Formal Semantics of Confidentiality in Multilevel Logic Databases

Adrian Spalka

Department of Computer Science III, University of Bonn
Römerstr. 164, D-53117 Bonn, Germany
Fax: - 49 - 228 - 550 382, Email: adrian@cs.uni-bonn.de

Abstract

*This paper presents a new formal approach to the definition of confidentiality in multilevel logic databases. We regard a multilevel secure database as an extension of an open database which preserves the database-semantics. We give four definitions of confidentiality which capture various degrees of information on secrets. Three of them are relevant in the presence of the Closed World Assumption. We present their formalisation within standard predicate logic and their interpretation for multilevel databases. From this viewpoint, the definitions lead to a formal semantics of the Simple-Security-Property and the *-property. In particular, we demonstrate that the traditional interpretation of these properties represents just a special case of our formalism.*

1 Introduction

In this section we give an informal definition of an ordinary and that of a multilevel logic database, we motivate our approach and, finally, discuss previous works and related approaches.

1.1 Overview

A state of the world as seen by a logic database (LDB) consists of facts, rules and general laws. The LDB maps a state of the world into a set of data and a set of integrity constraints. The LDB uses clauses for the uniform representation of data, constraints and queries. The symbols which can occur in a clause are stored in the LDB's signature. A LDB is valid if the data satisfy the integrity constraints, viz the data allow the derivation of the constraints.

In a multilevel state of the world, a set of security levels is assigned to each piece of information. According to Thuraisingham (1991), information in a multilevel state of the world is the knowledge of the truth value of a statement with respect to a particular security level. A multilevel database (MLDB) consists of two components: a database and a partially ordered classification scheme, where a set of security levels is assigned to each element of the signature, data item and integrity constraint. The classification in the multilevel database is assumed to correspond to the classification in the multilevel world. The handling of integrity constraints and the relationship of information at different levels are controversial issues; they are discussed in the next section.

A security policy regulates the access of processes to a MLDB. The security policy encountered most often is Bell and LaPadula's (BLP) interpretation of the mandatory access control, which is described in Landwehr (1981). BLP assigns a maximum security level to each process (or equivalently, the user on whose behalf the process executes) which is allowed to have access to the database. The security policy of BLP is formulated in terms of explicit primitive read- and write-operations, but its two most important properties are usually translated for MLDB in the following way:

- The Simple-Security-Property requires that a process is only allowed to select a data item if the process' security level is greater than or equal to the item's level.
- The *-property requires that a process is only allowed to modify the database in such a way that for each data item involved in the modification, ie insert-, delete- or update-operation, the item's security level is greater than or equal to the process' level.

Without going into details, we note that in order to avoid some of its implications, the *-property is often simplified to allow a modification only for data items

which have the same security level as the acting process.

The Simple-Security-Property implicitly expresses a MLDB's confidentiality requirements. It is understood that an object must be kept secret from a user if the object's security level is greater than or incomparable with the user's level.

1.2 Rationale

The use of standard predicate logic for the description of databases has a number of widely accepted advantages. To us, the two most important ones are the unambiguous semantics and the uniform representation of data and constraints. The most important semantical task of an ordinary, open LDB is to watch over the validity of the data with respect to the constraints. This is obviously not the only task of a LDB, but if the constraints are removed from a database, then, in our opinion, this is no longer a database. It is rather an arbitrary set of data with some sophisticated methods which can answer queries and modify the contents of this set.

The original definition of BLP expresses the confidentiality requirements of a multilevel system through read- and write-operations. This is appropriate in a file- and record-orientated environment in which the only (direct or indirect) way to obtain the contents of a record or file is by reading it itself. This view assumes that if only non-confidential information is transmitted to a user, then the confidential information is kept secret from him.

The situation changes when we move to a logic-based environment. To be able to *read* a clause from a set of clauses means to be able to derive it from the set. Here the read-operation should be replaced by the process of deriving a clause.

Now two problems emerge. Firstly, it is possible that a user can gain knowledge of a clause even if it is not transmitted to him. Secondly, in the original environment, the allowance and prohibition of a read-operation are complementary actions, and the confidentiality of, eg, a record is based on this fact. It is kept secret if it cannot be read. For a clause, the properties of being or not being derivable from a set of clauses are not the only possible relationships between a clause and a set of clauses. Therefore the precise meaning of the statement 'A clause is secret if it is not derivable' is 'The secrecy of a clause is preserved in any other case except when it is derivable'. Does this match our intuition? We argue that it does not and that in a definition of confidentiality, it is necessary to name explicitly the relationship that must hold between a clause and a set of clauses.

Let us at last assume that such a definition of confidentiality is given. From the viewpoint of logic, the only difference between any set of clauses and a set forming a database state is that the former's contents may be arbitrary, while the latter's must satisfy some (static) integrity constraints. Thus, to affect a clause's derivability or confidentiality, or in the broadest sense, its relationship to the data of a state, it may no longer suffice to modify just these data. From now on we must take also the integrity constraints into consideration, eg whether they allow a particular modification of the data, or whether they themselves can be modified. We are in no case allowed to ignore them– they form an integral part of a database.

In summary, the derivability or confidentiality of a clause depends on the data of a state. The contents of a state are in turn fixed up to a degree of freedom which is determined by the integrity constraints.

In this light we think it incorrect to speak of a *fundamental conflict* between confidentiality and integrity. It is possible that the degree of freedom is insufficient to keep a particular secret, but can we simply assume that a secret can always be kept? As in real life itself, if there are some known boundary conditions which uniquely identify a thing, then it is useless to try to keep it secret.

The main objective of this paper is to give an interpretation of the BLP appropriate to multilevel logic databases.

1.3 Related work

The relevant works most often concentrate either on a formal definition of confidentiality or a practical construction of multilevel relational databases.

According to Gougen/Meseguer (1984), a confidentiality requirement expresses that 'under certain conditions, certain individuals *should not* have access to certain information'*. Its formalisation as non-interference is specifically intended to model trusted processes, but the authors also introduce a simple model of a multilevel-secure database. In this context, they interpret non-interference as non-derivability.

Morgenstern (1987) notes that in order to keep a piece of information in a deductive database secret, it may not be sufficient to make it directly inaccessible. The author speaks of deductive databases in an informal manner and uses them mainly to accentuate

* Gougen/Meseguer (1984):75.

some new problems which arise during the transition from relational databases.

The first basic attempt of a formal treatment of confidentiality is presented in Thuraisingham (1991). The author's main idea is to formalise the multilevel security properties in NTML, a non-monotonic logic. Although this approach points to the right direction, NTML has been shown to be not sound.*

Berson/Lunt (1987a) and Berson/Lunt (1987b) investigate the possibility of the application of the MAC-model to deductive databases. They point out many new problems and suggest an approach to tackle them, but, due to the initial nature of these works, no solutions are offered.

Meadows/Jajodia (1987), Burns (1990) and Wiseman (1990) are examples of early approaches which consider a multilevel relational database in which primary key and foreign key constraints are the only classes of integrity constraints. Burns (1990) and Wiseman (1991) note that there is a fundamental conflict between secrecy and integrity, since each of them can only be enforced at the expense of the other.

The handling of polyinstantiation has also received a lot of attention, eg in Jajodia/Sandhu (1990), Sandhu/Jajodia/Lunt (1990) and Lunt (1991). Many of the proposed solutions are of a syntactical character, thus each solution solves one problem while opening the way for another.

Denning et al (1988), Jajodia/Sandhu (1990) and Jajodia/Sandhu (1991) are three of the first papers which recognise that not every tuple in a multilevel relational database (ML-RDB) corresponds to a true fact in the real world. To exclude the unwanted tuples from a security level, they introduce the notion of a filter function. However, their definition does not prevent the database from violating integrity.

In the approach of Smith/Winslett (1992), a tuple is only believable to a user if both have the same security level. This highly conservative assumption seems to be neither practically relevant, nor theoretically clear since the authors speak of believability in an informal manner.

The most recent paper on ML-RDB is Qian (1994). The author claims that integrity should be enforced at every security level only on those tuples which are believable at this level. However, to determine a tuple's believability, she uses purely syntactic filtering functions. Lastly, the author believes† that ML-RDB with general integrity constraints unavoidably introduce functions with random choice, ie a random semantics–a standpoint which in our opinion is definitely wrong.

2 Basic definitions

Following Cremers/Griefahn/Hinze (1993) we consider databases from the viewpoint of predicate logic. Thus the discussion and the results are also valid for relational databases in proof-theoretical representation.‡

2.1 Predicate logic

A signature is a pair $\Sigma = (\mathbf{FS}, \mathbf{PS})$. The set **FS** contains ranked function symbols and **PS** ranked predicate symbols. Both sets, **FS** and **PS**, are non-empty, finite and disjunct.

The set of terms over a signature Σ, $\mathbf{TE}^{\Sigma}$, is the smallest set with the following properties: each variable is a term; each constant, ie a function symbol of rank 0, is a term; let f be a function symbol of rank k and $t_1,\ldots,t_k$ terms, then $f(t_1,\ldots,t_k)$ is a term. A term is ground if it does not contain any variable.

Let r be a predicate symbol of rank k and $t_1,\ldots,t_k$ terms, then $r(t_1,\ldots,t_k)$ is an atomic formula or for short an atom. Let α be an atomic formula, then α is a positive literal and $\neg\alpha$ a negative literal. We denote the set of atomic formulae over Σ by $\mathbf{AF}^{\Sigma}$ and the set of literals over Σ by $\mathbf{LIT}^{\Sigma}$. A literal is ground if it comprises only ground terms.

A clause is a formula of the form

$$\alpha_1 \vee \ldots \vee \alpha_m \leftarrow \lambda_1 \wedge \ldots \wedge \lambda_n$$

in which all variables are assumed to be universally quantified. Each α_i in the head of the clause is an atom and each λ_j in its body a literal. A clause is range-restricted if each of its variables occurs also in a positive literal in its body; indefinite or disjunctive if $m > 1$; normal if $m = 1$; a query if $m = 0$; and ground if it comprises only ground literals. A normal clause is called a rule if $n \geq 1$ and a fact if $n = 0$.

We assume that all clauses are range-restricted. We denote the set of all these clauses over Σ by $\mathbf{CL}^{\Sigma}$ and its subset of normal clauses by $\mathbf{NCL}^{\Sigma}$.§

Let $I \subseteq \mathbf{CL}$ be a set of clauses, then $Th(I) \subseteq \mathbf{CL}$

---

* cf Garvey et al (1992):160.

† Qian (1994):213, line 15.

‡ cf Reiter (1984).

§ We omit the superscript whenever the respective signature is evident.

denotes all clauses which can be (logically) derived from I (for a clause φ, $\varphi \in Th(I)$ is also denoted as $I \vdash \varphi$). We denote the set of all atomic formulae in $Th(I)$ by $F(I)$, ie $F(I) = Th(I)|_{AF}$.

Let $I \subseteq \mathbf{NCL}^{\Sigma}$ be a set of normal clauses. The completion of I as defined in Cremers/Griefahn/-Hinze (1993)* is denoted by $\bar{I}$. We assume that the declarative semantics of a set $I \subseteq \mathbf{NCL}^{\Sigma}$ is given by its completion.

2.2 Logic databases

A scheme is a pair $DB = (\Sigma, C)$, where $\Sigma = (\mathbf{FS}, \mathbf{PS})$ is a signature and $C \subseteq \mathbf{CL}^{\Sigma}$ a set of clauses. Σ determines the set $\mathbf{CL}^{\Sigma}$, which is the language of *DB*. The elements of C are called static integrity constraints; they represent invariable properties of the world.

A valid state of *DB* is a set $I \subseteq \mathbf{NCL}^{\Sigma}$ the completion of which is consistent and which satisfies the static integrity constraints, viz $C \subseteq Th(\bar{I})$ holds. The present state of *DB*, $db = I$, can be any valid state I.

Let $\leftarrow \Lambda$ be a query, where $\Lambda = \lambda_1 \wedge \ldots \wedge \lambda_n$. The answer-set of $db = I$ to $\leftarrow \Lambda$ is the set of all ground substitutions, $\Pi = \{\pi_1, \ldots\}$, for the variables of Λ such that $\Lambda\pi_i \in Th(\bar{I})$.

We assume that a transaction can change the present state of a database. However, its formal definition is not needed in this paper.

2.3 Databases with users and rights

The database presented above is an open one because it cannot tell one user from another–it answers any query and follows any valid transaction in the same manner. A database must be able to recognise the users if it is expected to treat them differently. Therefore we add to our database a set P of all users or persons who have access to it. We also introduce for each user $p \in P$ the following rights:

- $RS_p \subseteq \mathbf{CL}^{\Sigma_p}$† determines the clauses a person may see as an element of I or C.
- $RD_p \subseteq RS_p$ determines the clauses a person is allowed to delete.
- $RI_p \subseteq \mathbf{CL}^{\Sigma_p}$ determines the clauses a person is allowed to insert.

Now we have arrived at a database which recognises different users and is able to behave in accordance with the stated rights. We call it a database with rights.

2.4 Personal database profiles

Let *DB* be a database with the scheme $DB = (\Sigma, C)$ and the state $db = I$. The application of *RS*, the right to see, to *DB* provides for each user p his profile DB_p with the scheme $DB_p = (\Sigma_p, C_p)$ and the state $db_p = I_p$.

One of the requirements to the profile is that it satisfies the confidentiality requirements for the user p. But there is more than this. Our starting point has been an open database. Then we have added users and rights to it. If a user possesses all rights, then his profile is identical to the whole database. Otherwise, his profile is different from it. Should the database semantics of the whole database or of a profile be allowed to vary depending on the actual settings of the rights? We maintain that the desirable answer is in both cases 'No'. We would like to look on a profile as an independent open database which respects the validity of the whole database. Thus we must determine the relationships between the original database and a profile, and between profiles.

First of all we must require that *DB* should always be valid and that validity of a state $db = I$ depends only on the constraints C, ie $C \subseteq Th(\bar{I})$.

Secondly, a state $db_p = I_p$ of the profile DB_p should also be always valid, and since DB_p should behave as if it were an autonomous database, its validity must not depend on anything else but the constraints of $DB_p = (\Sigma_p, C_p)$. Thus we require that $C_p \subseteq Th(\bar{I}_p)$.

Thirdly, a user's transaction can never violate C_p, but since *DB* is the ultimate authority on integrity, it must not happen that a transaction violates C, ie $C_p \subseteq Th(\bar{I}_p)$ and $C \not\subset Th(\bar{I})$. Formally this can be translated into the requirement

$$C_p \not\subset Th(\bar{I}_p) \vee C \subseteq Th(\bar{I})$$

or equivalently

---

* Cremers/Griefahn/Hinze (1993):60.

† For the moment it suffices to know that the sets of symbols of Σ_p, the signature of p, are subsets of the respective sets of Σ. The motivation for the removal of a symbol from Σ_p is given later.

$$C_p \subseteq Th(\bar{I}_p) \rightarrow C \subseteq Th(\bar{I})$$

Finally, we require that the validity of two profiles is independent from each other. This means that a valid transaction executed by one user may not invalidate the profile of another user. The formal interpretation depends on the relationship between the data of two profiles. They are obviously independent if they do not share any data. We later investigate the case when one is a subset of the other, which is usually considered to hold in multilevel databases.

3 Formal semantics of confidentiality

In this section we present a summary of the results of Spalka (1994). An object of protection in a logic database is either a symbol of the signature, an atomic formula, ie a fact, or a clause, ie a rule. However, atomic formulae play here a central role.

3.1 Confidentiality of symbols

Symbols of the signature cannot be directly manipulated. A symbol is only a part of a clause. To keep a symbol secret from a user can thus only mean that:

- This symbol does not appear in any clause of the user's data or constraints.
- The database responds with 'I don't understand', viz Syntax error, to a query or transaction of the user if it comprises this symbol.

Both points are immediately linked to the signature of the user-profile. They can be satisfied when the secret symbol is removed from it. One should however keep in mind that the removal of just one symbol from the signature can reduce the language by a considerable number of clauses.

3.2 Confidentiality of facts

Let α be a fact, I a set of clauses and α is derivable from I, ie $\alpha \in Th(I)$. Let us also assume that α should be kept secret from the user p with regard to I. As long as p does not mention α, its secrecy is preserved. But what should the database answer when the user asks

Does $\alpha \in Th(I)$ hold?

There are (at least) five possible answers: 'Yes', 'Maybe', 'No', 'I don't know' and 'I don't understand'.

The first answer tells the whole truth and obviously does not preserve secrecy. But which of the remaining four possibilities preserve secrecy? The second answer is not a lie, but it is also not the whole truth. The database admits that it knows the truth but it is not going to tell it. The 'No'-answer is a blunt lie. In the fourth case, the database admits to understand the question, but it pretends not to know the answer. Finally, in the last case, the database pretends not even to understand the question.

In general, each answer except 'Yes' is suitable to keep the secret. However, depending on the circumstances, an answer can be too *weak* in a particular situation. We see that there is no unequivocal definition of secrecy. Some things can be more secret than other. Each of the five answers gives the user a different amount of information on the secret. Since the amount of information is gradually decreasing with each point, we can say that each answer represents a degree of confidentiality.

We take the view that the decision on the real secrecy of a secret or on the amount of information about a secret which a user may acquire must be made by an application. Thus it is necessary to assign a degree of confidentiality to a confidentiality requirement. But first we translate the informal answers into formal expressions in the context of a logic database:

G0: $\alpha \in Th(\bar{I})$

G1: $\alpha \vee \alpha' \vee \alpha'' \vee \ldots \in Th(\bar{I})$

G2: $\alpha \notin Th(\bar{I})$

G3: $\alpha \notin Th(\bar{I})$ and $\neg\alpha \notin Th(\bar{I})$

G4: $\alpha \notin \mathbf{AF}$

A confidentiality requirement for an atomic formula is now a statement of the form 'α should be kept secret from p at the degree G' where G is one of G1 to G4.

G1 is the only degree of which we can say that it does not allow the database to lie to conceal a secret. It only provides him with a weaker information than it is capable of, but this information is still true. If we contemplate the possible consequences of a lie from a practical and ethical point of view, then it seems preferable to give imprecise rather than false information. This preference is also underlined by the effort needed to enforce G2, which may require the maintenance of a consistent set of lies.

Finally, we note that the traditional definition of confidentiality as non-derivability is equivalent to the G1-degree in our formalism.

3.3 Confidentiality of rules

In principle, it would be possible to define the confi-

dentiality of a clause in the same way as for an atomic formula. We believe that this is inappropriate. In our opinion, a reason for keeping a rule confidential is that it is used to derive confidential data. To give an example, let $s(X) \leftarrow r(X)$ be a confidential rule and $r(a)$ a fact. Then $s(a)$ should also be kept secret.

We thus say that the requirement to keep a rule confidential, means that:

i) This rule is not among the stored data or integrity constraints.

ii) The data which can be derived by this rule should also be kept secret.

Since a fact is a rule with an empty body, this definition is a proper extension of the definition of confidentiality of a fact.

4 Confidentiality in multilevel databases

This section discusses the adaptation of BLP based on the MAC-model to multilevel logic databases.

4.1 The MAC-model

The MAC-model can be defined as

$$\mathbf{M}_{MAC} = (O, S, SG, L)$$

O is a set of objects, ie units of protection. The set S contains subjects which represent users that work with the objects. SG is a partially ordered set* the elements of which are interpreted as security levels. $L: S \cup O \rightarrow SG$ is a function which places a security mark on every subject and object. The value of $L(o)$, $o \in O$, is interpreted as the object's degree of confidentiality, and the value of $L(s)$, $s \in S$, as the subject's degree of trustworthiness.

The MAC-model is assumed to satisfy two properties. The Simple-Security-Property states for a file-orientated environment that $L(s) \geq L(o)$ is necessary and sufficient in order that s may read o, and it is understood that any object which s may not read must be kept secret from him. The *-property states that $L(o) \geq L(s)$ is necessary and sufficient in order that s may create or write o.

Now we give an interpretation of the MAC-model for logic databases. The objects of O are identified with symbols of the signature, facts and clauses. The subjects of S are identified with the users in P and the database commands. SG and L are adopted as new components of DB. The interpretation of the two properties depends on the object. Before we go into details, let us take a look at the original intention of both properties.

4.1.1 The Simple-Security-Property

The function L enables us to relate an object and a subject. The Simple-Security-Property uses this relationship to express two points. Firstly, the property itself is the following implicit, generic confidentiality requirement: an object o should be kept secret from a subject s, if $L(s) \geq L(o)$ does not hold. Secondly, this property shows us how to satisfy this confidentiality requirement in a file-orientated environment: if o should be kept secret from s, then s should not be given read-access to o.

In its original definition, both points are merged into one statement. This is appropriate for a file-orientated environment, but for a logic database we must consider both points separately.

4.1.2 The *-property

A subject can actively or passively acquire knowledge either by executing read-operations or by waiting until other subjects execute write-operations which are addressed to him. The Simple-Security-Property is concerned with the first case. The *-property worries about other subjects' write-operations. Is this really something we need to worry about in a model?

The *-property limits a user's ability to perform modifications of a system. It prevents him from modifying an object the security level of which is *lower* than his own. This restriction is hard to understand when we keep in mind that a user is only assigned a specific security level if he is trusted to behave properly. Since the *-property does not state anything about a user's trustworthiness, we must try to give a different interpretation to it.

If this property is concerned about a situation in which a user may be misled to use an untrustworthy command which pretends to be trustworthy, then it can be safely abandoned if the implementation of the commands can be trusted. In this case the *-property does not belong to the model, but is rather an implementation requirement. If, on the other hand, its intention is that a system itself may not *write-down* any information not approved of by the Simple-Security-Property while it is processing a read-operation, then it is evidently not concerned about the possibility that the system will deliberately and intentionally violate the Simple-Security-Property. In our opinion, explicit

* Some authors define SG as a lattice.

modifications which violate the *-property should be admissible on account of their implied trustworthiness.

To us the *-property has only one meaningful interpretation: if two subjects, who may be users or commands, are able to communicate with each other, then a communication must be conducted in such a way that neither party will be provided with any *implicit* knowledge on information which should be kept secret from it and which is visible to the other party. If both subjects are users, then we can do nothing but to rely on their trustworthiness. If on the other hand a user is communicating with a database, then we must establish instructions for its behaviour. Yet in both cases we are forced to define the kind of implicit knowledge which may not be written down.

We advocate to choose an interpretation for the *-property which agrees to the assumptions about a subject's trustworthiness expressed by the function L. In particular, we do not regard the *-property as a restriction on explicit modifications, but only as a requirement to confine specific kinds of implicit information transfers.

In this light, in a theoretical model the *-property is subsumed by our interpretation of the Simple-Security-Property, since the kind or degree of information which a subject is allowed to have on a secret can be expressed within a confidentiality requirement in our formalism.

4.2 Confidential symbols

When symbols of the signature are objects of protection, the situation resembles very much that in a file-orientated environment.

Let a and b be two symbols and ph and pl two users such that $L(ph) > L(pl)$, $L(pl) = L(a)$ and $L(ph) = L(b)$. The signature of ph comprises both a and b, while according to section 3.1, b is not an element of pl's signature.

Thus for the users ph and pl, the Simple-Security-Property induces an inclusion-relation on their signatures.

4.3 Confidential facts

Let ph and pl be two users with their database profiles DB_{ph} and DB_{pl} so that $L(ph) > L(pl)$. Let moreover α be a fact from the data of the state $db_{ph} = I_{ph}$ and $L(ph) = L(\alpha)$. The Simple-Security-Property tells us that α should be kept secret from pl with regard to DB_{pl}. In section 3.2 we have shown that this requirement must be qualified with a degree of confidentiality, which can be G1, G2, G3 or G4.

4.3.1 G1

This weakest confidentiality-degree allows pl to have indefinite information on α. Let us consider the following example. Let

$$\Sigma = (\mathbf{FS} = \{a\}, \mathbf{PS} = \{q, r, s\})$$
$$C = \{q(X) \vee r(X) \leftarrow s(X)\}$$

be a LDB-scheme visible to the user pl. Let moreover $F(\bar{I}_{ph}) = \{r(a), s(a)\}$, and $r(a)$ should be kept secret from pl at G1-degree. pl must not be able to derive $r(a)$. Thus we reduce pl's set of positive data to $F(\bar{I}_{pl}) = \{s(a)\}$. Now the trouble is that I_{pl} does not satisfy C, and we owe the user an explanation. We suggest to tell him that his profile is weakly consistent, that is:

- the integrity constraints in C are always satisfied by the data in $db = I$
- his data may seem to violate C due to some secrets

Now the user is able to identify the violated constraint, and through a simple substitution he can find out that $q(a) \vee r(a) \in Th(\bar{I})$ holds, viz either $q(a)$ or is $r(a)$ true. Maybe $r(a)$ is true, or maybe not.

We see that the interpretation of G1 in a LDB involves some interactions and new conventions. The general enforcement of G1 is based on the following method. Firstly, reduce the data in a user's profile so that he can not derive the secret fact from it. Secondly, observe how the reduction affects the user's integrity constraints. If all constraints are satisfied, then the user cannot use them to derive any further information. If a constraint is violated, then we should hope that it is an indefinite clause, viz it will only tell the user that a disjunction of some facts is true. However, if this constraint is a definite clause, then it may not be possible to enforce G1.

Since G1-requirements only reduce the data of a profile but do not introduce any data, the data of a profile are always a subset of the global database's data. For our users ph and pl, the Simple-Security-Property induces an inclusion-relation on their positive data, ie facts:

$$F(\bar{I}_{pl}) \subseteq F(\bar{I}_{ph}).$$

Does the same relationship also hold for their sets of integrity constraints? The answer is a definite 'No'. The properties of *Th* as a hull-operator, the validity of a profile and the subset relation on the sets of data yield only the following inclusions:

$$C_{pl} \subseteq Th(\bar{I}_{pl})$$
$$C_{pl} \subseteq Th(\bar{I}_{ph})$$
$$C_{ph} \subseteq Th(\bar{I}_{ph})$$

The relationship $C_{pl} \subseteq C_{ph}$, or more general $Th(C_{pl}) \subseteq Th(C_{ph})$, does not follow from the above inclusions. In our opinion, to state it as a requirement would only limit the database's expressiveness.

We believe that integrity constraints must only satisfy the semantics-preserving properties of a personal database profile. Here the independence of the profiles of *pl* and *ph* has two consequences. Firstly, the construction of C_{pl} and C_{ph} must ensure that *pl*'s valid transactions do not invalidate *ph*'s profile. Secondly, the transactions of *ph* are guaranteed to respect the validity of *pl*'s profile if they only affect data of his own *level*. However, based on *ph*'s trustworthiness, he can be allowed to execute any transaction which leads even to a weakly consistent profile of *pl* as long as no secret fact at G1-degree is disclosed.

4.3.2 G2

Let α be a fact which should be kept secret from the user *pl* at G2-degree. The database is required to ensure that $\alpha \notin Th(\bar{I}_{pl})$ and $C_{pl} \subseteq Th(\bar{I}_{pl})$.

The difference between G1 and G2 is that G1 allows a profile to become weakly consistent, whereas G2 does not. This is necessary in order to avoid the derivation of any information which cannot be derived from I_{pl}, ie the database must always answer with a convincing 'No'. Let us consider a variant of the example of the previous section.

$$\Sigma = (\mathbf{FS} = \{a\}, \mathbf{PS} = \{q, r, s\})$$
$$C = \{q(X) \vee r(X) \leftarrow s(X)\}$$
$$F(\bar{I}) = \{r(a), s(a)\}$$

We require that $r(a)$ should be kept secret from *pl* at G2-degree. Now we are not allowed to set $F(\bar{I}_{pl}) = \{s(a)\}$ since this gives *pl* indefinite information on the secret.

We see that there are two reasons for weak consistency:

- the secret $r(a)$ is not derivable from I_{pl}
- $q(a)$, which is not secret, is not present $F(\bar{I})$.

Consequently there are two ways to make *pl*'s profile consistent:

- Show *pl* the secret, viz insert $r(a)$ into I_{pl}
- Insert something else into I_{pl} which makes it consistent, ie insert $q(a)$.

This example shows that $q(a)$ represents from the database's viewpoint a plausible lie for $r(a)$, ie it may serve as a cover story* for a secret fact. We say that $q(a)$ is an alias for $r(a)$. In general, each fact from the violated constraint's head except the secret is a plausible lie.† However, if this constraint is a definite clause, then it offers no aliases for the secret. In this case the constraint uniquely identifies the secret, and confidentiality at G2-degree cannot be enforced.

Since the alias is a member of $F(\bar{I}_{pl})$ but not of $F(\bar{I})$, $F(\bar{I}_{pl})$ is no longer a subset of $F(\bar{I})$. For our users *ph* and *pl*, the Simple-Security-Property does not imply an inclusion of the sets of integrity constraints for the same reasons as for G1. Moreover, it can be no longer interpreted even as an inclusion on the sets of their data because G2-requirements may lead to a deliberate inclusion of false information into a user's profile. G2 provides a higher degree of confidentiality than G1, but aliases do not come without problems.

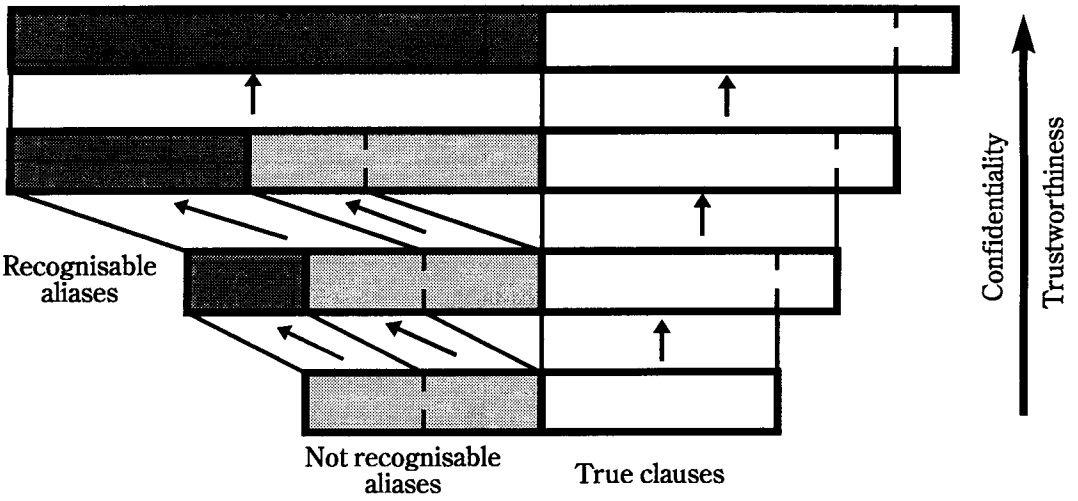
The next example motivates the interpretation of the Simple-Security-Property for G2-degree. Let us assume that the fact α must be kept secret from *pl* and that G2-secrecy can only be enforced if the alias β is inserted for α in *pl*'s data. *pl* cannot recognise β as an alias (it is placed in the light grey zone in the diagram on the next page). Let us assume that α is not secret to the user *ph* (it is located in his white zone). Now *ph* sees two different facts, which represent two different names for the same fact. How can *ph* recognise which of them is the true one, and which is an alias? If *ph* is considered trustworthy to see the truth, he must not be confused by false aliases.

We see that an alias inserted for a user at a low level can disturb the profile of a user at a higher level.

---

* cf Garvey/Lunt (1991).

† Brüggemann (1993) aptly points out that a good cover story is also expected to play down the covered secret as far as possible. Thus it would be advisable to measure the quality of a cover story with respect to a secret. Although we do not do it in this paper, our database can use some special predicates to express it, eg as an order on the possible plausible lies.



Thus we must provide for the possibility to *move* an alias from the light grey into the dark grey zone, viz out of the profile's data.

For a user p, the set of facts α which satisfy the condition $L(p) \geq L(\alpha)$ can be partitioned into three subsets:

i) true facts

ii) aliases which are not recognisable as such at p's security level $L(p)$

iii) recognisable aliases at $L(p)$

Thus for G2-degree the Simple-Security-Property induces between two users with adjacent security levels an inclusion-relation on the true facts and on the aliases which are not recognisable at both levels. For users with any two comparable security levels, the inclusion-relation holds only on the true facts.

4.3.3 G3

A confidentiality requirement at G3-degree can be expressed in standard predicate logic. However, it is trivially not satisfiable in databases with completion-semantics, viz, in which the Closed World Assumption is made. It tells us that for each atom α, either α or its negation $\neg\alpha$ is derivable. This obviously contradicts the formal G3-requirement.

4.3.4 G4

G4 is the strongest degree of confidentiality. It requires a database to give a user no information on a secret. According to section 3.2, this means that α is not a valid fact in the user profile's language, ie $\alpha \notin \mathbf{AF}^{\Sigma_{pl}}$. The only way to achieve it is to remove at least one symbol from the user's signature which he would need to construct the confidential fact. We see that confidentiality of facts at G4-degree can be reduced to confidentiality of symbols.

4.4 Confidential rules

The definition of confidentiality of a rule reduced to the rule itself requires that the rule should be neither an element of the data nor of the integrity constraints. Here further investigation is necessary in order to find out when and how this can be done without violating the database semantics.

5 Conclusion

In this paper we have presented a new approach to the definition of confidentiality in multilevel logic databases. An open deductive database has served as our starting point. With the introduction of users and rights we have defined the notion of global consistency and that of a personal database profile.

We have shown that secrecy has no unique meaning. We have given four possible definitions of secrecy, G1 to G4, which have been motivated by real-life situations. They correspond to the information which is contained in the informal answers 'Maybe', 'No', 'Don't know' and 'Don't understand', that is, they capture the various degrees of implicit information which a user may obtain on a secret. All definitions have been formalised within standard predicate logic. Three of them, G1 for indefinite, G2 for negative, and G4 for no information on secrets, are relevant in the presence of the Closed World Assumption. From the viewpoint of multilevel security, G1 to G4 provide a

formal semantics of the Simple-Security-Property and the *-property. In particular we have demonstrated that the traditional interpretation of these properties represents just a special case of our formalism.

References

Berson, Thomas A., and Teresa F. Lunt. 'Security Considerations for Knowledge-Based Systems'. *Third Expert Systems in Government Conference.* Reprint. 1987.

Berson, Thomas A., and Teresa F. Lunt. 'Multilevel Security for Knowledge-Based Systems'. *1987 IEEE Symposium on Security and Privacy.* IEEE Computer Society Press, 1987. pp 235-242.

Brüggemann, Hans Hermann. *Private communication.* Hildesheim, 1993.

Burns, Rae K. 'Integrity and Secrecy: Fundamental Conflicts in the Database Environment'. Ed Bhavani Thuraisingham. *3rd RADC Database Security Workshop 1990.* Bedford, Massachussets: Mitre, 1991. pp 37-40.

Cremers, Armin B., Ulrike Griefahn, and Ralf Hinze. *Deduktive Datenbanken.* Vieweg, 1993.

Denning, Dorothy E., Teresa F. Lunt, Roger R. Schell, William R. Shockley, and Mark Heckman. 'The SeaView Security Model'. *1988 Symposium on Security and Privacy.* IEEE Computer Society Press, 1988. pp 218-233.

Garvey, Thomas D., and Teresa F. Lunt. *Multilevel Security for Knowledge Based Systems.* Technical Report SRI-CSL-91-01. Menlo Park, CA: SRI International, 1991.

Garvey, Thomas D., Teresa F. Lunt, Xiaolei Qian, and Mark E. Stickel. 'Toward a tool to detect and eliminate inference problems in the design of multilevel databases'. Ed Bhavani Thuraisingham, and Carl E. Landwehr. *Database Security VI.* IFIP WG11.3 Workshop on Database Security 1992. Amsterdam: North-Holland, 1993. pp 149-167.

Gougen, Joseph A., and José Meseguer. 'Unwinding and Inference Control'. *1984 IEEE Symposium on Security and Privacy.* IEEE Computer Society Press, 1984. pp 75-86.

Jajodia, Sushil, and Ravi Sandhu. 'Polyinstantiation Integrity in Multilevel Relations'. *1990 IEEE Symposium on Research in Security and Privacy.* IEEE Computer Society Press, 1990. pp 104-115.

Jajodia, Sushil, and Ravi Sandhu. 'Toward a multilevel secure relational data model'. *ACM SIGMOD International Conference on Management of Data 1991.* 1991. pp 50-59.

Landwehr, Carl E. 'Formal Models for Computer Security'. *ACM Computing Surveys* 13.3 (1981): 247-278.

Lunt, Teresa F. 'Polyinstantiation: an Inevitable Part of a Multilevel World'. *The Computer Security Foundations Workshop IV.* IEEE Computer Society Press, 1991. pp 236-238.

Meadows, Catherine, and Sushil Jajodia. 'Integrity Versus Security In Multilevel Secure Databases'. Ed Carl E. Landwehr. *Database Security.* IFIP WG11.3 Workshop on Database Security 1987. Amsterdam: North-Holland, 1988. pp 89-101.

Morgenstern, Matthew. 'Security and Inference in Multilevel Database and Knowledge-Base Systems'. *1987 ACM SIGMOD Conference/ SIGMOD Record* 16.3 (1987):357-373.

Qian, Xiaolei. 'A Model-Theoretic Semantics of the Multilevel Relational Model'. Ed Matthias Jarke, Janis Bubenko, and Keith Jeffery. *Advances in Database Technology - EDBT'94.* LNCS, vol 779. Berlin et al: Springer-Verlag, 1994. pp 201-214.

Reiter, Raymond. 'Towards a Logical Reconstruction of Relational Database Theory'. Ed Michael L. Brodie, John Mylopoulos, and Joachim W. Schmidt. *On Conceptual Modeling.* New York: Springer-Verlag, 1984. pp 191-238.

Sandhu, Ravi, Sushil Jajodia, and Teresa F. Lunt. 'A new polyinstantiation integrity constraint for multilevel relations'. *The Computer Security Foundations Workshop III.* IEEE Computer Society Press, 1990. pp 159-165.

Smith, Kenneth, and Marianne Winslett. 'Entity Modeling in the MLS Relational Model'. *18th VLDB Conference.* 1992. pp 199-210.

Spalka, Adrian. 'Formal Semantics of Rights and Confidentiality in Definite Deductive Databases'. *IEEE Computer Security Foundations Workshop VII.* IEEE Computer Society Press, 1994. pp 47-58.

Thuraisingham, Bhavani. 'A Nonmonotonic Typed Multilevel Logic for Multilevel Secure Data/Knowledge Base Management Systems'. *IEEE Computer Security Foundations Workshop IV.* IEEE Computer Society Press, 1991. pp 127-138.

Wiseman, Simon. 'The Control of Integrity in Databases'. Ed Sushil Jajodia, and Carl E. Landwehr. *Database Security IV.* IFIP WG11.3 Workshop on Database Security 1990. Amsterdam: North-Holland, 1991. pp 191-203.

Wiseman, Simon. 'The conflict between confidentiality and integrity'. *IEEE Computer Security Foundations Workshop IV.* IEEE Computer Society Press, 1991. pp 241-242.