the set of pairs (John, 20K), (James, 35K), (Mary, 15K)).

The answer to the query Q4 is, "No." This is because from the axioms PA3, PA9, and PA11, it can be deduced that (¬ EMP(000, John, 20K), Secret) is a theorem.

The answer to the query Q5 is "Yes" as it can be derived from the axiom PA4 and the rule of inference DASL.

Table 2. Multilevel Database as an NTML Theory

| PA1: x(Y1), Y1,Y2,Z2,L1, EMP(X,Y2,L2) → T1 = SS#, T2 = NAME, T3 = SALARY, Unclassified |
| PA2: y(Y1, L1, SEN-EMP(X) → T = NAME, Unclassified) |
| PA3: → EMP(000, John, 20K, Unclassified) |
| PA4: → EMP(111, James, 35K, Unclassified) |
| PA5: → EMP(222, Mary, 15K, Unclassified) |
| PA6: (EMP(X,Y,Z), Unclassified) → Z ≤ 40K |
| PA7: (EMP(X,Y,Z), Unclassified) → Z ≤ 40K, Unclassified |
| PA8: EMP(y(X1, Y1, Z1, L1) → EMP(Y, Y2, Z2, L2) → Y1 = Y2, Z1 = Z2, Unclassified |
| PA9: EMP(X, Y2, Z2, L2) → (¬Y1 = Y2, Z1 = Z2, A, Not-less-than(L1, L2) |
| PA10: ¬y(XSS#, YNAME, ZSALARY, L1, EMP(X,Y,Z), L2) → Z ≤ 40K, Secret |
| PA11: → EMP(000, John, 50K, Secret) |
| PA12: → SEN-EMP(George, Secret) |
| PA13: → SEN-EMP(Mary, Unclassified) |
| PA14: → EMP(331, Jane, 70K, Secret) |

Advantages and Drawbacks

This approach permits disjunctive information to be represented. For example, one can express the information that "either the salary of John is less than 50K or John is a senior employee." However, this approach does have certain drawbacks. One is that in the database context, the elementary facts that have to be represented may be very large. Therefore, efficient proof techniques have to be developed to handle large amounts of facts and rules. Furthermore, negative information in this approach has to be explicitly stated. This makes the database even larger.
Another drawback to this approach is the verification of the general laws. That is, the general laws are treated as derivation rules with this approach. That is, they are not treated as integrity rules where they could be validated during database updates. By treating the general laws as derivation rules each time some information is added at a security level L, the proper axioms of the NTML theory may have to be modified. As a result, a new NTML theory is produced. Therefore, the consistency of the new theory has to be checked. If the new theory is inconsistent, then appropriate actions have to be taken. Possible ways to handle exceptions are:

(a) Do not permit the update.
(b) Modify the axioms so that consistency is maintained.
(c) Adopt the technique proposed in [Kowa78] where a new predicate INCONSISTENT is added to the NTML theory. This new predicate handles the exceptional situations.

3.3 MULTILEVEL DATABASE AS A MODEL

In the second approach, called the model theoretic approach, the set of elementary information is considered to be an interpretation of an NTML theory. This is the approach that is implicitly followed in a conventional relational database system. The proper axioms of the NTML theory are the general laws. All general laws are used as integrity rules and not as derivation rules. That is, since certain general laws are the proper axioms of an NTML theory, these laws must be satisfied by the multilevel database. The approach is illustrated in figure 2.

Query Evaluation

Query evaluation in this approach amounts to checking the satisfiability of the formulas which represent the multilevel database. That is, for a closed query posed by a user at level L, the truth or falsity of the corresponding formula is checked against the portion of the multilevel database visible at security level L. For an open query at level L, the set of values associated with the database at level L which satisfies the formula, when substituted for the free variables, will form the response.

We illustrate both open and closed query evaluation associated with this second approach with examples.

Consider the multilevel database shown in table 3. The schema S1, S2, and the general laws IR1 - IR5, expressed as NTML formulas, are treated as integrity rules. That is, whenever the database is updated, the integrity rules must be satisfied. For example, whenever a name is included in the relation SEN-EMP, it must be first verified that the name is an employee and that the corresponding salary is greater than or equal to 30K. Also, whenever an Unclassified tuple is modified or entered into the database, the salary value specified must be less than or equal to 40K. IR3 is the primary key constraint. Also note that all the variables in IR1 - IR3 are implicitly assumed to be universally quantified. IR1 is negated at the Secret level in IR4 as IR1 does not hold at the Secret level. For the laws IR2 and IR3, we do not duplicate them at the Secret level. This is because all of the laws are axioms of an NTML theory and therefore the NTML rules of inference can be used to deduce new laws at the Secret level. For example, from IR2 and the rule DASL, we have the following law:

\[
\text{EMP}(X,Y,Z) \rightarrow \text{SENT-EMP}(Y,Z), \text{Secret}.
\]

IR5 is the integrity rule which ensures that tuples are duplicated at the appropriate levels. It should also be noted that since the integrity rules and the schemas are axioms of an NTML theory, the rules of inference could be used to deduce new laws. IR5 states that if there is a tuple in the database at level L1, and if a second tuple in the database with the same primary key as the first one does not exist at a level L2 where L1 is just less than L2, then the first tuple in the database must also exist at level L2. Note that although IR5 is classified at the Unclassified level, the rule DASL will ensure that it is valid at all security levels.

We are, however, faced with the problem of finding a mechanism to enforce IR5. For example, consider the situation where the tuple (444, Paul, 30K, Unclassified) is added to the database. All of the integrity rules are satisfied at the Unclassified level and, therefore, the multilevel database is a model of the axioms at this level. However, at the Secret level, the Secret counterpart to IR5 (obtained via DASL) is not satisfied if (444, Paul, 30K, Secret) is not in the database.

Possible solutions to this problem are:

(a) Reject the update if (444, Paul, 30K, Secret) is not in the database; this has a problem in terms of security. That is, if an Unclassified user has requested the update (which will usually be the case), then by rejecting the update this user will know that the tuple does not exist at the higher level since he can read IR5.
(b) Enter the tuple (444, Paul, 30K, Secret) into the database. The level of the subject (who acts on behalf of the user) who attempts this update will be determined by the Security Policy enforced by the system (usually this subject will operate at the Secret level).

Note that, in this approach, NTML rules of inference cannot be used to infer new tuples.
Permit the update, but use the predicate INCONSISTENT (as described in [KOWA78]) to handle the exceptional situation. That is, some additional laws are introduced at the Secret level in order to handle the exceptional situations (note that if the laws are updated, then a new theory is obtained, which means the consistency of the new theory has to be checked).

Table 3. Multilevel Database as a Model

<table>
<thead>
<tr>
<th>S#</th>
<th>Name</th>
<th>Salary</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>John</td>
<td>20K</td>
<td>Unclassified</td>
</tr>
<tr>
<td>111</td>
<td>James</td>
<td>35K</td>
<td>Unclassified</td>
</tr>
<tr>
<td>222</td>
<td>Mary</td>
<td>15K</td>
<td>Secret</td>
</tr>
<tr>
<td>000</td>
<td>John</td>
<td>30K</td>
<td>Secret</td>
</tr>
<tr>
<td>111</td>
<td>James</td>
<td>35K</td>
<td>Secret</td>
</tr>
<tr>
<td>222</td>
<td>Mary</td>
<td>15K</td>
<td>Secret</td>
</tr>
<tr>
<td>333</td>
<td>Jane</td>
<td>70K</td>
<td>Secret</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R#</th>
<th>S#</th>
<th>Name</th>
<th>Salary</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>John</td>
<td>20K</td>
<td>Unclassified</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>James</td>
<td>35K</td>
<td>Unclassified</td>
<td></td>
</tr>
<tr>
<td>222</td>
<td>Mary</td>
<td>15K</td>
<td>Secret</td>
<td></td>
</tr>
<tr>
<td>000</td>
<td>John</td>
<td>30K</td>
<td>Secret</td>
<td></td>
</tr>
<tr>
<td>111</td>
<td>James</td>
<td>35K</td>
<td>Secret</td>
<td></td>
</tr>
<tr>
<td>222</td>
<td>Mary</td>
<td>15K</td>
<td>Secret</td>
<td></td>
</tr>
<tr>
<td>333</td>
<td>Jane</td>
<td>70K</td>
<td>Secret</td>
<td></td>
</tr>
</tbody>
</table>

Table 3. Multilevel Database as a Model

Next we describe query evaluation with this approach. Consider the following queries posed by a Secret user.

Q6: EMP(000, John, 20K)
Q7: EMP(000, John, 60K)
Q8: SEN-EMP(X)

Note that Q6 and Q7 are open queries while Q8 is a closed query. The answer to Q6 is "No" as the tuple (000, John, 20K) is not specified in the database. The answer to Q7 is "Yes" as the tuple (000, John, 60K) is specified in the database. The answer to the query Q8 is the set [John, Jane, James].

Advantages and Drawbacks

By treating all the general laws as integrity rules, the NTML theory does not change unless the general laws themselves evolve. Also negative information does not have to be explicitly specified. Anything that is not specified in the database is assumed to be true.

A drawback with this approach is that tuples have to be duplicated at different security levels. This is because none of the general laws are used as derivation rules and, therefore, new tuples cannot be deduced from existing tuples. Therefore, one cannot deduce that (111, James, 35K, Secret) from (111, James, 35K, Unclassified). Therefore, the tuple (111, James, 35K, Secret) has to be explicitly specified in the database (see the discussion on the rule IR5 given earlier).

When data is inserted, deleted or modified, the interpretation is also changing. Therefore, one has to verify that the new interpretation is still a model of the NTML theory. That is, when there is some change made by a user at level L, then it has to be ensured that at all security levels L* (L* ≥ L), the multilevel database is still a model of the theory. In general, it is not necessary to verify each law when the database is updated. That is, only certain laws may be falsified as a result of the update. It remains to be investigated as to whether the techniques developed for integrity checking in relational databases (see, for example, the work in NIC078b) can be adapted for multilevel databases.

3.4 INTEGRATED APPROACH

This approach is a mixture of the two previous approaches for handling the general laws. That is, some of the general laws are taken as integrity rules and the others as derivation rules. The database has two components. One is the explicit extension which is a model of an NTML theory whose proper axioms are the general laws which are regarded as integrity rules. The other is the implicit extension which is the set of all tuples which are derived from the explicit extension by virtue of the general laws which are used as derivation rules.

Figure 3. General Laws as Integrity Rules and Derivation Rules

Query Evaluation

Query evaluation at a level L depends on whether relations are defined explicitly or implicitly. Two ways to treat relation definitions have been identified. They are as follows:

8 Our approach to handling integrity constraints is discussed in detail in [THUR90b].
A Relation is either defined explicitly or implicitly. If it is defined implicitly, then it is defined in terms of the explicit relations. Note that each definition is a NTML formula and therefore has a security level attached to it. When a query is posed at level L, all references to implicit relations in the query formula are replaced by explicit relations according to the definitions of the implicit relations at level L. The modified query is evaluated against the explicit extension of the multilevel database. Note that the treatment of views in relational database systems takes this approach.

In the second approach, it is not necessarily the case that a relation is defined either implicitly or explicitly. That is, it is impossible for part of the relation to be defined explicitly and part of it implicitly. Therefore, evaluating a query at level L amounts to deducing facts which are tuples at level L. That is, evaluating a query R(x1, x2, x3, ...xn), expressed by (R(x1, x2, x3, ...xn)), amounts to obtaining all proofs of the formula (3 x1/t1, x2/t2, ...,xn/tn R(x1, x2, ...,xn), L).

An advantage of the second treatment of relation definition is that in the real world one could know various relationships without having to know exactly how these relationships were formed. That is, it is possible for people to be senior employees without actually having to make a salary of more than 30K.

We illustrate query evaluation with an example:

Suppose a Secret user poses a query to retrieve all senior employees. This query is expressed by the NTML formula:

SEN-EMP(X). Secret.

If the relation SEN-EMP is defined only explicitly (see table 4), the answer to the query will be the names (John, James, Jane). (Note that in table 4, S1 and S2 are schema rules, DR1 is a derivation rule which defines JUN-EMP, and IR1 and IR2 are integrity rules.)

If the relation SEN-EMP is defined only implicitly (table 5), the query is then modified to the following formula:

(3 X, Z EMP(X,Y,Z) & Z > 30K, Level = Secret).

The answer to the query is the set (John, James, Jane). (Note that in table 5, S1 is a schema rule, DR1 is a derivation rule which defines SEN-EMP, and IR1 - IR4 are integrity rules.)

If, however, a relation could be defined explicitly as well as implicitly (see table 6), then the answer to the query is the set (John, James, Jane, Jill). (Note that in table 6, S1 and S2 are schema rules, DR1 is a derivation rule, and IR1 - IR4 are integrity rules.)

Advantages and Drawbacks

By using some of the general laws as derivation rules, not all of the information has to be made explicit. This could save storage space. Also, with this approach it is possible to express general information about the perceived world without having to split it into sets of elementary information.

Table 4. Relation Defined Explicitly

<table>
<thead>
<tr>
<th>Relation: EMP(Y,Z)</th>
<th>Name</th>
<th>Salary</th>
<th>Level</th>
</tr>
</thead>
<tbody>
<tr>
<td>John</td>
<td>20K</td>
<td>Unclassified</td>
<td>Secret</td>
</tr>
<tr>
<td>Mary</td>
<td>15K</td>
<td>Secret</td>
<td>Secret</td>
</tr>
<tr>
<td>John</td>
<td>50K</td>
<td>Secret</td>
<td>Secret</td>
</tr>
<tr>
<td>Mary</td>
<td>10K</td>
<td>Secret</td>
<td>Secret</td>
</tr>
<tr>
<td>Jane</td>
<td>70K</td>
<td>Secret</td>
<td>Secret</td>
</tr>
</tbody>
</table>

Examining tables 4, 5, and 6 we see that by implicitly defining the relation SEN-EMP, certain tuples need not be explicitly expressed in the database. This could save storage space. As another example, consider the integrity rule IR2 expressed in table 4 (note that this is in fact the rule IR5 in table 3). This rule ensures that tuples are duplicated at different security levels. In the third approach, such duplication can be eliminated by using this rule as a derivation rule as given in table 7. Note that the derivation rule is in fact the inference rule DASL. However, this rule has to be explicitly stated if new tuples are to be deduced from existing tuples. By using IR2 in table 4 as a derivation rule as shown in table 7, some tuples in the relation EMP need not be explicitly specified.

Negative information is handled by assuming that anything that is not either explicitly or implicitly specified in the database is assumed to be false. However, one has to be careful so as to not introduce wrong information. That is, one has to forbid the derivation of negative information from negative information as negative information can be regarded as information that is not positively sure. That is, the following general law (L) cannot be used as derivation rules:


Note that the above law can be rewritten as:


The negative information → B1, → B2, .... → Bp(1) is included in the derivation of Bp. Since this negative information could be obtained by default, (i.e., the negative information could be obtained by the absence of positive information), one cannot be completely certain of the validity of the negative information. As a result, one cannot be certain of the validity of any
information derived from negative information. If \( p = 1 \) in the general law \( L \), then all information is deduced from positive information.

Table 5. Relation Defined Implicitly

| S1: \((V \times T_1, V \times T_2, Z \times L_1, L_2, EMP(X, Y, Z, L_1, L_2)) \rightarrow T_1 = S_4 \land T_2 = NAME \land T_3 = SALARY\), Unclassified |
| S2: \((V \times T_1, L_2, SEN-EMP(X, T_2)) \rightarrow T = NAME\), Unclassified |
| D1: \((EMP(X, Y, Z_1, L_2, L_3)) \rightarrow Z \leq 40K\), Unclassified |
| D2: \((EMP(X, Y, Z_1, L_2, L_3) \land Y_1 = Y_2 \land Z_1 = 52L)\), Unclassified |
| D3: \((\forall X \in S_4, Y \in NAME, Z \in SALARY, L_1, L_2, L_3, EMP(X, Y, Z_1, L_2, L_3) \land Y_1 = Y_2 \land Z_1 = 52L)\) 
  \[\text{A just less than (L_1, L_2) \rightarrow EMP(X, Y_1, Z_1, L_2, L_3)\},\text{Unclassified}\] |

<table>
<thead>
<tr>
<th>Relation Defined Implicitly</th>
<th>Relation Defined Explicitly and Implicitly</th>
</tr>
</thead>
</table>

Table 6. Relation Defined Explicitly and Implicitly

<table>
<thead>
<tr>
<th>Relation EMP</th>
<th>Relation SEN-EMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1: ((V \times T_1, V \times T_2, T_3, L_1, L_2, EMP(X, Y, Z, L_1, L_2)) \rightarrow T_1 = S_4 \land T_2 = NAME \land T_3 = SALARY), Unclassified</td>
<td></td>
</tr>
<tr>
<td>S2: ((V \times T_1, L_1, L_2, SEN-EMP(X, T_2)) \rightarrow T = NAME), Unclassified</td>
<td></td>
</tr>
<tr>
<td>D1: ((EMP(X, Y, Z_1, L_2, L_3)) \rightarrow Z \leq 40K), Unclassified</td>
<td></td>
</tr>
<tr>
<td>D2: ((EMP(X, Y, Z_1, L_2, L_3) \land Y_1 = Y_2 \land Z_1 = 52L)), Unclassified</td>
<td></td>
</tr>
</tbody>
</table>
| D3: \((\forall X \in S_4, Y \in NAME, Z \in SALARY, L_1, L_2, L_3, EMP(X, Y, Z_1, L_2, L_3) \land Y_1 = Y_2 \land Z_1 = 52L)\) 
  \[\text{A just less than (L_1, L_2) \rightarrow EMP(X, Y_1, Z_1, L_2, L_3)\},\text{Unclassified}\] |

Table 7. Eliminating Duplication Across Security Levels

<table>
<thead>
<tr>
<th>Relation EMP</th>
<th>Relation SEN-EMP</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1: ((V \times T_1, V \times T_2, Z \times L_1, L_2, EMP(X, Y, Z_1, L_2, L_3)) \rightarrow T_1 = S_4 \land T_2 = NAME \land T_3 = SALARY), Unclassified</td>
<td></td>
</tr>
<tr>
<td>S2: ((V \times T_1, L_1, L_2, SEN-EMP(X, T_2)) \rightarrow T = NAME), Unclassified</td>
<td></td>
</tr>
<tr>
<td>D1: ((EMP(X, Y, Z_1, L_2, L_3)) \rightarrow Z \leq 40K), Unclassified</td>
<td></td>
</tr>
</tbody>
</table>
| D2: \((V \times T_1, Z \times L_1, L_2, L_3, EMP(X, Y, Z_1, L_2, L_3) \land Y_1 = Y_2 \land Z_1 = 52L)\) 
  \[\text{A just less than (L_1, L_2) \rightarrow EMP(X, Y_1, Z_1, L_2, L_3)\},\text{Unclassified}\] |

4. CONCLUSION

It has been well established that Mathematical Logic provides a conceptual framework for databases. However, the logic as used to formalize database concepts does not provide a natural framework for multilevel databases. In a multilevel database, the data is assigned different sensitivity levels. Further, it is possible for an entity in the real world to have different database representations at different security levels. These considerations lead to the conclusion that a different logic is needed to formalize multilevel database concepts. In this paper, we have developed a logic we call NTML as a framework for multilevel database management systems. In particular, NTML syntax, semantics, and theories were described.

A contribution of this paper is the use of NTML to formalize the multilevel database concepts. We described three approaches for formalizing database concepts. In the first approach, the multilevel database, the schema, and the general laws (which are the integrity constraints) are taken to be the proper axioms of an NTML theory. The multilevel real world is an interpretation of this theory. Query evaluation in this approach amounts to theorem proving. When the database is updated, a new NTML theory is formed and, therefore, the consistency of the new theory has to be checked. In the second approach, the multilevel database is a model of the NTML theory whose proposed axioms are the general laws and the schema. Query evaluation in this approach amounts to checking the ability to satisfy the formulas against the model. When the database is updated, the model changes. Therefore, the validity of the axioms has to be checked against the new model. In the third approach, some of the general laws are treated as integrity rules, and the others as derivation rules. The multilevel database is a model of the integrity rules. The derivation rules are used to deduce new information.

In addition to the work reported in this paper, we have investigated several issues in the area of logics and multilevel data/knowledge base management systems. These include the use of NTML for element level classification, specifying data dependencies, and the inference problem, development of a
propositional logic for a multilevel environment, essential points towards a logic programming language based on NTML for multilevel data/knowledge base applications, handling negative information in a multilevel environment, techniques for integrity/security constraint processing, extensions to NTML for knowledge based systems, and security issues of a system based on NTML. Details of this investigation are reported in [THUR00b].

ACKNOWLEDGEMENT
We gratefully acknowledge the support of the Department of the Navy (SPAWAR) for sponsoring this work under contract N00014-89-C-0001.

REFERENCES


