

A Model Theoretic Semantics for Multi-Level Secure Deductive Databases*

Hasan M. Jamil

Department of Computer Science
Mississippi State University
jamil@CS.MsState.Edu

Gillian Dobbie[†]

Department of Computer Science
University of Auckland
gill@cs.auckland.ac.nz

Abstract

The impetus for our current research is the need to provide an adequate framework for *belief reasoning* in multi-level secure (MLS) databases. We demonstrate that a prudent application of the concept of *inheritance* in a deductive database setting will help capture the notion of *declarative belief* and belief reasoning in MLS databases in an elegant way. In this paper, we show that these concepts can be captured in a F-logic style declarative query language, called *MultiLog*, for MLS deductive databases for which a model theoretic semantics exists. This development is significant from a database perspective as it now enables us to compute the semantics of MultiLog databases in a bottom-up fashion. The semantics developed here is reminiscent of the stable model semantics of logic programs with negation. We also define a bottom-up procedure to compute unique models of stratified MultiLog databases. Finally, we also establish the equivalence of MultiLog's three logical characterizations – model theory, fixpoint theory and proof theory.

Key Words: MLS databases, security, belief assertion, reasoning, deductive databases, soundness and completeness.

1 Introduction

The multi-level secure (MLS) data model was proposed to overcome the limitations of the traditional authorization scheme that required defining complicated views on a per user basis that essentially limits access to an entire set of columns of a relation in an *all* or *nothing* fashion. Under this protocol, authorization at the individual data level could not be defined easily. It was shown that a more flexible and fine grain, yet effective, authorization protocol for increased sharing of knowledge could be easily provided by the MLS data model [3, 13, 6]. Hence, several algebra and SQL based languages were developed to manipulate and query MLS databases. Recently, at least two logic based query languages for MLS databases were proposed [2, 8] to extend the functionalities of relational MLS languages and to overcome their proven limitations. The needs and merits of a deductive metaphor of the MLS model is eloquently discussed in [16]. However, the paucity of attempts aimed at developing a logical characterization for MLS models suggests that MLS deductive databases are really at their embryonic stage. While several proposals addressed the general issue of authorization in a deductive framework [11, 1, 5, 15], to our

* Research supported in part by a grant from the National Science Foundation.

[†] Research completed while the author was visiting Mississippi State University in October 2000, while she was on sabbatical at the National University of Singapore.

knowledge, only Cuppens [2] and Jamil [8] addressed the issue of querying MLS deductive databases.

The impetus for MultiLog was to (i) develop a model and a query language which can guard against an additional security breach, called the surprise stories [8], (ii) provide linguistic instruments for capturing multiple belief modes, (iii) allow reasoning and recursion, (iv) support user defined belief modes making it possible to tailor the user view as needed, and finally (v) enable belief speculation and belief reasoning through ad hoc belief querying. MultiLog’s syntax is reminiscent of F-logic [12] but has several differences. The need for these extensions as well as the shortcomings of the MLS data model have been established in [2, 10, 8]. For the sake of conciseness we do not discuss the basic MLS data model in this paper. Interested readers may refer to [6] for an introductory reading.

Our goals for this paper are two-fold: (i) to develop a direct Herbrand semantics for a definite Horn clause fragment of MultiLog by defining a model theory and a fixpoint theory, and (ii) to establish equivalence of its three characterizations – proof theory (discussed in [8]), model theory and fixpoint theory. We make a critical observation that user views of the MLS databases at different security levels closely resemble the notion of *inheritance* in object oriented systems albeit in a slightly elaborate fashion. We utilize this connection and exploit our experience in dealing with inheritance in logic based systems [7, 4]. The so called *belief function* introduced in [8], which we adapt in this paper for defining the notion of belief in Herbrand sets, is an extension of the inheritability function discussed in [7]. Through the equivalence of the different logical characterizations, we establish the fact that MultiLog’s unique features and modeling capabilities, some of which are non-monotonic in nature, do not compromise the soundness and completeness of the language. This development is significant from a theoretical perspective, as it gives insight into the understanding of the logical behavior and mathematical foundations of the language.

The paper is organized as follows. An overview of the syntax is presented in section 2. Then the declarative semantics is given in section 3 by presenting an Herbrand semantics in section 3.2, and a corresponding fixpoint semantics in section 3.3. The logical equivalence of all three characterizations is established in section 3.3 before we conclude in section 4. We have omitted all proofs due to lack of space. The proofs can be found in the technical report in [9].

2 Overview of MultiLog Language

The language \mathcal{L} of MultiLog is a 7-tuple $\langle \mathcal{P}, \mathcal{F}, \mathbf{A}, \mathcal{V}, \mathcal{S}, \preceq, \mu \rangle$ where (i) \mathcal{P} is an infinite set of predicate names, (ii) \mathcal{F} is an infinite set of constants including the symbol `null`, denoted \perp , (iii) \mathbf{A} is a finite set of attribute names, (iv) \mathcal{V} is a denumerable set of variable names, (v) \mathcal{S} is a finite non-empty set of labels which includes a special symbol s_{\perp} intended to denote the security levels in our language, (vi) \preceq is a partial order on the symbols in \mathcal{S} that captures the idea of the hierarchy of security levels, and finally (for this paper) (vii) $\mu = \{cau, opt, fir\}$ is a finite set of symbols for belief modes. The symbols in \mathcal{L} are pairwise disjoint. The terms \mathcal{T} of \mathcal{L} are the set $\mathcal{F} \cup \mathcal{V}$.

Formulas and Databases: There are five types of atoms in our language: m-, b-, p-, l- and h-atoms¹.

- MLS atoms or m-atoms: Let p be a predicate symbol in \mathcal{P} of arity n - denoted $p/n, a$

¹Throughout the paper we use uppercase letters for variables and lowercase letters for constants. We use bold lowercase letters to denote arbitrary terms. While calligraphic uppercase letters are used to denote arbitrary formulas, uppercase italic letters A, B, F, G , etc. are used to denote ground formulas.

is an attribute name in \mathcal{A} , \mathbf{v} and \mathbf{k} are terms in \mathcal{T} , and \mathbf{s} and \mathbf{c} are symbols in $\mathcal{S} \cup \mathcal{V}$.

Then $\mathbf{s}[p(\mathbf{k} : a \xrightarrow{\mathbf{c}} \mathbf{v})]$ is an m-atom. Intuitively, an m-atom represents an attribute of a tuple as in an MLS relational database counterpart where a is an attribute name, \mathbf{v} is a value, \mathbf{k} is the object key and \mathbf{s} and \mathbf{c} are security labels. The label \mathbf{s} denotes the security level of the predicate p and mimics the tuple classification TC in the MLS relational model. The label \mathbf{c} mimics the attribute classification in the MLS relational model.

- Believed atoms or b-atoms: Let $\mathbf{s}[p(\mathbf{k} : a \xrightarrow{\mathbf{c}} \mathbf{v})]$ be an m-atom and $m \in \mu$ be a mode of belief by a rational agent. Then $\mathbf{s}[p(\mathbf{k} : a \xrightarrow{\mathbf{c}} \mathbf{v})] \ll m$ is a b-atom. Intuitively, a b-atom says that a rational agent believes $p(\mathbf{k} : a \xrightarrow{\mathbf{c}} \mathbf{v})$ at level \mathbf{s} in a mode m .
- Predicates or p-atoms: If p is a predicate symbol in \mathcal{P} of arity k - denoted p/k , and $\mathbf{t}_1, \dots, \mathbf{t}_k$ are terms in \mathcal{T} , then $p(\mathbf{t}_1, \dots, \mathbf{t}_k)$ is a p-atom. The sense of p-atoms is identical to predicates in classical logic.
- Level atoms or l-atoms: Let $level/1$ be a distinguished predicate symbol in \mathcal{P} , and \mathbf{s} be a symbol in $\mathcal{S} \cup \mathcal{V}$. Then $level(\mathbf{s})$ is an l-atom. An l-atom declares the existence of a security level in a database Δ .
- Hierarchy atoms or h-atoms: Let $order/2$ be another distinguished predicate symbol in \mathcal{P} , and \mathbf{l} and \mathbf{h} be two symbols in $\mathcal{S} \cup \mathcal{V}$. Then $order(\mathbf{l}, \mathbf{h})$ is an h-atom. Intuitively an h-atom asserts that the security level \mathbf{l} is lower than \mathbf{h} and that there are no other \mathbf{i} such that $order(\mathbf{l}, \mathbf{i})$ and $order(\mathbf{i}, \mathbf{h})$ hold.

Formulas of \mathcal{L} are defined as usual. A literal is either an atom (\mathcal{A}) or the negation of an atom ($\neg \mathcal{A}$). Following the custom in logic programming, we only consider the definite (Horn) clause fragment of our language. A clause in \mathcal{L} is an expression of the form $\mathcal{A} \leftarrow \mathcal{B}_1, \dots, \mathcal{B}_m$ such that \mathcal{A} and \mathcal{B}_i s are atoms of \mathcal{L} . If the consequent of a clause is an m-atom, we call the clause an m-clause. Similarly, we define p-, l- and h-clauses. We, however, do not have b-clauses as we do not allow b-atoms to appear in the consequent.

Example 2.1 Consider the MLS relation $Mission(Starship, C_1, Objective, C_2, Destination, C_3, TC)$ in Figure 1 adapted from [8]. In this example, and also throughout this paper, we consider only four security levels for simplicity. Namely, s_{\perp} , u , c , and s with a total order $s_{\perp} < u < c < s$. That is, we have in our database the atoms $order(s_{\perp}, u)$, $order(u, c)$, and $order(c, s)$. The attributes C_1 , C_2 , and C_3 respectively denote the attribute classifications for Starship, Objective and Destination, whereas TC denotes the tuple classification for $Mission$.

Tid	Starship	C_1	Objective	C_2	Destination	C_3	TC
t_1	Voyager	U	Spying	S	Mars	U	S
t_2	Phantom	U	Spying	S	Omega	U	S
t_3	Phantom	C	Supply	C	Venus	S	S
t_4	Atlantis	U	Diplomacy	U	Vulcan	U	C
t_5	Voyager	U	Training	U	Mars	U	U

Figure 1: MLS relation $Mission(Starship, C_1, Objective, C_2, Destination, C_3, TC)$.

In this scheme, Starship is the *apparent primary key* (AK), C_1 is the classification of the apparent primary key (C_{AK}), and for the attributes Objective and Destination respectively, the functional dependencies $\{Starship, C_{AK}, C_2\} \rightarrow Objective$, and $\{Starship, C_{AK}, C_3\} \rightarrow Destination$ hold where $C_{AK} = C_1$. Hence, the *primary key* of the relation is $\{Starship, C_{AK}, C_2, C_3\} = \{Starship, C_1, C_2, C_3\}$.

Now consider tuple t_2 in *Mission*. It is easy to see that a predicate representation of t_2 would be $mission(phantom, u, spying, s, omega, u, s)$. In MultiLog we represent t_2 as $s[mission(phantom : starship \xrightarrow{u} phantom)] \wedge s[mission(phantom : objective \xrightarrow{s} spying)] \wedge s[mission(phantom : destination \xrightarrow{u} omega)]$, i.e., as a conjunction of m-atoms of the form $\text{TC}[\text{rel}(\text{app_key}:\text{att_name} \xrightarrow{\text{att_class}} \text{att_val})]$. A more compact representation is an *m-molecule*, $s[mission(phantom : starship \xrightarrow{u} phantom, objective \xrightarrow{s} spying, destination \xrightarrow{u} omega)]$. An m-molecule is a syntactic variant of conjunctions of m-atoms. In general, for any classical predicate representation $p(\mathbf{k}, a_1, \mathbf{v}_1, \mathbf{c}_1, \dots, a_k, \mathbf{v}_k, \mathbf{c}_k, \mathbf{s})$ corresponding to an MLS tuple over the scheme $p(K, a_1, C_1, \dots, a_k, C_k, TC)$ where $K \subseteq \{a_1, \dots, a_k\}$ is the apparent primary key, the m-molecular representation can be written as $s[p(\mathbf{k} : a_1 \xrightarrow{c_1} \mathbf{v}_1, \dots, a_k \xrightarrow{c_k} \mathbf{v}_k)]$. Notice that, ordinarily an MLS tuple will be represented in classical first-order logic as a predicate of the form $p(\mathbf{v}_1, \mathbf{c}_1, \dots, \mathbf{v}_k, \mathbf{c}_k, \mathbf{s})$. The only difference with MLS tuples is, of course, that we include attribute names a_i s and the apparent key \mathbf{k} in our atoms/molecules as just shown. A similar approach was also taken in [12]. \square

Definition 2.1 (Depth of Atoms) Let \mathcal{A} be an atom in \mathcal{L} . If \mathcal{A} is p-, l- or h-atom then the depth of \mathcal{A} , written $depth(\mathcal{A})$ is s_\perp . It is l^2 , otherwise (i.e., m- or b-atoms).

Intuitively, s_\perp has the lowest security classification. This is a technical requirement and guarantees that all the l-, h- and p-atoms defined in s_\perp are visible to all the higher levels.

Definition 2.2 (Dependency Graph) Let Cl be a clause of the form $\mathcal{A} \leftarrow \mathcal{B}_1, \dots, \mathcal{B}_m$. Let \leftarrow denote the binary relationship *depends on*. For Cl , we say that \mathcal{A} depends on $\mathcal{B}_1, \dots, \mathcal{B}_m$, denoted $\mathcal{A} \leftarrow \mathcal{B}_1, \dots, \mathcal{A} \leftarrow \mathcal{B}_m$. The transitive closure of the relation \leftarrow with respect to \mathcal{A} is called the *dependency graph* of \mathcal{A} . \square

Definition 2.3 (Databases and Queries) A *database* Δ , or equivalently a program \mathbf{P} , in MultiLog is an expression of the form $\langle \mathcal{A}, \Sigma, \Pi, \mathcal{Q} \rangle$, where (i) \mathcal{A} is a set of l- and h-clauses (possibly empty) defining the security levels and inducing a partial order on the security levels (symbols in $\mathcal{S} \cup \mathcal{V}$), (ii) Σ is a set of m-clauses defining the secured data component of Δ , (iii) Π is a set of p-clauses (possibly empty), and finally (iv) \mathcal{Q} is a set of clauses of the form $\leftarrow \mathcal{B}_1, \dots, \mathcal{B}_m$, called the queries. For every clause $Cl = \mathcal{A} \leftarrow \mathcal{G} \in \mathcal{A}$, we require that the dependency graph of \mathcal{A} does not contain atoms other than h- or l-atoms³. \square

Similar to MLS relational models, we require that MultiLog database m-molecules satisfy several integrity constraints. First, we require that for every m-molecule there is a key attribute AK for which the value is \mathbf{k} . Hence, there must be an m-atom of the form $s[p(\mathbf{k} : a \xrightarrow{c} \mathbf{k})]$. That is, for every m-atom of the form $s[p(\mathbf{k} : b \xrightarrow{d} \mathbf{v})]$ in a database P^4 , we also have $s[p(\mathbf{k} : a \xrightarrow{c} \mathbf{k})]$. For such atoms, \mathbf{k} is identified as AK , \mathbf{c} as c_{AK} , \mathbf{s} as TC , and for all other atoms for which \mathbf{k} is the key, a is identified as A_i , \mathbf{c} as C_i and \mathbf{v} as A_i in a manner similar to Jajodia and Sandhu [6].

Definition 2.4 (Closure of Databases) Let $\Delta = \langle \mathcal{A}, \Sigma, \Pi, \mathcal{Q} \rangle$ be a MultiLog database. The *closure* $\Delta^* = \langle \mathcal{A}^*, \Sigma^*, \Pi^*, \mathcal{Q}^* \rangle$ of Δ is the smallest set of clauses satisfying the conditions below:

²We are assuming that the atom \mathcal{A} is of the form $l[p(\mathbf{k} : a \xrightarrow{c} \mathbf{v})]$ or $l[p(\mathbf{k} : a \xrightarrow{c} \mathbf{v})] \ll m$.

³Intuitively, it means that the ground closure of \mathcal{A} does not depend on the clauses defined in other components of Δ .

⁴In fact in the model of P defined shortly.

- $\Lambda \subseteq \Lambda^*, \Sigma^* = \Sigma, \Pi^* = \Pi, \mathcal{Q}^* = \mathcal{Q}$.
- $L \preceq L \leftarrow level(L) \in \Lambda^*$.
- $L \preceq L \leftarrow order(L, H) \in \Lambda^*$ and $H \preceq H \leftarrow order(L, H) \in \Lambda^*$.
- $L \preceq H \leftarrow L \preceq H', order(H', H) \in \Lambda^*$. □

In the remainder of this paper, unless stated otherwise, we consider only closed databases.

Definition 2.5 (Level of Databases) Let Δ be a consistent database, and u is a user with a clearance \bar{u} . The database Δ is in level \bar{u} , denoted $\langle \Delta, \bar{u} \rangle$, if the user u with clearance \bar{u} accesses Δ . □

Example 2.2 The *Mission* relation in Figure 1 is expressed in MultiLog as the level s database $\langle mission, s \rangle$ below.

$$\Lambda_{D_1} := \left| \begin{array}{l} level(u). \quad level(c). \quad level(s). \quad order(u, c). \quad order(c, s). \end{array} \right.$$

$$\Sigma_{D_1} := \left[\begin{array}{l} u[mission(voyager : starship \xrightarrow{u} voyager, objective \xrightarrow{u} training, destination \xrightarrow{u} mars)]. \\ c[mission(atlantis : starship \xrightarrow{u} atlantis, objective \xrightarrow{u} diplomacy, destination \xrightarrow{u} vulcan)]. \\ s[mission(phantom : starship \xrightarrow{c} phantom, objective \xrightarrow{c} supply, destination \xrightarrow{s} venus)]. \\ s[mission(phantom : starship \xrightarrow{u} phantom, objective \xrightarrow{s} spying, destination \xrightarrow{u} omega)]. \\ s[mission(voyager : starship \xrightarrow{u} voyager, objective \xrightarrow{s} spying, destination \xrightarrow{u} mars)]. \end{array} \right.$$

Belief modes give the user the choice to reason and theorize about the beliefs. In this paper, we describe three belief modes, and it is possible for users to define further belief modes. We now informally introduce the three belief modes.

If the user takes the *firm* view of the data, they believe the data visible at their security level only is correct and believable data. Consider the query $\leftarrow s[mission(V : W \xrightarrow{X} Y)] \ll fir$ that asks for the firm s level view of *mission*. The answer returned is as follows:

$$\begin{array}{l} s[mission(phantom : starship \xrightarrow{c} phantom, objective \xrightarrow{c} supply, destination \xrightarrow{s} venus)]. \\ s[mission(phantom : starship \xrightarrow{u} phantom, objective \xrightarrow{s} spying, destination \xrightarrow{u} omega)]. \\ s[mission(voyager : starship \xrightarrow{u} voyager, objective \xrightarrow{s} spying, destination \xrightarrow{u} mars)]. \end{array}$$

If the user takes the *optimistic* view, they believe what is created at all security levels is correct and believable data. Hence the following is the response if we consider the query $\leftarrow s[mission(V : W \xrightarrow{X} Y)] \ll opt$ that asks for the optimistic s level view of *mission*:

$$\begin{array}{l} s[mission(voyager : starship \xrightarrow{u} voyager, objective \xrightarrow{u} training, destination \xrightarrow{u} mars)]. \\ s[mission(atlantis : starship \xrightarrow{u} atlantis, objective \xrightarrow{u} diplomacy, destination \xrightarrow{u} vulcan)]. \\ s[mission(phantom : starship \xrightarrow{c} phantom, objective \xrightarrow{c} supply, destination \xrightarrow{s} venus)]. \\ s[mission(phantom : starship \xrightarrow{u} phantom, objective \xrightarrow{s} spying, destination \xrightarrow{u} omega)]. \\ s[mission(voyager : starship \xrightarrow{u} voyager, objective \xrightarrow{s} spying, destination \xrightarrow{u} mars)]. \end{array}$$

If the user takes the *cautious* view, the *visible* information at a given level that has the highest security classification is retained and the other data is filtered out. The assumption is that the higher level information is more reliable and takes priority over the lower level information. So, the query $\leftarrow s[mission(V : W \xrightarrow{X} Y)] \ll cau$ that asks for the cautious s level view of *mission* returns:

$$\begin{array}{l} s[mission(atlantis : starship \xrightarrow{u} atlantis, objective \xrightarrow{u} diplomacy, destination \xrightarrow{u} vulcan)]. \\ s[mission(phantom : starship \xrightarrow{c} phantom, objective \xrightarrow{s} spying, destination \xrightarrow{s} venus)]. \\ s[mission(voyager : starship \xrightarrow{u} voyager, objective \xrightarrow{s} spying, destination \xrightarrow{u} mars)]. \end{array}$$

In this example, we have demonstrated how belief atoms are used in queries and the results of the different belief modes. The use of belief atoms in the right hand side of a rule follows from this. \square

3 Declarative Semantics of MultiLog

The Herbrand semantics of MultiLog databases is defined in terms of a composite set-theoretic structure which provides a model for each of the security levels in \mathcal{L} (represented by each of the symbols in \mathcal{S}), including the level s_{\perp} – the system level which is not part of any database universe. In other words, each model in the composite structure interprets formulas pertaining to the corresponding levels in the security hierarchy. The notion of “belief” in such a structure is then captured using a function level semantics over the sets in the Herbrand structure, not as a set membership, as explained shortly.

These structures are similar to standard interpretations and models in classical logic. The difference between the structures for standard predicate logic programs and ours is that while we simultaneously interpret several program fragments that are interrelated through *inter structure entailment* relations in the composite structure (similar to message passing in object oriented systems), in the predicate logic case we interpret a single program with only standard entailment due to implication and conjunction. Unlike Miller’s Kripke-like structures [14], we are then able to define the notion of satisfaction in a standard way as a “set membership”.

The notion of Herbrand universe \mathcal{U} and base \mathcal{H}^5 is defined similar to the classical case. Formally, an *Herbrand structure* \mathbf{H} of \mathcal{L} is a tuple $\langle H(s) : s \in \mathcal{S} \rangle$ such that $H(s) \subseteq \mathcal{H}$ for every $s \in \mathcal{S}$. When $s \neq s_{\perp}$, $H(s)$ contains only m-atoms, otherwise $H(s_{\perp})$ contains only p-, l- and h-atoms. Intuitively, every $H(s)$, $s \neq s_{\perp}$ in \mathbf{H} interprets the associated data items belonging to the level s as ground m-atoms that are true with respect to level s in \mathbf{H} . To make a distinction between an interpretation corresponding to a security level and the interpretation structures for our language \mathcal{L} , we henceforth call them interpretations and T-interpretations respectively, since the latter is actually a tuple of simple interpretations or sets.

3.1 Formula Satisfaction and Models

Before we proceed to define the notion of truth and satisfaction in our language, we need to cast the belief function β defined in [8] in terms of sets of ground formulas in MultiLog.

Definition 3.1 (Belief in Herbrand Sets) Let S be an arbitrary set of m-atoms, $A = l[p(k : a \xrightarrow{c} v)]$ be a ground m-atom, and m be a belief mode in $\mu = \{fir, cau, opt\}$. Then, the belief function $\beta : \mathcal{P}(\mathbf{S}) \times \mathcal{S} \times \mu \rightarrow \mathcal{P}(\mathbf{S})$, where \mathbf{S} is a set of all possible m-atoms, is defined as follows:

$$\beta(S, l, m) = \left\{ \begin{array}{l} l[p(k : a \xrightarrow{c} v)] \end{array} \right\} \quad \left| \quad \begin{array}{l} \text{One of the following conditions hold:} \\ - m = fir \text{ and } l[p(k : a \xrightarrow{c} v)] \in S \\ - m = cau \text{ and } \exists l'[p(k : a \xrightarrow{c} v)] \in S, l' \preceq l, \text{ and} \\ \quad \neg \exists l''[p(k : a \xrightarrow{c'} v')] \in S, l'' \preceq l, \text{ and } c \prec c'. \\ - m = opt \text{ and } l'[p(k : a \xrightarrow{c} v)] \in S \text{ and } l' \preceq l. \end{array} \quad \square$$

⁵Note that b-atoms are not part of the Herbrand base \mathcal{H} as their satisfaction really depends on the interpretations and an entailment function β . But note that the data components of b-atoms, the corresponding m-atoms, are.

3.2 Herbrand Interpretations

We now define satisfaction of formulas in an Herbrand structure in a manner similar to the classical case. The major difference is that we now have to define satisfaction for formulas with respect to a T-interpretation (across a set of interpretations) and a user level. The definition of satisfaction must then correctly capture the meaning of secured data at a given level and the belief asserted by such data along the security hierarchy as intended by the m-atoms and b-atoms.

Definition 3.2 (Satisfaction of Formulas) Let \mathbf{H} be a T-interpretation, \bar{u} be a user clearance level, $H(i)$ be any arbitrary interpretation⁶ in \mathbf{H} where i is a security level in \mathbf{H} , and let n be the number of such security levels⁷. Let A and B denote ground atomic formulas, and F denote any arbitrary ground formula. Then, the satisfaction of ground formulas with respect to $H(i)$, denoted $H(i) \models_{\mathbf{H}, \bar{u}} F$, is defined as follows:

- (1) $H(s) \models_{\mathbf{H}, \bar{u}} A \iff A \in H(s)$ where $depth(A) = s$ and $s \leq \bar{u}$
- (2) $H(s) \models_{\mathbf{H}, \bar{u}} A \iff H(o) \models_{\mathbf{H}, \bar{u}} A$ where $depth(A) = o$ and $s \neq o$
- (3) $H(i) \models_{\mathbf{H}, \bar{u}} A \leftarrow B_1, \dots, B_m \iff H(i) \models_{\mathbf{H}, \bar{u}} B_g, g = 1, \dots, m \implies H(i) \models_{\mathbf{H}, \bar{u}} A$
- (4) $H(i) \models_{\mathbf{H}, \bar{u}} l[p(k : a \xrightarrow{c} v) \ll m \iff H' \models_{\mathbf{H}, \bar{u}} l[p(k : a \xrightarrow{c} v)]$ where
 $H' = \beta(\bigcup_{j=1}^n H(s_j), \bar{u}, m)$ such that
 $\forall j, s_j \leq l$ and $l \leq \bar{u}$

Finally, we say that $\mathbf{H} \models_{\bar{u}} A$ if and only if $H(l) \models_{\mathbf{H}, \bar{u}} A$, where $l = depth(A)$. \square

There are several important observations that one can make in the way satisfaction is defined in our Herbrand structures. First of all, every interpretation in \mathbf{H} assigns meaning to data items at a given security level. The interpretation is indexed by the label returned by the *depth* function in definition 2.1. The satisfiability of a formula does not depend *only* upon the membership of the formula in an interpretation as in its classical logic counterpart. In addition, satisfiability requires a formula to be visible, i.e., the visibility at a clearance level, which is a parameter in the definition of satisfaction. The length of the T-interpretation depends on the set of symbols in \mathcal{S} in a given database. Note that if \mathcal{S} contains only s_{\perp} , the T-interpretation essentially degenerates into a classical Herbrand interpretation as \mathcal{S} is non-empty and always includes s_{\perp} .

The way in which we have defined our interpretations and satisfaction, rules out the possibility of having interpretation $H(s)$ satisfying a formula of the form $l[p(k : a \xrightarrow{c} v)]$ such that $l \neq s$. But it may still satisfy, for example, a formula of the form $s[p(k : a \xrightarrow{c} v)]$ such that $s \leq c$. We thus require our candidate interpretations to satisfy additional consistency requirements as defined in [8] to be a model of a database. Note that these conditions are reminiscent and adaptation of the core integrity properties defined in Jajodia and Sandhu [6] with some significant differences. In particular, notice that except for the entity, null and polyinstantiation integrity constraints (core integrity properties of MLS relational model [6]), the closure and hierarchy integrity are the added requirements.

Recall that, using the cautious belief mode, if a fact isn't believed at a particular security level, then a fact at a lower level is believed. This concept of specificity is intimately related to the notion of inheritance and overriding in object-oriented systems. We attempt to clarify this through the example below and the definition that follows formalizes the idea of the so called *more specific* atoms.

⁶Note again that an interpretation is a Herbrand set, and a T-interpretation is a tuple of such Herbrand sets.

⁷Notice that in this definition the symbols s, i, o, u, l and s_j , as well as \bar{u} , represent security levels in \mathbf{H} .

Example 3.1 Consider the level s database $\langle D_1, s \rangle$ shown below.

$$\begin{array}{l}
\Lambda_{D_1} := \left\{ \begin{array}{l} r_1 : \text{level}(u). \\ r_2 : \text{level}(c). \\ r_3 : \text{level}(s). \\ r_4 : \text{order}(u, c). \\ r_5 : \text{order}(c, s). \end{array} \right. \quad \Sigma_{D_1} := \left\{ \begin{array}{l} r_6 : c[p(k : a \xrightarrow{u} v)]. \\ r_7 : c[p(k : a \xrightarrow{c} t)] \leftarrow p(j). \\ r_8 : s[p(K : A \xrightarrow{c} V)] \leftarrow \\ \quad c[p(K : A \xrightarrow{c} V)] \ll \text{cau}. \end{array} \right. \\
\mathcal{Q}_{D_1} := \left\{ r_{10} : ? s[p(k : a \xrightarrow{u} v)] \right.
\end{array}$$

If $p(j)$ is not true, then $s[p(k : a \xrightarrow{u} v)]$ is true. But if $p(j)$ is true, then $s[p(k : a \xrightarrow{c} t)]$ is true. We say that the atom $c[p(k : a \xrightarrow{c} t)]$ is more specific than the atom $c[p(k : a \xrightarrow{u} v)]$ because $u < c$ (because of $\text{order}(u, c) \in \Lambda_{D_1}$). \square

Definition 3.3 (More Specific) Let $\langle \Delta, \bar{u} \rangle$ be level \bar{u} database, where $A = l[r(k : a \xrightarrow{s} v)]$ and $A' = l[r(k : a \xrightarrow{s'} v')]$, where $s' < s$. We say A is *more specific* than A' .

Example 3.2 The m-atom $c[p(k : a \xrightarrow{c} t)]$ is more specific than $c[p(k : a \xrightarrow{u} v)]$ because $u < c$. We can't say anything about the specificity of $s[p(k : a \xrightarrow{c} t)]$ and $c[p(k : a \xrightarrow{u} v)]$.

Definition 3.4 (Consistent Database) A database $\langle \Delta, \bar{u} \rangle$ is consistent if there exists a mapping μ from the set of ground atoms to the set of non-negative integers, such that, for every atom in every ground instance $C\theta$ for every clause C in $\langle \Delta, \bar{u} \rangle$,

1. $\mu(A') \leq \mu(A)$, where A' is an atom in the body of the clause instance $C\theta$, and A is the head of the clause instance $C\theta$,
2. $\mu(B) < \mu(A')$, for every ground clause $C'\theta'$ where B is a cautious b-atom in the body of $C\theta$ and B' is a cautious b-atom in the body of $C'\theta'$, B is more specific than B' , and A' is the head of $C'\theta'$.
3. $\mu(A) < \mu(A')$, for every ground clause $C'\theta'$ where B is a cautious b-atom in the body of $C\theta$ and B' is a cautious b-atom in the body of $C'\theta'$, B is more specific than B' , A is the head of $C\theta$, and A' is the head of $C'\theta'$.
4. $\mu(A') < \mu(A)$, for every ground clause $C'\theta'$ where B is a cautious b-atom in the body of $C\theta$, A is the head of $C\theta$, A' is the head of $C'\theta'$ and A' is more specific than B . \square

Example 3.3 A database with a rule $s[p(k : a \xrightarrow{c} t)] \leftarrow s[p(k : a \xrightarrow{u} v)] \ll \text{cau}$ is not consistent. Using the definition of a consistent database, we find $s[p(k : a \xrightarrow{c} t)] < s[p(k : a \xrightarrow{c} t)]$ and there is no mapping from this to the set of non-negative integers. Intuitively, this is not a consistent database, since if the body of the rule is satisfied, then the head of the rule is satisfied, and the body is no longer satisfied.

Example 3.4 Consider the following ground level s database $\langle D_2, s \rangle$ for simplicity as a further example.

$$\begin{array}{l}
\Lambda_{D_2} := \left\{ \begin{array}{l} r_1 : \text{level}(u). \\ r_2 : \text{level}(c). \\ r_3 : \text{level}(s). \\ r_4 : \text{order}(u, c). \\ r_5 : \text{order}(c, s). \end{array} \right. \quad \Sigma_{D_2} := \left\{ \begin{array}{l} r_6 : c[p(k : a \xrightarrow{u} v)]. \\ r_7 : c[p(k : a \xrightarrow{c} t)] \leftarrow p(j). \\ r_8 : s[p(k : a \xrightarrow{c} t)] \leftarrow c[p(k : a \xrightarrow{c} t)] \ll \text{cau}. \\ r_9 : s[p(k : a \xrightarrow{u} v)] \leftarrow c[p(k : a \xrightarrow{u} v)] \ll \text{cau}. \end{array} \right.
\end{array}$$

Based on the definition for consistent databases, we get:

$$\begin{aligned}
p(j) &\leq c[p(k : a \xrightarrow{c} t)] && \text{(rule 1)} \\
c[p(k : a \xrightarrow{c} t)] &\leq s[p(k : a \xrightarrow{c} t)] && \text{(rule 1)} \\
c[p(k : a \xrightarrow{u} v)] &\leq s[p(k : a \xrightarrow{u} v)] && \text{(rule 1)} \\
c[p(k : a \xrightarrow{c} t)] &< s[p(k : a \xrightarrow{u} v)] && \text{(rule 2)} \\
s[p(k : a \xrightarrow{c} t)] &< s[p(k : a \xrightarrow{u} v)] && \text{(rules 3 and 4)}
\end{aligned}$$

This goes to show that there exists a mapping from the set of ground atoms to the set of non-negative integers, so database $\langle D_2, s \rangle$ is consistent.

Consider if the above database is altered in the following way: rules 8 and 9 are removed and replaced by $s[p(k : a \xrightarrow{c} t)] \leftarrow s[p(k : a \xrightarrow{u} v)] \ll cau$, then by rule 4, $s[p(k : a \xrightarrow{c} t)] < s[p(k : a \xrightarrow{c} t)]$ and there is no mapping from the set of ground atoms to the set of non-negative integers, so the database is not consistent. Intuitively, what does this mean? If the body of the clause is true, then the head of the clause becomes true and as a consequence the body of the rule is no longer true. This database has no meaning. \square

We now define the notion of a model for databases in MultiLog based on the notion of satisfaction and consistent interpretations.

Definition 3.5 (Consistent T-models) Let $\mathbf{H} = \langle H(s) : s \in \mathcal{S} \rangle$ be a Herbrand structure, or a T-interpretation, $\langle \Delta, \bar{u} \rangle$ be a database at level \bar{u} , and $|\langle \Delta, \bar{u} \rangle|$ be the Herbrand instantiation of $\langle \Delta, \bar{u} \rangle$. Then \mathbf{H} is a T-model for $\langle \Delta, \bar{u} \rangle$ iff for every clause cl in $|\langle \Delta, \bar{u} \rangle|$, $\mathbf{H} \models_{\bar{u}} cl$. \mathbf{H} is a consistent T-model for $\langle \Delta, \bar{u} \rangle$ iff \mathbf{H} is a T-model for $\langle \Delta, \bar{u} \rangle$ and is consistent as a structure. \square

Example 3.5 Consider the database D_1 in example 3.1, this time at level c (as opposed to level s as shown before), i.e., $\langle D_1, c \rangle$. For the database $\langle D_1, c \rangle$, let the T-model be M_1 , as shown below:

$$M_1 = \left\langle \underbrace{\left\{ \begin{array}{l} level(u), level(c), level(s), \\ order(u, c), order(c, s), p(j) \end{array} \right\}}_{M_1(s_{\perp})}, \underbrace{\emptyset}_{M_1(u)}, \underbrace{\left\{ \begin{array}{l} c[p(k : a \xrightarrow{u} v)], \\ c[p(k : a \xrightarrow{c} t)] \end{array} \right\}}_{M_1(c)}, \underbrace{\{s[p(k : a \xrightarrow{c} t)]\}}_{M_1(s)} \right\rangle$$

In the T-model M_1 above, the first set in the interpretation belongs to level s_{\perp} , i.e., $M_1(s_{\perp})$. The second set belongs to u , the third to level c , and the last to s . Now, several observations can be made here. Note that the database is at level c . Also note that $s[p(k : a \xrightarrow{c} t)] \in M_1, M_1(s)$ to be precise. Still $M_1 \not\models_c s[p(k : a \xrightarrow{c} t)]$. This is because $M_1(s) \not\models_{M_1, c} s[p(k : a \xrightarrow{c} t)]$ as it does not satisfy Condition 1 in the definition of satisfaction 3.2, i.e., $s \not\leq c$. But $M_1 \models_c c[p(k : a \xrightarrow{u} v)]$, and also $M_1 \models_c c[p(k : a \xrightarrow{c} t)]$. Yet, it is interesting to verify that $M_1 \models_c c[p(k : a \xrightarrow{c} t)] \ll cau$ but $M_1 \not\models_c c[p(k : a \xrightarrow{u} v)] \ll cau$. This observation follows from the definition of cautious belief in Herbrand sets, i.e., definition 3.1. \square

But not every consistent T-model is “intended”, a term which will be defined shortly. Recall that the satisfaction of b-atoms depends on the belief function β which makes use of the Herbrand sets in \mathbf{H} . Also recall that the set computed by β depends on the elements in the Herbrand sets corresponding to security levels dominated by \bar{u} . While the satisfaction of b-atoms is not affected by elements not required for a structure to be a T-model for a database $\langle \Delta, \bar{u} \rangle$, it potentially affects the beliefs of users as unwanted models may result. The following example helps clarify this point.

Example 3.6 Consider a level s database $\langle D_3, s \rangle$. Assume that database D_3 is derived from database D_1 of example 3.1 by replacing rule r_6 with $u[p(k : a \xrightarrow{u} v)]$, rule r_8 by $s[p(k : a \xrightarrow{u} v)] \leftarrow c[p(k : a \xrightarrow{u} v)] \ll cau$, and finally by deleting rule r_9 and adding two rules $r_{11} : p(X) \leftarrow r(X)$ and $r_{12} : r(X) \leftarrow q(X)$. Now for database $\langle D_3, s \rangle$ as defined, the intended model M_3 may be identified as follows:

$$M_3 = \left\{ \underbrace{\left\{ \begin{array}{l} level(u), level(c), level(s), \\ order(u, c), order(c, s) \end{array} \right\}}_{M_3(s_{\perp})}, \underbrace{\{u[p(k : a \xrightarrow{u} v)]\}}_{M_3(u)}, \underbrace{\emptyset}_{M_3(c)}, \underbrace{\{s[p(k : a \xrightarrow{u} v)]\}}_{M_3(s)} \right\}$$

However, it is easy to verify that M'_3 or M''_3 below are not intended although they are T-models for D_3 at level s .

$$M'_3 = \left\{ \underbrace{\left\{ \begin{array}{l} level(u), level(c), level(s), \\ order(u, c), order(c, s) \end{array} \right\}}_{M'_3(s_{\perp})}, \underbrace{\{u[p(k : a \xrightarrow{u} v)]\}}_{M'_3(u)}, \underbrace{\{c[p(k : a \xrightarrow{c} t)]\}}_{M'_3(c)}, \underbrace{\emptyset}_{M'_3(s)} \right\}$$

$$M''_3 = \left\{ \underbrace{\left\{ \begin{array}{l} level(u), level(c), level(s), \\ order(u, c), order(c, s), p(j) \end{array} \right\}}_{M''_3(s_{\perp})}, \underbrace{\{u[p(k : a \xrightarrow{u} v)]\}}_{M''_3(u)}, \underbrace{\{c[p(k : a \xrightarrow{c} t)]\}}_{M''_3(c)}, \underbrace{\emptyset}_{M''_3(s)} \right\}$$

M'_3 and M''_3 are not intended because they make $s[p(k : a \xrightarrow{u} v)]$ false, i.e., $M'_3 \not\models_s s[p(k : a \xrightarrow{u} v)]$ and $M''_3 \not\models_s s[p(k : a \xrightarrow{u} v)]$, for similar reasons (for $c[p(k : a \xrightarrow{c} t)]$ being in $M'_3(c)$ and as well as in $M''_3(c)$ that made satisfaction of $c[p(k : a \xrightarrow{u} v)] \ll cau$ not possible, instead forced $c[p(k : a \xrightarrow{c} t)]$ to be cautiously believed at level c). If either one of these T-models were minimal, it would have modeled $s[p(k : a \xrightarrow{u} v)]$, as dictated by logical entailment and implication. The following definition of *intended T-models* eliminates this possibility. \square

We can define a priority relationship between ground atoms, and use this to define a preference relationship between models. We prefer models in which there are fewer occurrences of higher priority atoms.

Definition 3.6 (Priority Relationship) Let $\langle \Delta, \bar{u} \rangle$ be level u database. We define a priority relationship $<_p$ on the ground atoms in $\langle \Delta, \bar{u} \rangle$. We write $a \leq_p b$ to denote $a <_p b$ or $a = b$. For every ground instance $C\theta$ of a clause in $\langle \Delta, \bar{u} \rangle$,

1. $A \leq_p B$, where A is the head of $C\theta$ and B is a ground atom in the body of $C\theta$.
2. $A <_p B'$, where A is the head of $C\theta$, B' is a cautious b-atom in the body of $C'\theta'$, B is a cautious b-atom in the body of $C\theta$, and B' is more specific than B .
3. $A \leq_p A'$, where A is the head of $C\theta$, B' is a cautious b-atom in the body of $C'\theta'$, and the head A' of $C'\theta'$ is more specific than A .

and for all ground atoms A, B, C and F :

- if $A \leq_p B$ and $B \leq_p C$, then $A \leq_p C$,
- if $A \leq_p B$ and $B <_p C$, (respectively $F <_p A$) then $A <_p C$
- if $A \leq_p B$ and $B <_p C$, (respectively $F <_p A$) then $A <_p C$ (respectively $F <_p B$).

Example 3.7 Consider the database in example 3.4, there are two ground rules $s[p(k : a \xrightarrow{u} v)] \leftarrow c[p(k : a \xrightarrow{u} v)] \ll cau$ and $s[p(k : a \xrightarrow{c} t)] \leftarrow c[p(k : a \xrightarrow{c} t)] \ll cau$, and $c[p(k : a \xrightarrow{c} v)]$ is more specific than $c[p(k : a \xrightarrow{u} t)]$. Based on case (2), in the above definition $s[p(k : a \xrightarrow{u} v)] <_p c[p(k : a \xrightarrow{c} t)]$.

Example 3.8 Again, consider the level s database $\langle D_2, s \rangle$ in example 3.4. Based on the definition for priority relationship, we get:

$$\begin{aligned} c[p(k : a \xrightarrow{c} t)] &\leq_p p(j) && \text{(rule 1)} \\ s[p(k : a \xrightarrow{c} t)] &\leq_p c[p(k : a \xrightarrow{c} t)] && \text{(rule 1)} \\ s[p(k : a \xrightarrow{u} v)] &\leq_p c[p(k : a \xrightarrow{u} v)] && \text{(rule 1)} \\ s[p(k : a \xrightarrow{u} v)] &<_p c[p(k : a \xrightarrow{c} t)] && \text{(rule 2)} \\ s[p(k : a \xrightarrow{u} v)] &\leq_p s[p(k : a \xrightarrow{c} t)] && \text{(rule 3)} \end{aligned}$$

This relationship is used to determine which model is the intended model, as formalized in the next definition. The goal of the following preference relation is to minimize higher priority atoms as much as possible.

Definition 3.7 (Intended T-model) Suppose that M and N are two distinct T-models of a database $\langle \Delta, \bar{u} \rangle$. Then N is preferable to M ($N \ll M$) if, for every atom A in $N - M$, there is an atom B in $M - N$ such that $A <_p B$. We write $N \leq M$ if $N \ll M$ or $N = M$. We say that model N is an intended T-model of $\langle \Delta, \bar{u} \rangle$ if there are no T-models of $\langle \Delta, \bar{u} \rangle$ preferable to N . \square

Example 3.9 Consider the models in example 3.6. In this case, definition 3.7 suggests that $M_3 \ll M'_3$ and $M_3 \ll M''_3$.

Example 3.10 Consider the database $\langle D_2, s \rangle$ in example 3.4 and the corresponding priority relationship described in example 3.8. For this database, the following candidate models can be identified.

$$\begin{aligned} M_2 &= \left\langle \underbrace{\left\{ \begin{array}{l} level(u), level(c), level(s), \\ order(u, c), order(c, s) \end{array} \right\}}_{M_2(s_{\perp})}, \underbrace{\{u[p(k : a \xrightarrow{u} v)]\}}_{M_2(u)}, \underbrace{\emptyset}_{M_2(c)}, \underbrace{\{s[p(k : a \xrightarrow{u} v)]\}}_{M_2(s)} \right\rangle \\ M'_2 &= \left\langle \underbrace{\left\{ \begin{array}{l} level(u), level(c), level(s), \\ order(u, c), order(c, s) \end{array} \right\}}_{M'_2(s_{\perp})}, \underbrace{\{u[p(k : a \xrightarrow{u} v)]\}}_{M'_2(u)}, \underbrace{\{c[p(k : a \xrightarrow{c} t)]\}}_{M'_2(c)}, \underbrace{\emptyset}_{M'_2(s)} \right\rangle \\ M''_2 &= \left\langle \underbrace{\left\{ \begin{array}{l} level(u), level(c), level(s), \\ order(u, c), order(c, s), p(j) \end{array} \right\}}_{M''_2(s_{\perp})}, \underbrace{\{u[p(k : a \xrightarrow{u} v)]\}}_{M''_2(u)}, \underbrace{\{c[p(k : a \xrightarrow{c} t)]\}}_{M''_2(c)}, \underbrace{\emptyset}_{M''_2(s)} \right\rangle \end{aligned}$$

The question is which is the intended model. There is only one atom in $M_2 - M'_2$, $A = s[p(k : a \xrightarrow{u} v)]$, and there is an atom, $B = c[p(k : a \xrightarrow{c} t)]$ in $M'_2 - M_2$, such that $A <_p B$ so M_2 is preferable to M'_2 . In fact the same applies for M_2 and M''_2 , so M_2 is preferable to M''_2 . \square

Lemma 3.1 (Intendedness of Some T-Models) Let $\langle \Delta, \bar{u} \rangle$ be a level \bar{u} database in which there are no cautious b-atoms. Then $\langle \Delta, \bar{u} \rangle$ has a unique minimal T-model, which is an intended T-model. \square

The ground clauses of a consistent database can be partitioned into levels based on the following relationship.

Definition 3.8 (Partitioned Database) Let $\langle \Delta, \bar{u} \rangle$ be a consistent level \bar{u} database. For all clauses $A \leftarrow B$ and $A' \leftarrow B'$, if $\mu(A) < \mu(A')$ then $(A \leftarrow B) <_l (A' \leftarrow B')$.

Let D_1, \dots, D_k be a partitioning of the ground clauses of a consistent database $\langle \Delta, \bar{u} \rangle$ that satisfies the $<_l$ relationship described above. Then we say that $\langle \Delta, \bar{u} \rangle$ has k partitions. \square

Example 3.11 Consider the database in example 3.4, the ground clause $s[p(k : a \xrightarrow{c} t)] \leftarrow c[p(k : a \xrightarrow{c} t)] \ll cau$ is in a lower partition than $s[p(k : a \xrightarrow{u} v)] \leftarrow c[p(k : a \xrightarrow{u} v)] \ll cau$ because $s[p(k : a \xrightarrow{c} t)]$ is more specific than $s[p(k : a \xrightarrow{u} v)]$.

Example 3.12 Again, consider the level s database $\langle D_2, s \rangle$ and its mapping in example 3.4. This database can be partitioned as follows.

$$P_1 = \left\{ \begin{array}{l} level(u), level(c), level(s), order(u, c), order(c, s), c[p(k : a \xrightarrow{u} v)], \\ c[p(k : a \xrightarrow{c} t)] \leftarrow p(j), s[p(k : a \xrightarrow{c} t)] \leftarrow c[p(k : a \xrightarrow{c} t)] \ll cau \end{array} \right\}$$

$$P_2 = \left\{ s[p(k : a \xrightarrow{u} v)] \leftarrow c[p(k : a \xrightarrow{u} v)] \ll cau \right\}$$

Consequently, the level s database $\langle D_2, s \rangle$ has 2 partitions. \square

Lemma 3.2 (Unique Minimal Extension) Let $\langle \Delta, \bar{u} \rangle$ be a consistent level \bar{u} database and D_1, \dots, D_{k+1} be a partitioning of the ground clauses of $\langle \Delta, \bar{u} \rangle$. Suppose D_1, \dots, D_k has a unique intended model M_k . Then there exists a unique minimal extension of M_k to a unique intended model M_{k+1} of $\langle \Delta, \bar{u} \rangle$. \square

Theorem 3.1 (Intendedness of T-Models) Let $\langle \Delta, \bar{u} \rangle$ be a consistent level \bar{u} database. Then $\langle \Delta, \bar{u} \rangle$ has exactly one intended T-model, which is denoted M_Δ . For every T-model M , we have $M_\Delta \ll M$. \square

Theorem 3.2 (Independent of Partitioning) Let $\langle \Delta, \bar{u} \rangle$ be a consistent level \bar{u} database. Then, M_Δ does not depend on the particular partitioning chosen for $\langle \Delta, \bar{u} \rangle$. \square

Consequently, every consistent MultiLog database has an *intended* T-model, and has at least one T-model, namely M_Δ . The declarative semantics of a MultiLog database is its intended T-model.

3.3 Fixpoint Theory

In this section, we present a constructive way of defining the least T-model for a MultiLog level \bar{u} database $\langle \Delta, \bar{u} \rangle$. The key idea is to construct the least T-model \mathbf{M}_Δ of a database

$\langle \Delta, \bar{u} \rangle$ by means of a bottom-up least fixpoint computation based on an immediate consequence operator $\mathbf{T}_{\Delta}^{\bar{u}}$. Since our T-interpretations are tuples of interpretations, we define $\mathbf{T}_{\Delta}^{\bar{u}}$ in terms of the immediate consequence transformation of each of the levels in $\langle \Delta, \bar{u} \rangle$.

Definition 3.9 (Fixpoint Operator) Let Δ be a closed consistent database at level \bar{u} and let $\widehat{\Delta} = \langle \widehat{A}, \widehat{S}, \widehat{H}, \widehat{Q} \rangle$ be its Herbrand instantiation defined as usual. Let I be an Herbrand interpretation for Δ . We define $\mathbf{T}_{\Delta}^{\bar{u}}$ to be the immediate consequence operator such that $\mathbf{T}_{\Delta}^{\bar{u}} = \langle T_{\Delta}^{\bar{u}}(I(s)) : s \in S \rangle$. The operator $T_{\Delta}^{\bar{u}}$ for each component $I(s) \in I$ is defined similar to the classical case as $T_{\Delta}^{\bar{u}} : \mathcal{P}(\mathcal{H}) \mapsto \mathcal{P}(\mathcal{H})$, such that

$$T_{\Delta}^{\bar{u}}(I(s)) = \{A \mid A \leftarrow G \in \widehat{\Delta}, \text{depth}(A) = s \text{ and } I(s) \models_{I, \bar{u}} G\}$$

Example 3.13 Consider the level s database $\langle D_4, s \rangle$ derived from database D_3 in example 3.6 by adding the rule $r_{13} : q(j)$ in Π_{D_3} . For the database $\langle D_4, s \rangle$, the intended T-model M_4 may be identified as follows:

$$M_4 = \left\langle \underbrace{\left\{ \begin{array}{l} \text{level}(u), \text{level}(c), \text{level}(s), q(j), \\ r(j), p(j), \text{order}(u, c), \text{order}(c, s) \end{array} \right\}}_{M_4(s_{\perp})}, \underbrace{\{u[p(k : a \xrightarrow{u} v)]\}}_{M_4(u)}, \underbrace{\{c[p(k : a \xrightarrow{c} t)]\}}_{M_4(c)}, \underbrace{\emptyset}_{M_4(s)} \right\rangle$$

However, if we consider the sets computed at every stage of $T_{\Delta}^{\bar{u}}$, we have the following sequence,

$$\begin{aligned} \delta_1 &= \{\text{level}(u), \text{level}(c), \text{order}(u, c), q(j), u[p(k : a \xrightarrow{u} v)]\} \\ \delta_2 &= \{r(j), s[p(k : a \xrightarrow{u} v)]\} \\ \delta_3 &= \{p(j)\} \\ \delta_4 &= \{c[p(k : a \xrightarrow{c} t)]\} \end{aligned}$$

giving us the T-model

$$M'_4 = \left\langle \underbrace{\left\{ \begin{array}{l} \text{level}(u), \text{level}(c), \\ \text{level}(s), q(j), r(j), p(j), \\ \text{order}(u, c), \text{order}(c, s) \end{array} \right\}}_{M'_4(s_{\perp})}, \underbrace{\{u[p(k : a \xrightarrow{u} v)]\}}_{M'_4(u)}, \underbrace{\{c[p(k : a \xrightarrow{c} t)]\}}_{M'_4(c)}, \underbrace{\{s[p(k : a \xrightarrow{u} v)]\}}_{M'_4(s)} \right\rangle$$

which is not intended as the component model $M'_4(s)$ is not minimal, i.e., $M'_4(s) \neq \emptyset$. As such the query returns the answer true. \square

From the example, it is obvious that $\mathbf{T}_{\Delta}^{\bar{u}}$ is not monotonic. If we can prove that $\mathbf{T}_{\Delta}^{\bar{u}}$ is monotonic in restricted circumstances, then we can take the fixpoint of $\mathbf{T}_{\Delta}^{\bar{u}}$ in those circumstances.

Theorem 3.3 (Monotonicity of $\mathbf{T}_{\Delta}^{\bar{u}}$) Let $\langle \Delta, \bar{u} \rangle$ be a consistent level \bar{u} database, and D_1, \dots, D_k be a partitioning of the ground clauses as described in section 3.2

1. $\mathbf{T}_{D_1}^{\bar{u}}$ is monotonic.
2. Let M_{k-1} be the T-model of $D_1 \cup \dots \cup D_{k-1}$. $\mathbf{T}_{D_k}^{\bar{u}}$ is monotonic if I is initially M_{k-1} .

The monotonicity of $\mathbf{T}_{\Delta}^{\bar{u}}$ guarantees the existence of a fixpoint. As such, the bottom-up fixpoint computation is defined as follows:

$$\mathbf{T}_{\Delta}^{\bar{u}} \uparrow^0 (I) = I$$

$$\begin{aligned}\mathbf{T}_{\Delta}^{\bar{u}} \uparrow^{n+1} (I) &= \mathbf{T}_{\Delta}^{\bar{u}} (\mathbf{T}_{\Delta}^{\bar{u}} \uparrow^n (I)) \\ \mathbf{T}_{\Delta}^{\bar{u}} \uparrow^{\omega} (I) &= \sqcup_{n < \omega} \mathbf{T}_{\Delta}^{\bar{u}} \uparrow^n (I).\end{aligned}$$

Let D_1, \dots, D_n be a partitioning on the clauses of database $\langle \Delta, \bar{u} \rangle$. Then a T-interpretation, M_{Δ}^* is computed using the fixpoint operator to find a fixpoint of $\mathbf{T}_{D_1}^{\bar{u}}, \mathbf{T}_{D_2}^{\bar{u}}$ and so on.

$$\begin{aligned}M_1 &= \mathbf{T}_{D_1}^{\bar{u}} \uparrow^{\omega} (\emptyset) \\ M_i &= \mathbf{T}_{D_i}^{\bar{u}} \uparrow^{\omega} (M_{i-1}), \text{ for } 1 < i \leq n \\ M_{\Delta}^* &= M_n.\end{aligned}$$

Example 3.14 Consider the database in example 3.8. The ground rule $s[p(k : a \xrightarrow{u} v)] \leftarrow c[p(k : a \xrightarrow{u} v)] \ll cau$ is in a higher partition so the fixpoints of the other partitions are found first, and the body of this rule is no longer satisfied, so neither is the head.

Example 3.15 Consider the level s database $\langle D_2, s \rangle$ in example 3.4 and its partitioning as presented in example 3.12. Using the fixpoint operator, we proceed as follows in order to compute its T-model $M_{D_2}^*$.

$$\begin{aligned}\mathbf{T}_{D_2}^s \uparrow^0 (\emptyset) &= \emptyset \\ \mathbf{T}_{D_2}^s \uparrow^1 (\emptyset) &= \{level(u), level(c), level(s), order(u, c), order(c, s), c[p(k : a \xrightarrow{u} v)]\} \\ \mathbf{T}_{D_2}^s \uparrow^{\omega} (\emptyset) &= \mathbf{T}_{D_2}^s \uparrow^1 (\emptyset) \\ M_1 &= \mathbf{T}_{D_2}^s \uparrow^{\omega} (\emptyset) \\ \\ \mathbf{T}_{D_2}^s \uparrow^0 (M_1) &= M_1 \\ \mathbf{T}_{D_2}^s \uparrow^1 (M_1) &= \{s[p(k : a \xrightarrow{u} v)]\} \\ \mathbf{T}_{D_2}^s \uparrow^{\omega} (M_1) &= M_1 \cup \mathbf{T}_{D_2}^s \uparrow^1 (M_1) \\ M_{D_2}^* &= \mathbf{T}_{D_2}^s \uparrow^{\omega} (M_1).\end{aligned}$$

Each M_i is a fixpoint of $\mathbf{T}_{D_i}^s$, and we show below that $M_{D_2}^*$ is a rather special T-model of $\langle D_2, s \rangle$.

Theorem 3.4 (Soundness and Completeness) Let $\langle \Delta, \bar{u} \rangle$ be a consistent level \bar{u} database. Then M_{Δ}^* does not depend on the particular partitioning chosen for $\langle \Delta, \bar{u} \rangle$, and M_{Δ}^* is the intended T-model of $\langle \Delta, \bar{u} \rangle$. \square

We have the following theorem as a corollary which shows the equivalence between the model theoretic semantics and the proof theory [8].

Theorem 3.5 (Equivalence) Let $\langle \Delta, \bar{u} \rangle$ be a level \bar{u} database, \mathbf{M}_{Δ} be its intended T-model, G be any ground goal, \vdash be the provability relationship introduced in [8], and ϵ be the empty substitution. Then, we have

$$\langle \Delta, \bar{u} \rangle \vdash_{\epsilon} G \iff \mathbf{M}_{\Delta} \models_{\bar{u}} G$$

4 Conclusion

To our knowledge, MultiLog is the first logic based query language for MLS databases. It provides support for multiple belief models and ad hoc belief reasoning. It is free

from security breaches such as surprise stories identified in [8]. It supports the possibility of tailoring belief functions according to application needs making it incremental and customizable.

On the theoretical front, we have developed a sound and complete proof procedure and fixpoint operator to constructively compute intended models of databases. We have utilized a critical connection between the concept of inheritance in object-oriented languages and the views at different levels in a MLS database. This connection has helped us develop the logical semantics presented in this paper. The establishment of the equivalence of the three logical characterizations – proof theoretic, model theoretic and fixpoint – in the context of MLS deductive databases is significant from a theoretical perspective, as it gives insight into the understanding of the logical behavior and mathematical foundations of the language. In the context of databases, the model theoretic semantics and its fixpoint procedure as we proposed here is significant as it now enables us to compute the semantics of MLS databases in a bottom-up fashion.

References

- [1] P. Bonatti, S. Kraus, and V. S. Subrahmanian. Foundations of secure deductive databases. *IEEE TKDE*, 7(3):406–422, 1995.
- [2] F. Cuppens. Querying a multilevel database: A logical analysis. In *Proc. of the VLDB Conference*, pages 484–494, 1996.
- [3] D. E. Denning, T. F. Lunt, R. R. Schell, M. Heckman, and W. R. Shockley. A multilevel relational data model. In *IEEE Symp. on Security and Privacy*, 1987.
- [4] G. Dobbie and R. Topor. A Model for Sets and Multiple Inheritance in Deductive Object-Oriented Systems. In *Proc. DOOD*, pages 473–488, December 1993.
- [5] S. Jajodia, P. Samarati, and V. S. Subrahmanian. A logical language for expressing authorizations. In *IEEE Symp. on Security and Privacy*, pages 31–42, 1997.
- [6] S. Jajodia and R. Sandhu. Toward a multilevel secure relational data model. In *ACM SIGMOD*, pages 50–59, 1991.
- [7] H. M. Jamil. Implementing abstract objects with inheritance in Datalog^{neg}. In *Proc. of the VLDB Conf.*, pages 56–65, 1997.
- [8] H. M. Jamil. Belief reasoning in MLS deductive databases. In *ACM SIGMOD*, pages 109–120, 1999.
- [9] H. M. Jamil and G. Dobbie. On the logical foundations of multi-level secure databases. Technical Report TR-IDB-2002-03, Department of Computer Science, Mississippi State University, USA, May 2002. In preparation.
- [10] N. A. Jukic and S. V. Vrbsky. Asserting beliefs in MLS relational models. In *Sigmod Record*, pages 30–35, Ithaca, NY, 1997. ACM Press.
- [11] V. Kessler and G. Wedel. AUTLOG – an advanced logic of authentication. Manuscript.
- [12] M. Kifer, G. Lausen, and J. Wu. Logical Foundations for Object-Oriented and Frame-Based Languages. *JACM*, 42(3):741–843, July 1995.
- [13] T. F. Lunt, D. E. Denning, R. R. Schell, M. Heckman, and W. R. Shockley. The seaview security model. *IEEE Trans. on Soft. Engg.*, 16(6):593–607, 1990.
- [14] D. Miller. A Logical Analysis of Modules in Logic Programming. *Journal of Logic Programming*, 6(1/2):79–108, jan/mar 1989.
- [15] A Spalka. Fundamental forms of confidentiality in deductive databases. Manuscript.
- [16] W. Winiwarter. Why is deduction required for database systems ? - some case studies. In *Proc. of the 2nd Data Engineering Forum*, Tokyo, Japan, November 1995.