Logical Foundations of Multilevel Databases

Frederic Cuppens
ONERA/CERT
2 Avenue E. Belin
31055, Toulouse Cedex
France
email: cuppens@cert.fr

Alban Gabillon
Université de Toulon et du Var
GECT
B.P.132, 83 957, La Garde cedex.
France
email : gabillon@univ-tln.fr

Abstract

In this paper, we propose a formal model for multilevel databases. This model aims at being a generic model, that is it can be interpreted for any kind of database (relational, object-oriented…). Our model has three layers. The first layer corresponds to a model for a non-protected database. The second layer corresponds to a model for a multilevel database. In this second layer, we propose a list of theorems that must be respected in order to build a secure multilevel database. We also propose a new solution to manage cover stories without using the ambiguous technique of polyinstantiation. The third layer corresponds to a model for a *MultiView* database, that is, a database that provides at each security level a consistent view of the multilevel database. Finally, as an illustration, we interpret our 3-layer model in the case of an object-oriented database.

Keywords: Database Security, Security Model, Multilevel Security Policy, Cover Story Management, Mathematical Logic.

1. Introduction

The work presented in this paper applies to the context of multilevel security policies. In a multilevel security policy every piece of information is associated with a *classification* level, and every agent is associated with a *clearance* level. Classification and clearance levels are taken from a set of security levels associated with a partial order relation. Traditional examples of security levels are U (Unclassified), C (Confidential), S (Secret) and TS (Top Secret) with the following order relation: U < C < S < TS. The *confidentiality property* in the multilevel security policy states that an agent can only know a given piece of information if the clearance level of this agent is higher than or equal to the classification level of the information.

Projects, whose objective is to realize a database management system, which supports a multilevel security policy, have been undertaken for more than twenty-five years. Several commercial products coming from the main vendors of database management systems are now available.

Currently, database security is also an active field of research. Several theoretical models have been suggested for multilevel databases (see for instance [DLSSH88,Wil89,HOST90,SW92, CS95,QL96,JK90,Lun90,ML92,KTT89]). However, none of them is fully satisfactory because they are not free of semantic ambiguities. Most of these ambiguities come from the use of the concept called *polyinstantiation*. Polyinstantiation occurs when different tuples with the same key, each at a different classification level, are allowed. Even though models for multilevel security generally include the possibility to have polyinstantiated tuples, there is currently no consensus on a non-ambiguous interpretation of polyinstantiation.

The objective of this paper is to propose a formal model for multilevel database. This model is then interpreted in the case of an object-oriented database. The proposed model has three layers (see figure 1). The first layer corresponds to a model for a non-protected database. The second layer corresponds to a model for a multilevel database. This multilevel database is obtained by applying the classification process to data stored in the non-protected database. However, this model includes the possibility of classifying data, which are not stored in the non-protected database. These data correspond to *cover stories*. Cover stories are lies introduced in the multilevel database in order to protect the existence of higher classified data. In this second layer, it is possible to explicitly state which information is a cover story. We claim that this approach is free of ambiguity and is therefore better than using polyinstantiation. Finally, the purpose of the third layer is to derive, from the second layer, a consistent view of the multilevel database at each security level.

We show that there is a one to one correspondence between the second layer and the third layer. This means that, knowing how data are classified and which data are cover stories, we can provide each user with a consistent view of the multilevel database. Conversely, knowing what is the view of the multilevel database at each security level, we can derive how data are classified and which data correspond to cover stories. From a more practical point of view, this also means that there are two ways of implementing a multilevel database:

- storing how data are classified, which data are cover stories, and then deriving a view at each security level (implementation of second layer) or,

- storing a view at each security level and then deriving data classification and cover stories (implementation of third layer). We argue that implementing the third layer is generally easier. We illustrate it in the context of an object-oriented database.

The remainder of this paper is organized as follows. Section 2 proposes a model for a non-protected database (first layer of figure 1). Section 3 is a model for a multilevel database (second layer). In this section, two inference control rules are introduced: the first rule, called deductive channels control rule, is to prevent illegal inference of higher classified information; the second one, called signalling channels control rule, is to protect the existence of sensitive information. We also formally define the concept of cover story and provide means to explicitly state which data are cover stories. Section 4 is a model for a *MultiView* database (third layer). We provide axioms, which guarantee a one to one correspondence between the second and third layers. Sections 5 through 7 provide interpretation of the three layers in the case of an object-oriented database. In section 8, we suggest some ideas to implement a MultiView object-oriented database. Finally, section 9 concludes this paper.

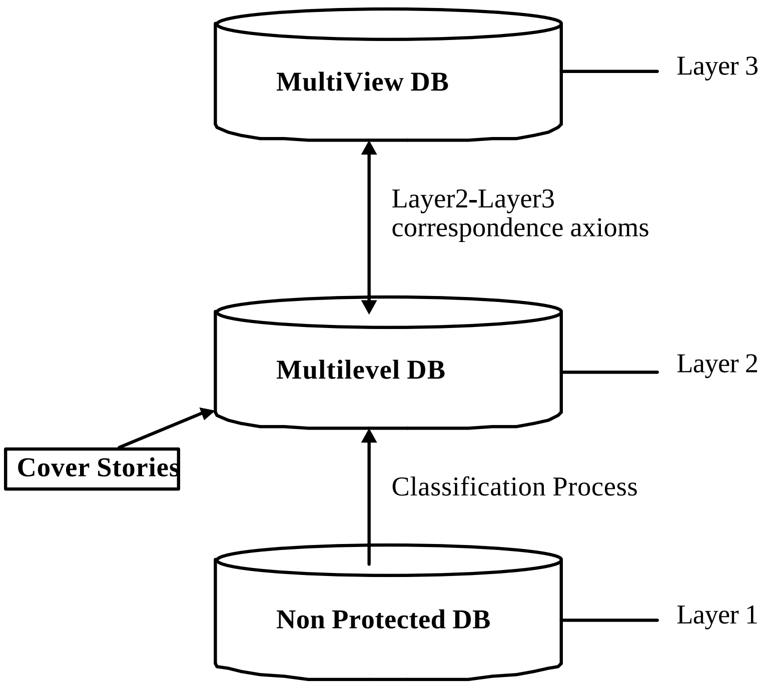

Figure 1. Three-layers model

2. Non-protected database

Mathematical logic has been used to formalize databases in two main directions. These directions are usually called the proof theoretics approach and the model theoretics approach. The former represents a database as a logical theory, the latter represents a database as an interpretation of a logical theory [Rei83]. In the remainder of this paper, we adopt the proof theoretics approach, that is, each database is associated with its logical theory *T*.

.

2.1. Language

The language L of the theory T is based on first-order logic with equality. Thus, in order to represent the non-protected Database we use first-order predicates.

Example:

Employee(*Dupont*) (that reads "Dupont is an employee")

Salary(*Dupont*,2000) (that reads "Dupont's salary is 2000")

2.2. Axiomatic

The axiomatic A of the theory T corresponds to the classical axiom schemata of first order logic with equality plus some proper axioms. These proper axioms are divided into two sets:

1. A set *DB* of *atomic facts* that represents the Database content

   Example:

   Employee(*Dupont*), *Salary*(*Dupont*,2000) …

2. A set *IC* of *integrity constraints*

   Example:

   $\forall x \forall y, Salary(x,y) \rightarrow Employee(x)$ (1)
   (that reads "If x has a salary then x is an employee")

   $\forall x, Employee(x) \rightarrow \exists y, Salary(x,y)$ (2)
   (that reads "each employee has a salary")

We shall assume that the database content enforces the set of integrity constraints, that is, the integrity constraints may be derived from the database content: $DB \vdash IC$

Among all possible types of integrity constraints, we distinguish two particular types of integrity constraints that will be useful for the remainder of this paper.

- ***Type 1***

$$\forall x_1 \ldots \forall x_n, P_1(x_1) \wedge \ldots \wedge P_n(x_n) \rightarrow Q(y) \quad (3)$$

where $x_1,\ldots,x_n$ are tuples of variables respectively compatible with the arity of predicates $P_1,\ldots,P_n$ and y is another tuple compatible with the arity of Q. We assume that each variable in tuple y appears in at least one of the tuples $x_1,\ldots,x_n$.

Integrity constraint (1) is of *Type 1*.

- ***Type 2***

$$\forall x_1 \ldots \forall x_n, P_1(x_1) \wedge \ldots \wedge P_n(x_n) \rightarrow \exists y, Q_1(y_1) \wedge \ldots \wedge Q_p(y_p) \quad (4)$$

where $x_1,\ldots,x_n$, $y_1,\ldots,y_p$ are tuples of variables respectively compatible with the arity of predicates $P_1,\ldots,P_n$, $Q_1,\ldots,Q_p$ and y is another tuple of variables. We assume that each variable in tuple y

appears in at least one of the tuples $y_1,\ldots,y_p$ and each variable in tuples $y_1,\ldots,y_p$ appears in at least one of the tuples $x_1,\ldots,x_n$, y.

Integrity constraint (2) is of *Type 2*.

Finally, notice that we could have considered the following form of integrity constraint as another particular *Type* 3:

$$\forall x_1 \ldots \forall x_n, P_1(x_1) \wedge \ldots \wedge P_n(x_n) \rightarrow Q_1(y_1) \wedge Q_2(y_2) \wedge \ldots \wedge Q_p(y_p)$$

where $x_1,\ldots,x_n$ are tuples of variables respectively compatible with the arity of predicates $P_1,\ldots,P_n$ and $y_1,\ldots,y_p$ are other tuples compatible with the arity of $Q_1,\ldots,Q_p$. Each variable in tuples $y_1,\ldots,y_p$ appears in at least one of the tuples $x_1,\ldots,x_n$.

However such an integrity constraint can be decomposed into several integrity constraints of *Type* 1:

$$\forall x_1 \ldots \forall x_n, P_1(x_1) \wedge \ldots \wedge P_n(x_n) \rightarrow Q_1(y_1)$$

$$\forall x_1 \ldots \forall x_n, P_1(x_1) \wedge \ldots \wedge P_n(x_n) \rightarrow Q_2(y_2)$$

…

$$\forall x_1 \ldots \forall x_n, P_1(x_1) \wedge \ldots \wedge P_n(x_n) \rightarrow Q_p(y_p)$$

3. Multilevel database

The central idea of our approach is to assign a classification level to any piece of information represented by an atomic formula in the non-protected database.

Let us extend the theory T, used to represent the non-protected database, into a theory T^c, used to represent the multilevel database.

3.1. Language

The language L^c used to represent the multilevel database is an extension of the language L developed in section 2.1.

We first extend our language to represent a finite set of classification levels. For this purpose, we shall use a one-place predicate *level*. The formula *level*(l) reads "l is a classification level". We assume that the set of levels is a lattice associated with a partial order relation ≤. Therefore, the *least upper bound* and *greatest lower bound* are defined. For this purpose, we shall use two functions *lub* and *glb*. If l_1 and l_2 are two security levels, then *lub*(l_1, l_2) and *glb*(l_1, l_2) are respectively the least upper bound and greatest lower bound of l_1 and l_2. There is a level that is also lower than all other levels, we denote it *Low* and a level that is higher than all other levels, we denote it *High*. The formula $l_1 \leq l_2$ reads "l_1 is dominated by l_2". Axioms associated with the predicates *Level* and ≤ are the following:

Only levels can be compared with ≤:

$$\forall l_1 \forall l_2, l_1 \leq l_2 \rightarrow level(l_1) \wedge level(l_2) \quad (5)$$

$\leq$ is transitive:

$$\forall l_1 \forall l_2 \forall l_3, l_1 \leq l_2 \wedge l_2 \leq l_3 \rightarrow l_1 \leq l_3 \qquad (6)$$

$\leq$ is reflexive:

$$\forall l_1, level(l_1) \rightarrow l_1 \leq l_1 \qquad (7)$$

$\leq$ is anti-symmetric:

$$\forall l_1 \forall l_2, l_1 \leq l_2 \wedge l_2 \leq l_1 \rightarrow l_1 = l_2 \qquad (8)$$

In the remainder of this paper we shall also use < and <>:

The formula $l_1 < l_2$ reads "l_1 is strictly dominated by l_2".

$$\forall l_1 \forall l_2, l_1 < l_2 \leftrightarrow l_1 \leq l_2 \wedge \neg(l_1 = l_2)$$

The formula $l_1 <> l_2$ reads "l_1 and l_2 are not comparable"

$$\forall l_1 \forall l_2, l_1 <> l_2 \leftrightarrow level(l_1) \wedge level(l_2) \wedge \neg(l_1 \leq l_2) \wedge \neg(l_2 \leq l_1)$$

For each predicate *P* of arity *n*, which is used to represent the non-protected database content, there is a predicate P^c of arity (n+1). Predicates P^c are used to represent the multilevel database content. Intuitively, if *l* is a classification level then $P^c(t_1, \ldots, t_n, l)$ reads "the piece of information $P(t_1, \ldots, t_n)$ is classified at level *l*"

Example:

In this example we assume that our set of security levels contains two security levels only {(U)nclassified, (S)ecret}(*High*=S and *Low*=U).

$Employee^c(Dupont, U)$
(that reads "the fact that Dupont is an employee is an unclassified information")

$Salary^c(Dupont, 2000, S)$
(that reads "the fact that Dupont's salary is 2000 is a secret information")

Notice that we might also apply the classification process to the predicates *Level* and $\leq$:

- We might consider a two place predicate $Level^c$ with the sentence $Level^c(l_1, l_2)$ which reads "the fact that l_1 is a security level is classified at level l_2". For example the fact $Level^c(Top\_Secret, Secret)$ means that the Top Secret security level itself is classified at the Secret level. As a consequence, only secret and top secret users can learn that there exists a Top Secret level and therefore, there may be some top-secret data in the multilevel database. However, for the sake of simplicity we do not include this possibility in our model. This implicitly means that we assume the extension of the predicate *Level* to be classified at the lowest level *Low*:

  $$\forall l_1 \forall l_2, level^c(l_1, l_2) \leftrightarrow level(l_1) \wedge l_2 = Low \qquad (9)$$

- We might consider a three place predicate $\leq^c$ with the sentence $\leq^c(l_1, l_2, l_3)$ that reads "the fact that level l_1 is dominated by level l_2 is classified at level l_3". We shall also not use this possibility in the remainder of this paper. This means that we assume the extension of the predicate ≤ to be classified at the lowest level *Low*:

$$\forall l_1 \forall l_2 \forall l_3, \leq^c (l_1, l_2, l_3) \leftrightarrow l_1 \leq l_2 \wedge l_3 = Low \qquad (10)$$

Finally, we assume the integrity constraints to be all implicitly classified at the lowest level *low*.

3.2. Axiomatic

The axiomatic A^c of the theory T^c includes the following axioms:

1. All axioms *A* of theory *T* (cf. section 2.2)
2. Axioms (5) to (10).
3. A set of *atomic facts* that represents the multilevel database content

   Example:

   $Employee^c(Dupont, U)$, $Salary^c(Dupont, 2000, S)$ …
4. An axiom schema which is used to specify the link between the non-protected database and the multilevel database:

   For each *n*-place predicate *P*, we have the following axioms:

   $$\forall t_1 \ldots \forall t_n, P(t_1, \ldots, t_n) \rightarrow \exists l, Level(l) \wedge P^c(t_1, \ldots, t_n, l) \qquad (11)$$

   This axiom says that any fact in the non-protected database is classified in the multilevel database.
5. An axiom schema saying that a fact cannot be associated with two distinct comparable security levels.

   $$\forall t_1 \ldots \forall t_n \forall l \forall l', P^c(t_1, \ldots, t_n, l) \wedge P^c(t_1, \ldots, t_n, l') \wedge l \leq l' \rightarrow l = l' \qquad (12)$$

   This axiom says that if a fact $P(t_1, \ldots, t_n)$ is classified with two levels *l* and *l'* and if *l'* dominates *l* then *l* and *l'* are the same. This axiom implicitly suggests that $P(t_1, \ldots, t_n)$ may be associated with more than one security level provided these security levels cannot be compared between each other.

   Example :

   Let us assume we have the following set of security levels:

   $\{U, C_1, C_2, S\}$

   The following partial order is defined in this set:

   $C_1 > U, C_2 > U, S > C_1, S > C_2$ and $C_1 <> C_2$

   According to axiom schema (12), $Employee^c(Dupont, U) \wedge Employee^c(Dupont, C_1)$ is not possible since $C_1 > U$ whereas $Employee^c(Dupont, C_1) \wedge Employee^c(Dupont, C_2)$ is possible since $C_1 <> C_2$.

3.3. Inference channels control theorems

Our objective in this section is to derive general constraint theorems that must be enforced when classifying the database content (for instance, what condition(s) must be satisfied when classifying *Salary*(*Dupont*,2000) at level *l* and *Employee*(*Dupont*) at level *l'* ?). These inference channels control theorems can be classified into the following two categories:

- *Deductive channels control theorems*. These theorems must be respected in order to prevent unauthorized disclosure of facts. They are derived from the integrity constraints of *Type 1*.
- *Signalling channels control theorems*. These theorems must be respected in order to prevent unauthorized disclosure of the existence of facts. They are derived from the integrity constraints of *Type 2*.

3.3.1. Deductive channels control theorems

When classifying any piece of information at a given classification level, the following deductive channels control rule must be enforced (this rule is added to the axiomatic A^c):

- ***Rule 1***

If $\forall x_1 \ldots \forall x_n, P_1(x_1) \wedge \ldots \wedge P_n(x_n) \rightarrow Q(y)$ is an integrity constraint of *Type 1* (see section 2.2) in the non-protected database, then the following constraint must be enforced in the multilevel database:

$$\forall x_1 \ldots \forall x_n \forall l_1 \ldots \forall l_n, P_1^c(x_1, l_1) \wedge \ldots \wedge P_n^c(x_n, l_n) \rightarrow \exists l, Q^c(y, l) \wedge l \leq lub(l_1, l_2, \ldots, l_n)$$

If this rule is not satisfied, then a subject cleared at level $lub(l_1,l_2, \ldots ,l_n)$ can access every $P_i(x_i)$ and use the axiom[1] (3) to derive $Q(y)$. In other words, if the classification of $Q(y)$ is not lower than or equal to $lub(l_1,l_2, \ldots ,l_n)$ then a deductive channel that enables prohibited information to be disclosed is created.

Example:

By directly applying *Rule 1* onto axiom (1) we can derive the following theorem:

$$\forall x \forall y \forall l \forall l', Salary^c(x, y, l') \rightarrow \exists l, Employee^c(x, l) \wedge l \leq l' \qquad (13)$$

This theorem says that if the fact "*y* is the salary of *x*" is classified at level *l'* then there must exist a security level *l* protecting the fact that "*x* is an employee" with *l'* dominating *l*.

3.3.2. Signalling channels control theorems

When classifying any piece of information at a given classification level, the following signalling channels control rule must be enforced in order to protect the *existence* of sensitive information (this rule is added to the axiomatic A^c):

---

[1] Recall that we implicitly assumed all integrity constraints to be classified at the lowest level *Low*. Now if we assume that integrity constraints might be classified then the conclusion of *Rule 1* would be modified as follows:

$$\forall x_1 \ldots \forall x_n \forall l_1 \ldots \forall l_n \forall l, P_1^c(x_1, l_1) \wedge \ldots \wedge P_n^c(x_n, l_n) \rightarrow \exists l, Q^c(y, l) \wedge l \leq lub(l_1, l_2, \ldots, l_n, l_c)$$

where l_c is the classification level of the integrity constraint used in the premise of *Rule 1.* If we assume that l_c=*Low* then we obtain *Rule 1*.

- ***Rule 2***

If $\forall x_1 \ldots \forall x_n, P_1(x_1) \wedge \ldots \wedge P_n(x_n) \rightarrow \exists y, Q_1(y_1) \wedge \ldots \wedge Q_p(y_p)$ is an integrity constraint of *Type 2* (see section 2.2) in the non-protected database, then the following constraint must be enforced in the multilevel database:

$$\forall x_1 \ldots \forall x_n \forall l_1 \ldots \forall l_n, P_1^c(x_1, l_1) \wedge \ldots \wedge P_n^c(x_n, l_n)$$
$$\rightarrow \exists y \exists l'_1 \ldots \exists l'_p, Q_1^c(y_1, l'_1) \wedge \ldots \wedge Q_p^c(y_p, l'_p) \wedge lub(l'_1, l'_2, \ldots, l'_p) \leq lub(l_1, l_2, \ldots, l_n)$$

If this rule is not satisfied, then a subject cleared at level $lub(l_1,l_2, \ldots ,l_n)$ can access every $P_i(x_i)$ and use the axiom[2] (4) to derive the existence of the facts $Q_1(y_1), \ldots , Q_p(y_p)$ whereas some of these $Q_i(y_i)$ are higher classified than $lub(l_1,l_2, \ldots ,l_n)$. Therefore, a signalling channel that enables the existence of prohibited information to be disclosed is created.

Example :

By directly applying *Rule 2* onto axiom (2) we can derive the following theorem:

$$\forall x \forall l, Employee^c(x,l) \rightarrow \exists y \exists l', Salary^c(x,y,l') \wedge l' \leq l \qquad (14)$$

This theorem says that "if the fact that x is an employee is classified at level l then x must have a salary y classified at a level l' dominated by l." Notice that in our particular example, by combining theorem (13) and theorem (14) we obtain the following interesting theorem:

$$\forall x \forall l, Employee^c(x,l) \rightarrow \exists y, Salary^c(x,y,l) \qquad (15)$$

This theorem says, "if the fact that x is an employee is classified at level l then there must exist a salary y for x classified at level l."

Proof:
Let us assume that $\exists x \exists l, Employee^c(x,l)$.
From (14) we can derive that $\exists y \exists l', Salary^c(x,y,l') \wedge l' \leq l$
From (13) we can derive that $\exists l'', Employee^c(x,l'') \wedge l'' \leq l'$.
From (12) we can derive that $l = l''$. Therefore since $l'' \leq l' \leq l$, we have $l = l'$
Thus, we have $\forall x \forall l, Employee^c(x,l) \rightarrow \exists y \exists l', Salary^c(x,y,l') \wedge l = l'$ which is equivalent to (15).

3.4. Cover Story

3.4.1. Definition of a Cover Story

Axiom (11) (cf. section 3.2) says that each fact of the non-protected database is associated with a security level in the corresponding multilevel database. Note that we might also state that each

[2] If we assume that integrity constraints might be classified then the conclusion of *Rule 2* would be modified as follows:

$$\forall x_1 \ldots \forall x_n \forall l_1 \ldots \forall l_n, P_1^c(x_1, l_1) \wedge \ldots \wedge P_n^c(x_n, l_n) \rightarrow \exists y \exists l'_1 \ldots \exists l'_p, Q_1^c(y_1, l'_1) \wedge \ldots \wedge Q_p^c(y_p, l'_p) \wedge lub(l'_1, l'_2, \ldots, l'_p) \leq lub(l_1, l_2, \ldots, l_n, l_c)$$

where l_c is the classification level of the integrity constraint used in the premise of *Rule 2*. If we assume that l_c=*Low* then we obtain *Rule 2*.

classified fact of the multilevel database also belongs to the non-protected database. This is captured by the following sentence:

$$\forall t_1 \ldots \forall t_n \forall l, P^c(t_1,\ldots,t_n,l) \rightarrow level(l) \wedge P(t_1,\ldots,t_n) \qquad (16)$$

This formula says that the extension of the P^c predicates is obtained by classifying facts belonging to the extension of the P predicates. However, it is sometimes necessary to introduce in the multilevel database some facts that do not belong to the non-protected database. For instance, let us consider our example of section 3.1:

$Employee^c(Dupont,U) \wedge Salary^c(Dupont,2000,S)$

This means, the fact "Dupont is an employee" is unclassified and the fact "2000 is Dupont's salary" is secret. Theorem (15) says that a value for Dupont's salary must also be provided at the unclassified level. For example, we may have:

$Salary^c(Dupont,1500,U)$

Semantically, value 1500 can have two interpretations:

- Employee Dupont has two salaries, one that is secret and equal to 2000 and one that is unclassified and is equal to 1500.
- 1500 is a *cover story* that is, a *lie* unclassified users are provided with. This lie is used to hide the existence of the more sensitive data 2000.

Note that, in this example, 1500 cannot be interpreted as a second salary if one have the following integrity constraint:

$$\forall x \forall y, Salary(x,y) \wedge Salary(x,y') \rightarrow y = y' \qquad (17)$$

Indeed in that case, employee Dupont cannot have two distinct salaries. If (17) is stated as an integrity constraint then 1500 is automatically interpreted as a cover story.

We formally define a cover story as follows:

- $P(c_1,\ldots,c_n)$ is a cover story iff (16) is not satisfied that is, there exists a level l such as $P^c(c_1,\ldots,c_n,l)$ belongs to the multilevel database and $P(c_1,\ldots,c_n)$ does not belong to the non-protected database.

In our example given above, $Employee(Dupont)$ and $Salary(Dupont,2000)$ are not cover stories since $Employee^c(Dupont,U)$ and $Salary^c(Dupont,2000,S)$ belong to the multilevel database and $Employee(Dupont)$ and $Salary(Dupont,2000)$ belong to the non-protected database (cf. section 2.1). On the contrary, $Salary(Dupont,1500)$ is a cover story since $Salary^c(Dupont,1500,U)$ belongs to the multilevel database but $Salary(Dupont,1500)$ does not belong to the non-protected database.

3.4.2. Honest predicates versus Liar predicates

In the previous section 3.4.1, we have seen that the $Salary^c$ predicate may contain some cover stories. Every predicate P^c may actually contain some cover stories. For instance, let us consider the following multilevel database:

$Employee^c(Dupont, U) \wedge Employee^c(Durand, U) \wedge Salary^c(Dupont, 2000, S)$

Now let us assume that the corresponding non-protected database is the following:

$Employee(Dupont) \wedge Salary(Dupont, 2000)$

Employee(*Durand*) does not belong to the non-protected database whereas $Employee^c(Durand,U)$ belongs to the multilevel database. According to the definition, *Employee(Durand)* is a cover story. The fact that Durand is an employee is a lie.

However, if there is no need to introduce a cover story in $Employee^c$, then the Security Administrator may decide that (16) is an axiom for $Employee^c$.

Thus, we define an *Honest predicate* as follows:

- A predicate P^c that satisfies (16) is an Honest predicate.

We define a *Liar predicate* as follows:

- A predicate P^c that does not satisfy (16) is a Liar predicate.

The Security Administrator is completely free of deciding whether a predicate P^c is an Honest or a Liar predicate. However depending on the predicate, such decision may have some consequences on security (see section 3.4.4 below).

3.4.3. Cover Story management

Let us consider a secret user seeing the following multilevel database:

$Employee^c(Dupont, U) \wedge Salary^c(Dupont, 2000, S) \wedge Salary^c(Dupont, 1500, U)$

If we assume that (17) is not an integrity constraint then this secret user may find two possible interpretations for $Salary^c(Dupont,1500,U)$ (cf. section 3.4.1). He may think that 1500 is the value of a second salary for Dupont or he may think that 1500 is a cover story used to hide from unclassified users the actual secret value 2000. However, he has no means to know which of the two interpretations is the correct one. Therefore, in order to make things clear for the secret user there must be something in the multilevel database telling him whether 1500 is a cover story or a second salary for Dupont. For this purpose, we propose to extend the theory T^c into a theory T^{cs} as follows:

- The language L^{cs} is an extension of L^c (cf. section 3.1):

For each Liar predicate P^c of arity n+1, there is a predicate P^{cs} of arity n+1. Predicates P^{cs} are used to represent the cover stories. Intuitively, if l is a classification level then $P^{cs}(t_1,\ldots,t_n,l)$ reads "the fact that $P(t_1,\ldots,t_n)$ is a cover story is classified at level l".

<u>Example</u>:

$Salary^{cs}(Dupont, 1500, S)$ (that reads "the fact that *Salary*(*Dupont*,1500) is a cover story is classified at level S").

- The axiomatic A^{cs} of the theory T^{cs} includes the following axioms:

1. All axioms A^c of theory T^c (cf. section 3.2 + *Rule 1* + *Rule 2*)
2. Axiom (16) for all Honest predicates.
3. A set of *atomic facts* that represents the cover stories.

   Example : $Salary^{cs}(Dupont, 1500, S)$
4. A set of axioms which are used to specify the links between the P, P^c and P^{cs} predicates:

   For each n-place predicate P^{cs}, we have the following axiom:

   $$\forall t_1 \ldots \forall t_n \forall l, P^{cs}(t_1, \ldots, t_n, l) \rightarrow \exists l', l' < l \wedge P^c(t_1, \ldots, t_n, l') \qquad (18)$$

   Axiom (18) says that if the fact that $P(t_1,\ldots,t_n)$ is a cover story is classified at level l then there exists a level l' strictly dominated by l such that $P(t_1,\ldots,t_n)$ is classified at level l'.

   Note that from the axioms (18) and (12) we can derive the following theorem:

   $$\forall t_1 \ldots \forall t_n \forall l, P^{cs}(t_1, \ldots, t_n, l) \rightarrow \neg P^c(t_1, \ldots, t_n, l) \qquad (19)$$

   Theorem (19) says that if the fact that $P(t_1,\ldots,t_n)$ is a cover story is classified at level l then $P(t_1,\ldots,t_n)$ cannot be classified at level l.

   Proof:

   Let us assume that $\exists t_1 \ldots \exists t_n \exists l, P^{cs}(t_1, \ldots, t_n, l)$.
   From (18) we can derive that $\exists l', P^c(t_1, \ldots, t_n, l') \wedge l' < l$
   Axiom (12) can be rewritten as $\forall t_1 \ldots \forall t_n \forall l \forall l', P^c(t_1, \ldots, t_n, l) \wedge l < l' \rightarrow \neg P^c(t_1, \ldots, t_n, l')$
   Therefore, from (12) and from $P^c(t_1, \ldots, t_n, l') \wedge l' < l$ we can derive that $\neg P^c(t_1, \ldots, t_n, l)$.

   For each n-place predicate P^{cs}, we have the following axiom:

   $$\forall t_1 \ldots \forall t_n \forall l, P^c(t_1, \ldots, t_n, l) \wedge \neg P(t_1, \ldots, t_n) \rightarrow \exists l', l < l' \wedge P^{cs}(t_1, \ldots, t_n, l') \qquad (20)$$

   Axiom (20) says that if $P^c(t_1,\ldots,t_n,l)$ belongs to the multilevel database and if $P(t_1,\ldots,t_n)$ does not belong to the non-protected database then $P(t_1,\ldots,t_n)$ is classified as a cover story at a level l' strictly dominating l. Note that this axiom can also be rewritten as follows:

   $$\forall t_1 \ldots \forall t_n \forall l, P^c(t_1, \ldots, t_n, l) \rightarrow (P(t_1, \ldots, t_n) \vee \exists l', l < l' \wedge P^{cs}(t_1, \ldots, t_n, l')) \qquad (20bis)$$

Finally, A^{cs} is also extended with the following rule:

- ***Rule 3***

In the following $a_1,\ldots,a_n$ are tuples of constants respectively compatible with the arity of predicates $P_1,\ldots,P_n$:

If $\{P_1(a_1), P_2(a_2),..., P_n(a_n)\}$is a minimal inconsistent[3] set and

if we have $P_1^c(a_1,l_1) \wedge \ldots \wedge P_n^c(a_n,l_n) \wedge l = \mathrm{lub}(l_1,\ldots,l_n)$,

then we have:

$P_1^{cs}(a_1,l) \vee \ldots \vee P_n^{cs}(a_n,l)$

This means, at least one of the $P_i(a_i)$ is a cover story and the fact that it is a cover story is classified at level *l*.

In the following, we give simple examples of multilevel databases containing cover stories:

Example 1:

This first example shows how *Rule 3* can sometimes allow us to implicitly derive that an atomic fact is actually a cover story:

Let us consider the following database and let us assume that $Salary^{cs}$ is a Liar predicate:

$Employee^c(Dupont,U) \wedge Salary^c(Dupont,2000,S) \wedge Salary^c(Dupont,1500,U)$

and let us assume that a person's salary must be unique, that is, (17) is an integrity constraint.

The set $\{Salary(Dupont,1500), Salary(Dupont,2000)\}$is then minimal inconsistent.

Moreover, since we have $Salary^c(Dupont,2000,S) \wedge Salary^c(Dupont,1500,U)$, using *Rule 3* we can derive $Salary^{cs}(Dupont,2000,S) \vee Salary^{cs}(Dupont,1500,S)$.

$Salary^{cs}(Dupont,2000,S)$ is not possible since it contradicts the theorem (19). Thus, $Salary^{cs}(Dupont,1500,S)$ can be derived.

On the contrary, if (17) is not an integrity constraint then $\{Salary(Dupont,1500), Salary(Dupont,2000)\}$ is consistent. Therefore, if $Salary(Dupont,1500)$ is really a cover story (and not a second salary for Dupont) then the Security Administrator must explicitly introduce in the multilevel database the fact $Salary^{cs}(Dupont,1500,S)$.

Example 2:

In this example both predicates $Employee^c$ and $Salary^c$ are Liar predicates

Let us consider the following multilevel database and let us assume that a person can have several salaries, that is, (17) is not an integrity constraint:

$Employee^c(Dupont,U)$

$\wedge Salary^c(Dupont,2000,S) \wedge Salary^c(Dupont,1500,U) \wedge Salary^c(Dupont,1000,U)$

$\wedge Employee^c(Durand,U) \wedge Salary^c(Durand,1000,U)$

$\wedge Employee^{cs}(Durand,S) \wedge Salary^{cs}(Dupont,1500,S) \wedge Salary^{cs}(Durand,1000,S)$

[3] A set *I* is minimal inconsistent iff *I* is inconsistent and every strict subset of *I* is not inconsistent. *I* is inconsistent iff $I \vdash \neg IC$ with *IC* being the set of integrity constraints (see section 2.2) (If $IC = \{c_1, c_2,..., c_n\}$ with each c_i being an integrity constraint then $\neg IC = \neg(c_1 \wedge c_2 \wedge \ldots \wedge c_n) = \neg c_1 \vee \neg c_2 \vee \ldots \vee \neg c_n$).

This database satisfies axioms and theorems (12) to (15) plus (18) and (19)

In this example we have three cover stories:

1. *Employee*(*Durand*)
2. *Salary*(*Durand*,1000)
3. *Salary*(*Dupont*, 1500)

- Interpretation of data made by unclassified users is as follows:

Unclassified users cannot see any cover story. Therefore, they think that $Employee^c$ and $Salary^c$ are Honest predicates. They learn that Dupont and Durand are employees. They learn that Dupont has two salaries, 1000 and 1500. Finally they learn that Durand's salary is 1000.

- Interpretation of data made by secret users is as follows:

Secret users can see all cover stories. They learn that the facts "Durand is an employee" and "Durand's salary is 1000" are lies. They learn that Dupont has two actual salaries, 1000 and 2000 and they learn that Dupont's third salary, 1500, is a lie.

Example 3:

Let us assume we have the following set of security levels:

$\{U, C_1, C_2, S\}$

The following partial order is defined in this set:

$C_1 > U, C_2 > U, S > C_1, S > C_2$ and $C_1 <> C_2$

Let us also assume that a person's salary is unique, that is, (17) is an integrity constraint.

$$Employee^c(Dupont, C_1) \wedge Employee^c(Dupont, C_2)$$
$$\wedge Salary^c(Dupont, 2000, C_1) \wedge Salary^c(Dupont, 1500, C_2)$$
$$\wedge Salary^{cs}(Dupont, 1500, S)$$

This database satisfies axioms and theorems (12) to (15) plus (18) and (19)

In this example we have one cover story:

Salary(*Dupont*, 1500)

- Unclassified users see an empty database.
- Interpretation of data made by C_1 users is as follows:

C_1 users cannot see the cover story. Therefore, they think that $Salary^c$ is an Honest predicate. They learn that Dupont is an employee and that his salary is 1500.

- Interpretation of data made by C_2 users is as follows:

C_2 users cannot see the cover story. Therefore, they also think that $Salary^c$ is an Honest predicate. They learn that Dupont is an employee and that his salary is 2000.

- Interpretation of data made by secret users is as follows:

Secret users can see the cover story. They learn that Dupont is an employee. They can see both salaries 1500 and 2000 but they know that 1500 is a lie.

3.4.4. Avoiding Cover Stories

Let us consider theorem (15) again. It says that a subject who is permitted to observe the existence of an employee must be provided with a value for the salary of this employee.

Let us consider now the following database and let us assume that (17) is an integrity constraint.

$$Employee^c(Dupont,U) \wedge Salary^c(Dupont,2000,S)$$

As seen in section 3.4.1, a value for the salary of Dupont must be provided at the unclassified level in order to avoid a signalling channel, for instance:

$$Salary^c(Dupont,1500,U)$$

However, this corresponds to inserting a lie in the database. The security administrator may consider that inserting a lie is not compatible with the database integrity. In this case, the security administrator has three solutions:

1. Leave the situation as such, that is, accept the theorem (15) to be violated. This solution is not very satisfactory, since a signalling channel is left open. A malicious user can use this channel to build a bad covert channel, which provides unclassified users with high-classified information they are not permitted to know. Therefore, we will not consider this solution in the remainder of this paper.

2. A second solution to prevent the insertion of cover stories into $Salary^c$ is to state that the security level protecting the salary of an employee must be equal to the security level protecting the existence of the employee himself:

   $$\forall x \forall y \forall l \forall l', Salary^c(x,y,l) \wedge Employee^c(x,l') \rightarrow l = l'$$

   Thanks to this axiom, users who are permitted to observe an employee classified at level l are prevented to insert a salary value for this employee, at a level strictly dominating l.

   This solution reduces considerably the expressive power of the multilevel security policy. However, it can be used for some particular predicates (cf. for example the predicate $Type^c$ in section 6.4).

3. Another possible solution to prevent the insertion of a cover story was first suggested by [SJ92]. It consists in using a special symbol denoted by *Restricted*. Intuitively, *Restricted* means that the value exists but is higher classified. This is captured in the following sentence:

   $$\forall x \forall l, Salary^c(x,Restricted,l) \rightarrow \exists y \exists l', Salary^c(x,y,l') \wedge y \neq Restricted \wedge l < l'$$

   Notice that this logical sentence represents the first attempt ever made to formally define the semantics of this special value *Restricted*.

   However, we cannot satisfy theorem (15) by inserting $Salary^c(Dupont,Restricted,U)$ in the multilevel database. Indeed, inserting $Salary^c(Dupont,Restricted,U)$ in the database cannot be done without inserting $Salary^{cs}(Dupont, Restricted, S)$(otherwise (17) is violated) which sounds contradictory since *Restricted* is the opposite of a cover story. Therefore, in this particular case, $Salary^{c}(Dupont, Restricted, U)$ must not be seen as an atomic fact of the multilevel database but as a *notation* for the following sentence:

$\exists y \exists l, Salary^{c}(Dupont, y, l) \wedge y \neq Restricted \wedge U < l$

which means "there exists a salary value for Dupont but you are not permitted to learn it". Such a sentence is classified at the level where the *Restricted* value should be inserted. In our example, the above sentence is unclassified.

Notice that this solution does not prevent theorem (15) to be violated. Its main advantage over solution 1 is that the special *Restricted* value (that is, the sentence $\exists y \exists l, Salary^{c}(Dupont, y, l) \wedge y \neq Restricted \wedge U < l$) explicitly tells the unclassified users that "there exists a salary value for Dupont but you are not permitted to learn it". Therefore the situation is no longer ambiguous and is consistent with the security administrator policy. In particular, it is no longer possible for a malicious user to build a covert channel.

3.4.5. Comparison with Polyinstantiation

In the literature, the technique of polyinstantiation is generally used to manage cover stories. However, we claim that this technique is unclear and leads to some ambiguities in the interpretation of data. Let us describe some problems that are related to the technique of polyinstantiation:

1. Ambiguous interpretation of data

Consider the following two examples:

Example a:

Let us assume that we have the following set of security levels:

$\{U, S\}$

Consider the following database:

$Employee^{c}(Dupont, U) \wedge Salary^{c}(Dupont, 2000, S)$

As shown in section 3.4.1, a value for Dupont's salary must also be provided at the unclassified level in order to avoid a signalling channel, for instance:

$Salary^{c}(Dupont, 1500, U)$

The value of Dupont's salary becomes then a polyinstantiated set of two classified values $\{(2000,S),(1500,U)\}$. In the literature, the interpretation that is generally made about these values is that 2000 is Dupont's correct salary and 1500 is a cover story that unclassified users are provided with. However, as shown in section 3.4.1, this interpretation implicitly assumes that (17) is an integrity constraint. Indeed if (17) is not an integrity constraint then there is another possible interpretation (cf. section 3.4.1): Dupont has two salaries, one secret 2000 and one unclassified 1500. If (17) is not an integrity constraint, then the technique of polyinstantiation does not provide us with any means to know without any ambiguities which of the two possible interpretations is the correct one. On the contrary, with the technique presented in section 3.4.3 we know that if $Salary^{cs}(Dupont, 1500, S)$ belongs to the database then 1500 is a cover story otherwise it is Dupont's second salary.

<u>Example b:</u>

In this example, we assume we have the following set of security levels:

$\{U, C_1, C_2, S\}$

The following partial order is defined in this set:

$C_1 > U, C_2 > U, S > C_1, S > C_2$ and $C_1 <> C_2$

Let us also assume that (17) is an integrity constraint.

Consider the following database:

$$Employee^c(Dupont, C_1) \wedge Employee^c(Dupont, C_2)$$
$$\wedge Salary^c(Dupont, 2000, C_1) \wedge Salary^c(Dupont, 1500, C_2)$$

Dupont's salary is a polyinstantiated set of two classified values $\{(2000, C_1),(1500, C_2)\}$.

With the polyinstantiation technique;

- if (17) is not an integrity constraint then a secret user cannot interpret these two values. He has no means to determine whether 2000 and 1500 are two distinct salaries for Dupont or whether one of these two values is a cover story.
- if (17) is an integrity constraint then the only thing a secret user can learn is that one of these two values is a cover story, but this secret user is unable to say which one.

On the contrary, with the technique presented in section 3.4.3 we know that if $Salary^{cs}(Dupont, 1500, S)$ belongs to the database then 1500 is a cover story otherwise it is Dupont's second salary.

2. <u>Poor expressive power</u>

The technique of polyinstantiation allows us to introduce a lie in a predicate that describes an entity attribute, but it does not provide us with any solution to introduce a cover story in a predicate that describes an entity itself. For example, with the polyinstantiation technique, it is possible to introduce a cover story in $Salary^c$ (this predicate describes the salary attribute of the employee entity) but it is not possible to introduce a lie in $Employee^c$ (this predicate describes the employee entity). As a matter of fact, introducing a lie saying that Durand is an employee is impossible with the polyinstantiation technique. With the new technique we propose, we show in section 3.4.3 how we can easily do this.

3. <u>Entity polyinstantiation</u>

[Lun91] describes this type of polyinstantiation in the context of relational databases. In this context, entity polyinstantiation occurs when a relation contains multiple tuples with the same apparent primary key values, but having different classifications for this apparent primary key.

In this paper, If we adapt this definition to our logical representation of a multilevel database then we might obtain the following:

There is entity polyinstantiation when an atomic fact is associated with at least two comparable security levels, that is, each time axiom (12) is violated.

Example

Let us assume we have the following database

$Employee^c(Dupont, U) \wedge Employee^c(Dupont, S)$

Lunt [Lun91] suggests the following interpretation: there exist two distinct "Dupont" in the external world, one whose existence is secret and one whose existence is unclassified. We reject this interpretation since it makes the interpretation of other data more difficult. For instance, let us assume that $Salary^c(Dupont, 2000, S)$ belongs to the multilevel database. Shall we consider that this is the salary of "Dupont unclassified" or the salary of "Dupont secret" ?

We claim that if there exist two distinct "Dupont" in the external world, then they must be distinguished by their name. For instance we may give the name Dupont_U to "Dupont unclassified" and the name Dupont_S to "Dupont secret"[4]. Our database becomes then as follows:

$Employee^c(Dupont\_U, U) \wedge Employee^c(Dupont\_S, S)$

which is compatible with axiom (12).

Nevertheless, recall that axiom (12) implicitly states that it is possible to have a same atomic fact associated with more than one security level, provided these security levels cannot be compared between each other.

Example:

In this example, let us assume we have the following set of security levels.

$\{U, C_1, C_2, S\}$ with the following partial order defined in it:

$C_1 > U, C_2 > U, S > C_1, S > C_2$ and $C_1 <> C_2$

Let us assume we have the following database:

$Employee^c(Dupont, C_1) \wedge Employee^c(Dupont, C_2)$

We claim that this situation has nothing to do with "entity polyinstantiation". Our interpretation of this is that the fact "Dupont is an employee" is classified at two incomparable security levels C_1 and C_2.

[4] In [SJ93], Sandhu and Jajodia suggest to cope with this problem of "entity polyinstantiation" by inserting in the database expressions like Salary(Dupont,S,2500,S) and Salary(Dupont,U,2000,S). Salary(Dupont,S,2500,S) represents the (secret) salary of "Dupont secret" whereas Salary(Dupont,U,2000,S) represents the (secret) salary of "Dupont unclassified". However this technique creates an ambiguity: security levels are sometimes used to identify the sensitivity of data (eg. S in Salary(Dupont,U,2000,S)) and sometimes used to identify entities (eg. U in Salary(Dupont,U,2000,S)).

4. MultiView database

The logical representation of a multilevel database given in section 3 is a complete one. This logical representation provides us with a multilevel policy offering a great expressive power. It allows us to manage fine classification grains and cover stories. This logical approach can be used to model any kind of multilevel databases, such as multilevel relational databases and, as we will see in section 6 and 7, multilevel object-oriented databases. This logical approach defines precisely the concept of “classified data” by providing a clear semantics for each association between a grain of classification and a security level. This logical approach also precisely defines the concept of cover story without using the confusing technique of polyinstantiation.

However, this formalization of a multilevel database does not clearly indicate what part of the multilevel database a user cleared at level *l* can see. The purpose of this section is to refine this logical representation in order to clearly and formally define what is the part of the multilevel database that each user is supposed to observe[5].

Intuitively, we decompose a *n*-level logical database into *n* views. The whole multilevel database is then seen as a collection of views. We call this set of views MultiView Database. Schematically, each view is associated with a given level of classification and contains a consistent and complete set of data whose classifications are lower than or equal to the level of the view. A user cleared at level *l* can then observe every view whose security level is dominated by *l*.

In section 8, we shall describe the implementation of a MultiView database for the particular case of an object-oriented database.

4.1. Language

The language L^v which is used to represent a MultiView Database is an extension of the language L^{cs} developed in section 3.4.3. For each n+1-place predicate P^c, there is a n+1-place predicate P^v. Predicates P^v are used to represent the MultiView database content. Intuitively, if *l* is a classification level, then $P^v(t_1, \dots, t_n, l)$ reads “the piece of information $P(t_1, \dots, t_n)$ belongs to the view classified at level *l*”[6].

Example:

$Employee^v(Dupont, U)$
(that reads “the fact that Dupont is an employee belongs to the unclassified view”)

$Employee^v(Dupont, S)$
(that reads “the fact that Dupont is an employee belongs to the secret view”)

4.2. Axiomatic

The axiomatic A^v of the theory T^v includes the following axioms:

[5] The approach presented in this section matches with the “Trusted approach” suggested in [Cup96].

[6] Notice the difference of interpretation between $P^c(t_1, \dots, t_n, l)$ and $P^v(t_1, \dots, t_n, l)$. $P^c(t_1, \dots, t_n, l)$ means “the piece of information $P(t_1, \dots, t_n)$ is classified at level *l* whereas $P^v(t_1, \dots, t_n, l)$ reads “the piece of information $P(t_1, \dots, t_n)$ *belongs to the view classified at level l”*.

1. All axioms A^{cs} of theory T^{cs} (cf. section 3.4.3) + *Rule 1* + *Rule 2* + *Rule 3*.
2. A set of *atomic facts* that represents the MultiView database content:

   Example:

   $Employee^v(Dupont,U)$, $Employee^v(Dupont,S)$ …

3. A set of axioms which is used to specify the links between the multilevel database and the MultiView database:

   For each Honest predicate P^c, we have the following axiom:

   $$\forall t_1 \ldots \forall t_n \forall l \forall l', P^c(t_1,\ldots,t_n,l) \wedge l \leq l' \rightarrow P^v(t_1,\ldots,t_n,l') \qquad (21)$$

   This axiom says that a fact classified at level *l* belongs to every view whose associated security level dominates *l*.

   For each Liar predicate P^c, we have the following two axioms:

   $$\forall t_1 \ldots \forall t_n \forall l, P^c(t_1,\ldots,t_n,l) \rightarrow P^v(t_1,\ldots,t_n,l) \qquad (22)$$

   This axiom says that a fact classified at level *l* belongs to the view of level *l*. However, since this fact belongs to a Liar predicate it might be a cover story, therefore we cannot be sure that it also belongs to the views of level strictly dominating *l*. This is captured by axiom (23):

   $$\forall t_1 \ldots \forall t_n \forall l \forall l', P^v(t_1,\ldots,t_n,l) \wedge Justbelow(l,l') \wedge \neg P^{cs}(t_1,\ldots,t_n,l') \rightarrow P^v(t_1,\ldots,t_n,l') \qquad (23)$$

   where *Justbelow* is defined as follows:

   $$\forall l \forall l', Justbelow(l,l') \leftrightarrow l < l' \wedge \neg(\exists l'', l < l'' \wedge l'' < l')$$

   Axiom (23) says that if a fact belongs to a view of level *l* and if this fact is not referred as a cover story in a level *l'* such as *l* is just below *l'*, then this fact also belongs to the view of level *l'*. Notice that axiom (23) can be rewritten as follows:

   $$\forall t_1 \ldots \forall t_n \forall l \forall l', P^v(t_1,\ldots,t_n,l) \wedge Justbelow(l,l') \wedge \neg P^v(t_1,\ldots,t_n,l') \rightarrow P^{cs}(t_1,\ldots,t_n,l') \qquad (23bis)$$

   It says that if a fact belongs to a view at level *l* and if there is a level *l'* such as *l* is just below *l'* and this fact does not belong to the view at level *l'*, then this fact is a cover story of level *l'*.

   Finally, the following axiom applies to both Liar and Honest predicates:

   $$\forall t_1 \ldots \forall t_n \forall l, P^v(t_1,\ldots,t_n,l) \wedge (\forall l', Justbelow(l',l) \rightarrow \neg P^v(t_1,\ldots,t_n,l')) \rightarrow P^c(t_1,\ldots,t_n,l) \qquad (24)$$

   This axiom says that if a fact belongs to the view at level *l* and if this fact does not belong to any view classified with a level just below the level *l*, then this fact is classified at level *l* in the multilevel database. Note that this axiom is still valid if *l* = *Low*.

Note also that axioms (21) to (24) establish a one to one correspondence between a MultiView database and a multilevel database:

- If we make the *Closed World Assumption (CWA)* [Rei78,Rei83] on the extensions of the P^v predicates[7], then from the facts belonging to the P^v predicates we can derive the facts belonging to the P^c and P^{cs} predicates, using axioms (23b) and (24).
- If we make the *CWA* on the extensions of the P^{cs} predicates[8], then from the facts belonging to the P^c and P^{cs} predicates we can derive the facts belonging to the P^v predicates, using axioms (21), (22) and (23).

Example:

In this example, we assume that we have the following set of security levels:

$\{U, C_1, C_2, S\}$

The following partial order is defined in it:

$C_1 > U, C_2 > U, S > C_1, S > C_2$ and $C_1 <> C_2$

Consider the following *multilevel* database content:

$$Employee^c(Dupont, C_1) \wedge Employee^c(Dupont, C_2)$$
$$\wedge Salary^c(Dupont, 2000, C_1) \wedge Salary^c(Dupont, 1500, C_2)$$
$$\wedge Salary^{cs}(Dupont, 1500, S)$$

Consider the following *MultiView* database content:

$$Employee^v(Dupont, C_1) \wedge Employee^v(Dupont, C_2) \wedge Employee^v(Dupont, S)$$
$$\wedge Salary^v(Dupont, 2000, C_1) \wedge Salary^v(Dupont, 1500, C_2) \wedge Salary^v(Dupont, 2000, S)$$

Let us show that the MultiView database can be derived from the multilevel database:

- From $Employee^c(Dupont,C_1)$ and (21) we can derive $Employee^v(Dupont,C_1)$ and $Employee^v(Dupont, S)$
- From $Employee^c(Dupont,C_2)$ and (21) we can derive $Employee^v(Dupont,C_2)$
- From $Salary^c(Dupont,1500,C_1)$ and (22) we can derive $Salary^v(Dupont,1500,C_1)$
- From $Salary^c(Dupont,2000,C_2)$ and (22) we can derive $Salary^v(Dupont,2000,C_2)$
- From $Salary^c(Dupont,2000,C_2)$, Ø$Salary^{cs}(Dupont,2000,S)$ (derived from the *CWA* made on the P^{cs} predicates) and (23) then we can derive $Salary^v(Dupont,2000,S)$

Let us now show that the multilevel database can be derived from the MultiView database:

- From $Employee^v(Dupont,C_1)$ and (24) we can derive $Employee^c(Dupont,C_1)$
- From $Employee^v(Dupont,C_2)$ and (24) we can derive $Employee^c(Dupont,C_2)$

[7] For each predicate P^V of arity n+1, if $P^V(t_1,t_2, \ldots,t_n,l)$ is not derivable from the theory T^V, then we shall assume that Ø$P^V(t_1,t_2,\ldots,t_n,l)$ is derivable from T^V.

[8] For each predicate P^{CS} of arity n+1, if $P^{CS}(t_1,t_2, \ldots,t_n,l)$ is not derivable from the theory T^V, then we shall assume that Ø$P^{CS}(t_1,t_2,\ldots,t_n,l)$ is derivable from T^V.

- From $Salary^{v}(Dupont,1500,C_1)$ and (24) we can derive $Salary^{c}(Dupont,1500,C_1)$
- From $Salary^{v}(Dupont,2000,C_2)$ and (24) we can derive $Salary^{c}(Dupont,2000,C_2)$
- From $Salary^{v}(Dupont,1500,C_1)$, $\emptyset Salary^{v}(Dupont,1500,S)$ (derived from the *CWA* made on the P^{v} predicates) and (23b) then we can derive $Salary^{cs}(Dupont,1500,S)$

5. Non-protected object-oriented database

First-order logic has been largely used to formalize relational databases. In the context of object-oriented databases, it is less obvious to use first-order logic as a formal language. It seems that several concepts such as methods require higher order logics or modal logics (dynamic logic for instance) to be properly formalized [Wie91]. However, our goal is not to develop such a complete formalism. We simply need to formally represent the main object-oriented concepts in order to apply the classification process (see section 6 below).

Using the approach presented in section 2, we propose a list of nine predicates allowing us to represent the content of any object-oriented database. Then, we introduce a list of integrity constraints inherent to the object paradigm. These integrity constraints must be respected in any object-oriented database.

5.1. Language

The language *L* of the theory *T* contains nine predicates that allow us to represent the content of any object-oriented database:

- Two one place predicates *Object* and *Class*.
- Five two place predicates *CA*, *OA*, *Method*, *Instance* and *Isa*.
- Two three place predicates *Val* and *Type*.

Intuitively, these predicates must be interpreted as follows:

- *Object(o):* “*o* is an object”.
- *Class(c):* “*c* is a class”.
- *CA(c,a):* “*a* is an attribute of class *c*”.
- *OA(o,a):* “*a* is an attribute of object *o*”.
- *Method(c,m):* “*m* is a method of class *c*”.
- *Instance(o,c):* “*o* is an instance of class *c*”.
- *Isa(c,c’):* “*c* is a subclass of *c’* ”.
- *Val(o,a,v):* “*v* is the value of attribute *a* in object *o*”.
- *Type(c,a,t)*: “*t* is the type of attribute *a* in class *c*”.

5.2. Axiomatic

As defined in section 2.2, the axiomatic *A* of the theory *T* contains the following axioms:

1. A set *DB* of *atomic facts* that represents the object-oriented database content.

   Example:

   Class(*Supersonic_Aircraft*)

(that reads “Supersonic_Aircraft is a class”)

Object(*FireFox*)
(that reads “FireFox is an object”)

CA(*Supersonic_Aircraft*,*Speed*)
(that reads “Speed is an attribute of Supersonic_Aircraft”)

Instance(*FireFox*,*Supersonic_Aircraft*)
(that reads “FireFox is an instance of Supersonic_Aircraft”)

Val(*FireFox*,*Speed*,*mach* 6)
(that reads “mach 6 is the value of the Speed of FireFox”)

$\forall a \forall c, CA(c,a) \rightarrow Class(c)$	(25)	$\forall a \forall o, OA(o,a) \rightarrow Object(o)$	(26)
$\forall m \forall c, Method(c,m) \rightarrow Class(c)$	(27)	$\forall a \forall o, OA(o,a) \leftrightarrow \exists v, Val(o,a,v)$	(28)
$\forall a \forall c, CA(c,a) \leftrightarrow \exists t, Type(c,a,t)$	(29)	$\forall o \forall c, Instance(o,c) \rightarrow Object(o) \wedge Class(c)$	(30)
$\forall c \forall c', Isa(c,c') \rightarrow Class(c) \wedge Class(c')$	(31)	$\forall c \forall c' \forall c'', Isa(c,c') \wedge Isa(c',c'') \rightarrow Isa(c,c'')$	(32)
$\forall c \forall c' \forall a, Isa(c,c') \wedge CA(c',a) \rightarrow CA(c,a)$	(33)	$\forall c \forall c' \forall m, Isa(c,c') \wedge Method(c',m) \rightarrow Method(c,m)$	(34)
$\forall o \forall a, OA(o,a) \leftrightarrow \exists c, CA(c,a) \wedge Instance(o,c)$	(35)	$\forall c \forall c' \forall o, Instance(o,c) \wedge Isa(c,c') \rightarrow Instance(o,c')$	(36)
$\forall o, Object(o) \rightarrow \exists c, Instance(o,c)$	(37)		
$\forall c \forall o \forall a \forall v \forall t, Val(o,a,v) \wedge Instance(o,c) \wedge Type(c,a,t) \wedge Class(t) \rightarrow Instance(v,t)$			(38)

(25) says that if *a* is an attribute of *c* then *c* is a class.
(26) says that if *a* is an attribute of *o* then *o* is an object.
(27) says that if *m* is a method of *c* then *c* is a class.
(28) says that if *a* is an attribute of object *o*, then there exists a value *v* for this attribute. Conversely it says that if *v* is the value of *a* in (object) *o* then *a* is an attribute of *o*.
(29) says that if *a* is an attribute of class *c*, then there exists a type *t* for this attribute. Conversely it says that if *t* is the type of *a* in *c* then *a* is an attribute of *c*.
(30) says that if *o* is an instance of *c* then *o* is an object and *c* is a class.
(31) says that if *c* is a subclass of *c’* then *c* and *c’* are classes.
(32) says that if *c* is a subclass of *c’* and *c’* is a subclass of *c’’* then *c* is a subclass of *c’’* (transitivity of *Isa*).
(33) says that if *a* is an attribute of *c’* and *c* is a subclass of *c’* then *a* is also an attribute of *c* (inheritance property 1).
(34) says that if *m* is a method of *c’* and *c* is a subclass of *c’* then *m* is also a method of *c* (inheritance property 2).
(35) says that if *a* is an attribute of *o* then there exists a class *c* such as *o* is an instance of *c*. Conversely, if *a* is an attribute of *c* and *o* is an instance of *c* then *a* is an attribute of *o* (inheritance property 3).
(36) says that if *o* is an instance of *c* and *c* is a subclass of *c’* then *o* is an instance of *c’*.
(37) says that if *o* is an object then there exists a class *c* such as *o* is an instance of *c*.
(38) says that if *o* is an instance of *c* and *a* is an attribute of *c* and the type of *a* is class *t* then the value of *a* in *o* must be an instance of *t*.

Frame 1. Integrity constraints of an object-oriented database

2. A set *IC* of basic *integrity constraints* (cf. Frame 1) that are inherent to the object paradigm and that must be consistent with the database content. From a theoretical point of view, this set

of integrity constraints must be derivable from the set of atomic facts that make up the database content ($DB \vdash IC$).

Integrity constraints (25) to (38) are all integrity constraints of *Type* 1 or *Type* 2 (see section 2.2). Note however that integrity constraint (28) stands for two integrity constraints:

$\forall a \forall o, OA(o,a) \rightarrow \exists v, Val(o,a,v)$ which is of *Type* 2

$\forall a \forall o \forall v, Val(o,a,v) \rightarrow OA(o,a)$ which is of *Type* 1

Same particularity appears with integrity constraints (29) and (35).

Notice also that integrity constraint (30) stands for two integrity constraints of *Type* 1:

$\forall o \forall c, Instance(o,c) \rightarrow Object(o)$

$\forall o \forall c, Instance(o,c) \rightarrow Class(c)$

Same particularity happens with integrity constraint (31).

Finally notice that we present the concept of method using a very simple syntactical definition. This definition will allow us to include in our model the possibility to classify the existence of methods. In [CG97], we also include the possibility to protect the program associated with a method. For the sake of simplicity, we prefer to omit this possibility here.

5.3. Example of a non-protected object-oriented database

Instead of giving a long list of atomic facts, we represent our example of a non-protected object-oriented database using a figure (cf. figure 2). One can check that this non-protected object-oriented database respects integrity constraints (25) to (38).

In this example class *Hypersonic_Aircraft* is a subclass of class *Supersonic_Aircraft*. Object *Mirage* is an instance of class *Supersonic_Aircraft* whereas object *FireFox* is an instance of class *Hypersonic _Aircraft* and consequently (cf. axiom (36)) of class *Supersonic_Aircraft*.

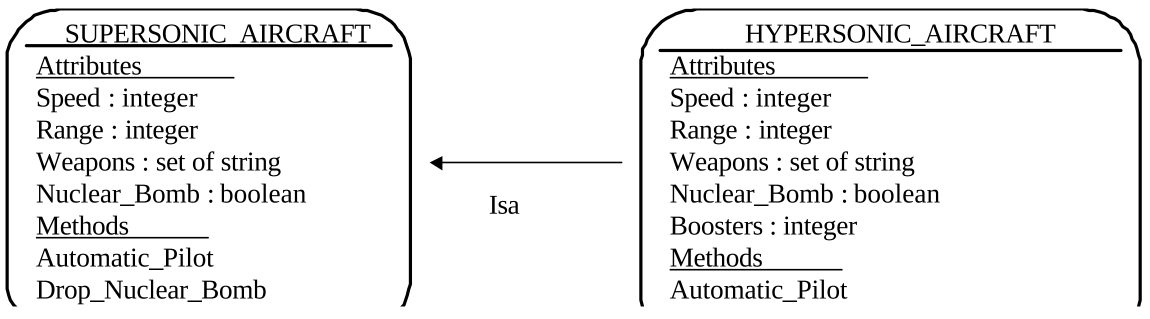

Figure 2: Example of an object-oriented database

6. Multilevel object-oriented database

In this section, we define a model for a multilevel object-oriented database. We previously called this model, the Single-View model [CG96a]. In this model, all the information related to a real-world entity is encapsulated in a single multilevel object.

6.1. Language

As defined in section 3.1 and section 3.4.3, the language L is extended to the language L^{cs} as follows:

- For each predicate P of arity n, there is a predicate P^c of arity (n+1). Therefore, we obtain nine multilevel predicates $Object^c$, $Class^c$, CA^c, OA^c, Isa^c, $Instance^c$, $Method^c$, Val^c and $Type^c$.
- For each Liar predicate P^c of arity n+1, there is a predicate P^{cs} of arity n+1. For the sake of simplicity, we assume in this paper that only Val^c is a Liar predicate (having other Liar predicates would not create any problem). Therefore, Val^{cs} is included in L^{cs}.

6.2. Axiomatic

As defined in section 3.2 and 3.4.3, the axiomatic A^{cs} of the theory T^{cs} includes the following axioms:

1. Axioms from A (cf. section 5.2)
2. Axioms (5) to (12) + *Rule* 1 + *Rule* 2
3. Axiom (16) for all Honest predicates (that is, all of them except Val^c).
4. Axiom (18) to (20) for all Liar predicates (that is, only Val^c).
5. *Rule* 3

6. A set of *atomic facts* that represents the multilevel object-oriented database content.

   Example:

   $Class^{c}(Hypersonic\_Aircraft,C)$
   (that reads “the fact that Hypersonic_Aircraft is a class is confidential”)

   $Object^{c}(FireFox,U)$
   (that reads “the fact that FireFox is an object is unclassified”)

   $Instance^{c}(FireFox, Hypersonic\_Aircraft,S)$
   (that reads “the fact that FireFox is an instance of Hypersonic_Aircraft is secret”)

   $Val^{c}(FireFox,Speed,mach\ 6,S)$
   (that reads “the fact that mach 6 is the value of the Speed of FireFox is secret”)

   $Val^{c}(FireFox,Speed,mach\ 2,U)$
   (that reads “the fact that mach 2 is the value of the Speed of FireFox is unclassified”)

7. A set of *atomic facts* that represents the cover stories.

   Example:

   $Val^{cs}(FireFox,Speed,mach\ 2,S)$
   (that reads “the fact that $Val(FireFox,Speed,mach\ 2)$ is a cover story is secret”)

6.3. Inference channels control theorems

Our objective in this section is to derive general constraints that must be enforced when classifying the object-oriented database content. As explained in section 3.3, we must apply deductive channels control *Rule 1* or signalling channels control *Rule 2* to each of the integrity constraints from (25) through (38). If the integrity constraint is of *Type 1* then we must apply *Rule* 1, if it is of *Type* 2 then we must apply *Rule 2.* We then obtain a list of deductive channels control theorems and a list of signalling channels control theorems. Notice that Olivier and Solms [OS94] have already proposed a less complete list of such theorems.

6.3.1. Deductive channels control theorems

Deductive channels control theorems (cf. Frame 2) are derived by applying *Rule* 1 on integrity constraints (25) to (38) that are of *Type* 1. Recall that each of the integrity constraints (28), (29) and (35) actually stands for two integrity constraints, one of *Type* 1 and one of *Type* 2. Recall also that each of the integrity constraints (30) and (31) stands for two integrity constraints of *Type* 1 (see section 6.2).

Some of these theorems contradict some rules enounced in other security models. For example, theorem (44) says that if the fact “*o* is an instance of class *c*” is classified at level *l* then there must exist a level *l’* and a level *l’’*, both dominated by *l*, which respectively protect the facts “*o* is an object” and “*c* is a class”. Some authors [Lun90,JK90,ML92] pretend that the security level protecting an objet *o* must dominate the security level protecting the class *c* which *o* is an instance of. This theorem (44), formally derived and proved, shows that this does have to be the case (see section 6.4 for a practical counter-example).

Notice that theorem (51) precisely enounces the constraint that must be enforced when assigning a security level to an attribute value that is itself an object.

$\forall a \forall c \forall l, CA^{c}(c,a,l) \rightarrow \exists l', Class^{c}(c,l') \wedge l' \leq l$	(39)	$\forall a \forall o \forall l, OA^{c}(o,a,l) \wedge \rightarrow \exists l', Object^{c}(o,l') \wedge l' \leq l$	(40)
$\forall m \forall c \forall l, Method^{c}(c,m,l) \rightarrow \exists l', Class^{c}(c,l') \wedge l' \leq l$	(41)	$\forall a \forall o \forall v \forall l, Val^{c}(o,a,v,l) \rightarrow \exists l', OA^{c}(o,a,l') \wedge l' \leq l$	(42)
$\forall a \forall c \forall t \forall l, Type^{c}(c,a,t,l) \rightarrow \exists l', CA^{c}(c,a,l') \wedge l' \leq l$	(43)		
$\forall o \forall c \forall l, Instance^{c}(o,c,l) \rightarrow \exists l' \exists l'', Object^{c}(o,l') \wedge Class^{c}(c,l'') \wedge \text{lub}(l',l'') \leq l$			(44)
$\forall c \forall c' \forall l, Isa^{c}(c,c',l) \rightarrow \exists l' \exists l'', Class^{c}(c,l') \wedge Class^{c}(c',l'') \wedge \text{lub}(l',l'') \leq l$			(45)
$\forall c \forall c' \forall c'' \forall l \forall l', Isa^{c}(c,c',l) \wedge Isa^{c}(c',c'',l') \rightarrow \exists l'', Isa^{c}(c,c'',l'') \wedge l'' \leq \text{lub}(l,l')$			(46)
$\forall c \forall c' \forall a \forall l \forall l', Isa^{c}(c,c',l) \wedge CA^{c}(c',a,l') \rightarrow \exists l'', CA^{c}(c,a,l'') \wedge l'' \leq \text{lub}(l,l')$			(47)
$\forall c \forall c' \forall m \forall l \forall l', Isa^{c}(c,c',l) \wedge Method^{c}(c',m,l') \rightarrow \exists l'', Method^{c}(c,m,l'') \wedge l'' \leq \text{lub}(l,l')$			(48)
$\forall o \forall a \forall c \forall l \forall l', CA^{c}(c,a,l) \wedge Instance^{c}(o,c,l') \rightarrow \exists l'', OA^{c}(o,a,l'') \wedge l'' \leq \text{lub}(l,l')$			(49)
$\forall c \forall c' \forall o \forall l \forall l', Isa^{c}(c,c',l) \wedge Instance^{c}(o,c,l') \rightarrow \exists l'', Instance^{c}(o,c',l'') \wedge l'' \leq \text{lub}(l,l')$			(50)
$\forall c \forall o \forall a \forall v \forall t \forall l_1 \forall l_2 \forall l_3 \forall l_4, Val^{c}(o,a,v,l_1) \wedge Instance^{c}(o,c,l_2) \wedge Type^{c}(c,a,t,l_3) \wedge Class^{c}(t,l_4)$ $\rightarrow \exists l_5, Instance^{c}(v,t,l_5) \wedge l_5 \leq \text{lub}(l_1,l_2,l_3,l_4)$			(51)

(39) says that if the fact "*a* is an attribute of class *c*" is classified at level *l* then there must exist a level *l'* dominated by *l* that protects the fact "*c* is a class". (Combine *Rule* 1 with axiom (25))

(40) says that if the fact "*a* is an attribute of object *o*" is classified at level *l* then there must exist a level *l'* dominated by *l* that protects the fact "*o* is an object". (Combine *Rule* 1 with axiom (26))

(41) says that if the fact "*m* is a method of class *c*" is classified at level *l* then there must exist a level *l'* dominated by *l* that protects the fact "*c* is a class". (Combine *Rule* 1 with axiom (27))

(42) says that if the fact "*v* is the value of the attribute *a* in object *o*" is classified at level *l* then there must exist a level *l'* dominated by *l* that protects the fact "*a* is an attribute of object *o*". (Combine[9] *Rule* 1 with axiom(28))

(43) says that if the fact "*t* is the type of the attribute *a* in class *c*" is classified at level *l* then there must exist a level *l'* dominated by *l* that protects the fact "*a* is an attribute of class *c*". (Combine *Rule* 1 with axiom (29))

(44) says that if the fact "*o* is an instance of class *c*" is classified at level *l* then there must exist a level *l'* and a level *l''* both dominated by *l* and respectively protecting the facts "*o* is an object" and "*c* is a class". (Combine[10] *Rule* 1 with axiom (30)).

(45) says that if the fact "*c* is a subclass of class *c'* " is classified at level *l* then there must exist a level *l'* and a level *l''* both dominated by *l* and respectively protecting the facts "*c* is a class" and "*c'* is a class". (Combine *Rule* 1 with axiom (31))

(46) says that if the fact "*c* is a subclass of class *c'* " is classified at level *l* and if the fact "*c'* is a subclass of *c''* " is classified at level *l'* then there must exist a level *l''* dominated by the least upper bound of *l* and *l'* that protects the fact "*c* is a subclass of class *c''* " (Combine *Rule* 1 with axiom (32))

(47) says that if the fact "*c* is a subclass of class *c'* " is classified at level *l* and if the fact "*a* is an attribute of *c'* " is classified at level *l'* then there must exist a level *l''* dominated by the least upper bound of *l* and *l'* that protects the fact "*a* is an attribute of class *c*" (Combine *Rule* 1 with axiom (33))

(48) says that if the fact "*c* is a subclass of class *c'* " is classified at level *l* and if the fact "*m* is a method of *c'* " is classified at level *l'* then there must exist a level *l''* dominated by the least upper bound of *l* and *l'* that protects the fact "*m* is a method of class *c*" (Combine *Rule* 1 with axiom (34))

(49) says that if the fact "*a* is an attribute of *c*" is classified at level *l* and if the fact "*o* is an instance of class *c*" is classified at level *l'* then there must exist a level *l''* dominated by the least upper bound of *l* and *l'* that protects the fact "*a* is an attribute of object *o*" (Combine *Rule* 1 with axiom (35))

(50) says that if the fact "*c* is a subclass of *c'* " is classified at level *l* and if the fact "*o* is an instance of *c*" is classified at level *l'* then there must exist a level *l''* dominated by the least upper bound of *l* and *l'* that protects the fact "*o* is an instance of class *c'* " (Combine *Rule* 1 with axiom (36))

(51) says that if the fact "*t* is a class" is classified at level l_4, if the fact "*t* is the type of attribute *a* in class *c*" is classified at level l_3, if the fact "*o* is an instance of class *c*" is classified at level l_2 and if the fact "*v* is the value of attribute *a* in object *o*" is classified at level l_1 then there must exist a level l_5 dominated by the least upper bound of l_1, l_2, l_3 and l_4 that protects the fact "*v* is an instance of *t*" (Combine *Rule* 1 with axiom (38))

Frame 2. Deductive channels control theorems for a multilevel object-oriented database

[9] As already mentioned, (28) stands for two integrity constraints, one of type 1 and one of type 2 (cf section 5.2). (42) is actually obtained by applying *Rule* 1 on the integrity constraint of type 1. Similar comment can be made for (43) and (49).

[10] As already mentioned, (30) stands for two integrity constraints of type 1 (cf section 5.2). (44) is actually obtained by successively applying *Rule* 1 on these two integrity constraints and then by combining the two resulting theorems. Similar comment can be made for (45).

6.3.2. Signalling channels control theorems

Signalling channels control theorems (cf. Frame 3) are derived by applying *Rule* 2 on integrity constraints from (25) through (38) which are of *Type* 2. Recall that each of the integrity constraints (28), (29) and (35) actually stands for two integrity constraints, one of *Type* 1 and one of *Type* 2 (see section 6.2).

$\forall a \forall o \forall l, OA^c(o,a,l) \rightarrow \exists v \exists l', Val^c(o,a,v,l') \wedge (l' \leq l)$ (52)	$\forall a \forall o \forall l, OA^c(o,a,l) \rightarrow \exists v, Val^c(o,a,v,l)$ (53)
$\forall a \forall c \forall l, CA^c(c,a,l) \rightarrow \exists t, Type^c(c,a,t,l)$ (54)	$\forall o \forall l, Object^c(o,l) \rightarrow \exists c, Instance^c(o,c,l)$ (55)
$\forall o \forall a \forall l, OA^c(o,a,l) \rightarrow \exists c \exists l' \exists l'', CA^c(c,a,l') \wedge Instance^c(o,c,l'') \wedge l = \text{lub}(l', l'')$ (56)	

(52) says that if the classification of the fact "*a* is attribute of object *o*" is classified at level *l* then there must exist a value *v* for this object attribute whose classification level *l'* is dominated by *l*. (Combine *Rule* 2 with axiom (28))

(53) says that if the classification of the fact "*a* is attribute of object *o*" is classified at level *l* then there must exist a value *v* for this object attribute whose classification level is *l*. This theorem is obtained by combining the theorem (52) with theorem (42). The proof is similar to the proof made for (15)

(54) says that if the classification of the fact "*a* is attribute of class *c*" is classified at level *l* then there must exist a type *t* for this class attribute whose classification level is *l*. Applying Rule 2 onto axiom (29) and then combining the result with theorem (43) gives this theorem. The proof is similar to the proof made for (15).

(55) says that if the classification of the fact "*o* is an object" is classified at level *l* then there must exist a class *c* which *o* is an instance of and such as the instance link between *o* and *c* is equal to *l*. Applying Rule 2 onto axiom (37) and then combining the result with theorem (44) gives this theorem. The proof is similar to the proof made for (15).

(56) says that if the classification of the fact "*a* is attribute of object *o*" is classified at level *l* then there must exist a class *c* such that "*o* is an instance of *c*" and "*a* is attribute of *c*" are two existing facts respectively classified at level *l'* and *l''* such as *l*=lub(*l'*,*l''*). Applying Rule 2 onto axiom (35) and then combining the result with theorem (49) gives this theorem. The proof is similar to the proof made for (15).

Frame 3. Signalling channels control theorems for a multilevel object-oriented database

6.4. Example of a multilevel object-oriented database

Figure 3 presents a multilevel database derived from the non-protected object-oriented database represented in figure 2. This multilevel database is consistent with theorems (39)-(56). We appeal the reader's attention to the following points:

- There are three security levels U = Low (Unclassified), C (Confidential) and S = High (Secret) with U < C < S.
- Unclassified information is in regular font, confidential information is in *italic* and secret information is in **bold**.
- The existence of class *Hypersonic_Aircraft* is confidential.
- The existence of attribute *Nuclear_Bomb* is secret in every class.
- The existence of both objects *Mirage* and *FireFox* is unclassified.

- The fact that *FireFox* is an instance of class *Hypersonic_Aircraft* is secret, but the fact it is an instance of Supersonic_Aircraft is unclassified. This is an illustration of theorem (44) and a counter-example to a rule enounced by [Lun90,JK90,ML92]. This rule states that an object must always be classified at a level dominating the security level assigned to the class, which this object is an instance of.

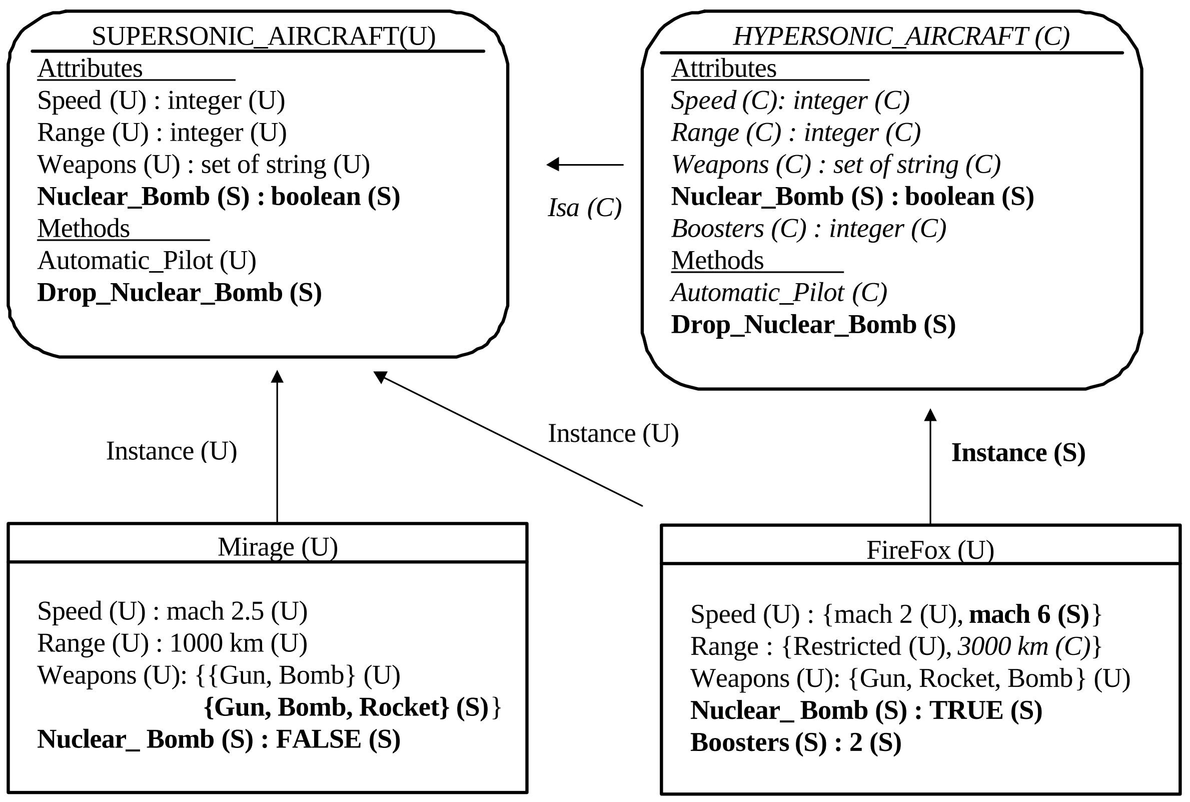

Figure 3: Example of a multilevel object-oriented database

- The secret value *mach* 6 is the actual *Speed* value of *Firefox*, whereas the unclassified value *mach* 2 stands for a cover story. This means that $Val^{cs}(FireFox,Speed,mach\ 2,S)$ belongs to the multilevel database.
- Attribute *Weapons* of *Mirage* is associated with a set of values: *Gun* and *Bomb* that are both unclassified and *Rocket* that is secret. This means that $Val^{c}(Mirage,Weapon,Gun,U)$, $Val^{c}(Mirage,Weapon,Bomb,U)$ and $Val^{c}(Mirage,Weapon,Rocket,S)$ belong to the multilevel database.

- Attribute *Range* of *FireFox* is associated with the *Restricted* special value at the unclassified level. This means that unclassified users are permitted to know that a higher classified value exists for the *Range* attribute. Recall that from a theoretical point of view (cf. section 3.4.4) it does not mean that $Val^{c}(FireFox,Range,Restricted,U)$ belongs to the database. This *Restricted* value in figure 3 stands for the following unclassified sentence:

  $$\exists v \exists l, Val^{c}(Firefox, Range, v, l) \wedge v \neq Restricted \wedge U < l.$$

  As a consequence, theorem (53) is violated for this particular attribute value.

- Solution 2 of section 3.4.4 is applied in order to avoid the insertion of cover stories concerning the type of the attributes. In other words, we assume the following sentence to be an axiom:

  $$\forall a \forall c \forall t \forall l \forall l', CA^c(c,a,l) \wedge Type^c(c,a,t,l') \rightarrow l = l'$$

  It also means that, in our particular example, we estimate that it is not necessary to classify a class attribute and the type of this attribute independently from each other. However, in some applications, being able to classify the type of an attribute and the attribute itself separately could be a useful possibility.

We also provide the following comments on figure 3:

- Every attribute and method of the confidential class *Hypersonic_Aircraft* is classified at the confidential level or higher (theorems (39) and (41)).
- Inheritance link between class *Hypersonic_Aircraft* and *Supersonic_Aircraft* must be at least confidential (theorem (45))
- Since the inheritance link between the classes *Hypersonic_Aircraft* and *Supersonic_Aircraft* is confidential, the attributes *Speed, Range* and *Weapons* of class *Hypersonic_Aircraft* are confidential (they cannot be secret) (theorem (47)).
- The existence of attribute *Boosters* is confidential in class *Hypersonic_Aircraft* but is secret in object *FireFox*. This is due to theorem (56).
- For every attribute value different from *Restricted*, theorem (53) is satisfied.
- The instance links between *Mirage* and *Supersonic_Aircraft* and between *FireFox* and *Supersonic_Aircraft* are unclassified. This is due to theorem (55).

7. MultiView object-oriented database

The Single-View model presented in the previous section is a complete model. However, as suggested in section 4, we think that it is necessary to precisely define what part of the multilevel database each user may observe. In this section, we shall apply the principle described in section 4 that is, we shall decompose a *n*-level object oriented database into *n* views. Recall that each view is associated with its own security level and contains a complete and consistent set of data whose classifications are lower than or equal to the level of the view. In section 8, we shall investigate how to implement a MultiView object-oriented database. The technique we suggest will allow us to avoid unnecessary replications.

7.1. Language

As defined in section 4.1, the language L^{cs} is extended into the language L^v as follows: for each predicate P^c of arity n+1, there is a predicate P^v of arity (n+1). Therefore, we obtain nine MultiView predicates $Object^v$, $Class^v$, CA^v, OA^v, Isa^v, $Instance^v$, $Method^v$, Val^v and $Type^v$.

7.2. Axiomatic

According to section 4.2, the axiomatic A^v of the theory T^v includes the following axioms:

1. Axioms from A^{cs} (cf. section 6.2).
2. Axiom (21) for all Honest predicates
3. Axioms (22) and (23) for all Liar predicates (that is, only Val^c)
4. Axiom (24)
5. A set of atomic facts that represents the MultiView object-oriented database

   Example:

   $Class^v(Supersonic\_Aircraft,U)$
   (that reads "the fact that Supersonic_Aircraft is a class belongs to the unclassified view")

   $Class^v(Supersonic\_Aircraft,C)$
   (that reads "the fact that Supersonic_Aircraft is a class belongs to the confidential view")

   $Class^v(Supersonic\_Aircraft,S)$
   (that reads "the fact that Supersonic_Aircraft is a class belongs to the secret view")

   $Val^v(FireFox,Speed,mach\ 6,S)$
   (that reads "the fact that mach 6 is the value of the Speed of FireFox belongs to the secret view")

7.3. Example of a MultiView object-oriented database

Figure 4 presents a MultiView database derived from the multilevel object-oriented database represented in figure 3. For a better reading of the figure, inherited attributes and methods of each class view are not shown. We appeal the reader's attention to the following points:

- Every information classified at level *l* that is not a cover story or the special *Restricted* value is replicated at all levels dominating the level *l* (axiom (21)).
- Attribute values are derived from axiom (22) (for example value *mach* 2 in the unclassified view or value *mach* 6 in the secret view) or from axiom (23) (for example value *mach* 2 in the confidential view).

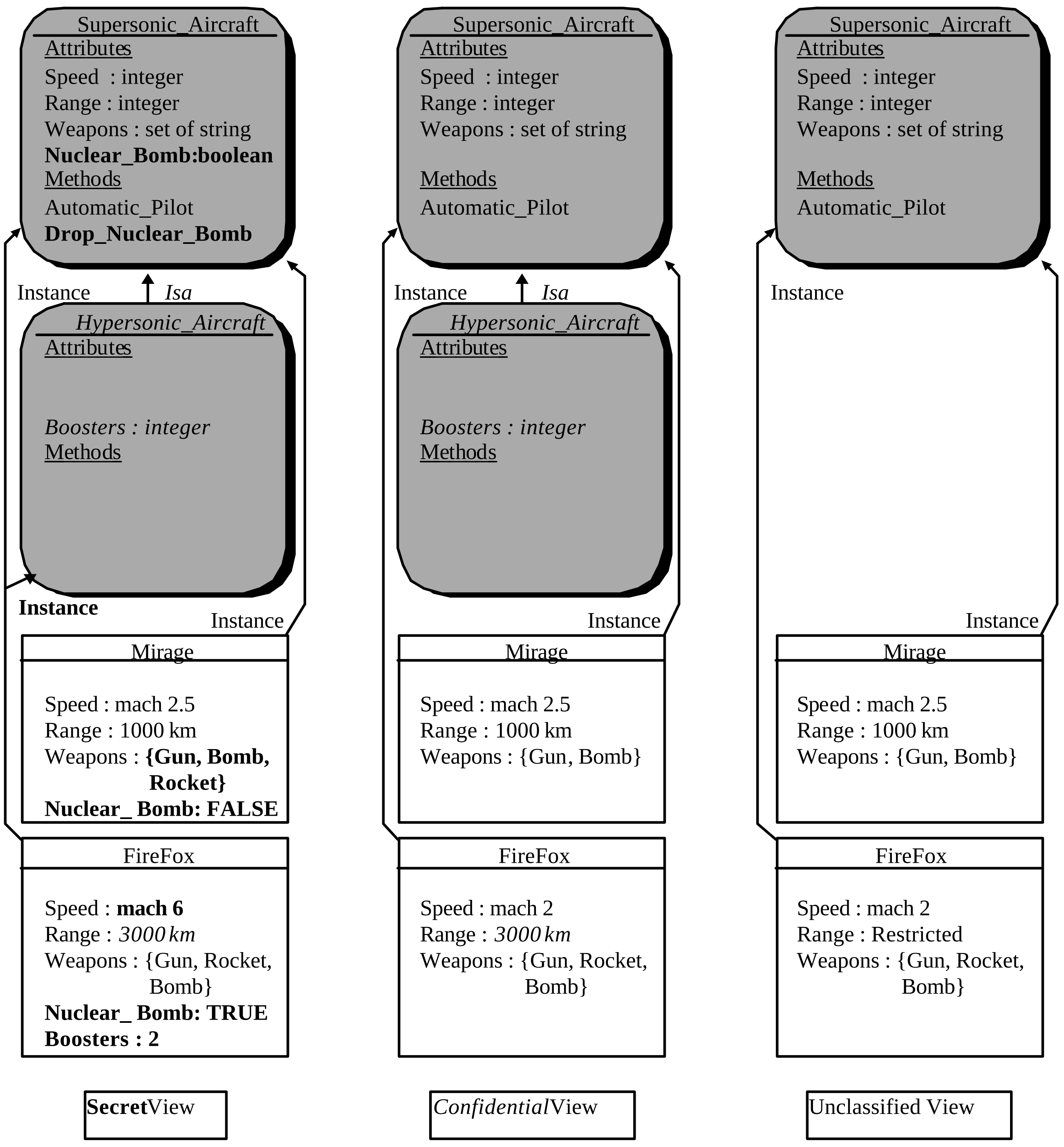

Figure 4: Example of a MultiView object-oriented database

8. Implementing a MultiView object-oriented database

We can choose to implement either the layer 2 or the layer 3 of our 3-layer model. Indeed, in section 4, we showed that there exists a one-to-one correspondence between these two layers.

If we choose to implement the layer 2 then we must build a mechanism, which provides a user with a consistent view of the multilevel database. This view must be complete and compatible with the user's clearance level. In [BCGY94b], we propose a model (the *Virtual View* model) implementing such a mechanism. Its principle is the following: a user of level *l* is provided with a *virtual view* of level *l* derived from the multilevel database. This *virtual view* is consistent, complete and compatible with the user's clearance level. The main disadvantage of this approach is that the mechanism that builds the virtual view must be trusted.

We claim that implementing the layer 3 is easier. Indeed, in the MultiView model, a user of level *l* may directly observe all the views of level dominated by *l* (see [CG96b] for more details). Therefore, access controls are easy to define and there is no need of a special intermediate trusted mechanism between the user and the database. Thus, in this section, we propose some basic principles to implement the layer 3 for the particular case of an object-oriented database.

8.1. Principles

In this section we propose a short sketch of implementation for a MultiView object-oriented database. More precisely, we suggest some principles for implementing a MultiView object-oriented database. However, we do not claim that these principles are complete. In particular, we do not investigate the problem of updating a MultiView database. The reader can refer to [CG96a] or [Gab95] for more explanations.

The principles we suggest for implementing a MultiView object-oriented database are the following:

- Each view of level *l* is implemented as a single-level database of level *l*.
- In the MultiView model, a multilevel entity (object or class) is represented by several views of this entity. Since a major principle of the object-oriented paradigm is the unicity of the object identifier, each object or class must be uniquely identified. We propose to implement this principle as follows: each view at level *l* of a multilevel object *o* (respectively a multilevel class *c*) is uniquely identified by the pair (*o*,*l*) (respectively (*c*,*l*)).
- All single-level class and object views sharing the same security level are stored into a single-level object-oriented database. Each single-level database can be managed as a non-protected object-oriented database. However, in order to avoid unnecessary replications, we propose implementing some dynamic links between the different single-level databases.
- Axiom (23) says that an attribute value $Val^v(o,a,v,l)$ must be propagated in all[11] the levels *l'* that are just above the level *l*, where $Val(o,a,v)$ is not referred as a cover story. We propose to implement this axiom as follows: For all these level *l'*, $Val^v(o,a,v,l')$ is evaluated using a pointer to the view at level *l*. This pointer value is a syntactical expression which enables the system to automatically retrieve the value *v* from the database of level *l* (see [BCGY94a] for more details).
- By induction on axioms (21) and (24), we can easily prove that for all Honest predicates P^c the following theorem can be derived:

[11] Recall that we may have a partial order defined on the set of security levels

$$\forall t_1 \ldots \forall t_n \forall l \forall l', P^v(t_1,\ldots,t_n,l) \wedge l \le l' \to P^v(t_1,\ldots,t_n,l')$$

It means in particular that every class attribute and every method belonging to a low level database must be replicated in all higher classified databases. In order to implement a dynamic replication mechanism, we introduce an inheritance link between any class view (*c*,*l*) and class view (*c*,*l'*) with *l* being just below *l'*. Low-level attributes and low-level methods are then automatically replicated to the higher level databases by inheritance.

8.2. Example of an implemented MultiView object-oriented database

Figure 5 corresponds to the example of a MultiView object-oriented database presented in figure 4. For a better reading of the figure, inherited attributes and methods of each class view are not shown. We appeal the reader's attention to the following points:

- The views represented in figure 4 are now implemented as single-level databases.
- Any view at level *l* of a multilevel object *o* (respectively class *c*) is now uniquely identified by a pair (*o*,*l*) (respectively (*c*,*l*)).
- Attribute values are automatically replicated by using pointers. For example, the value of the *Speed* attribute of (*FireFox,C*) is evaluated using a pointer to the value *mach 2* of the *Speed* attribute of (*FireFox,U*). An arrow represents this pointer. This pointer is actually a syntactical expression of the form "(*FireFox,U*).*Speed*". More can be found about these pointers in [BCGY94a]. Notice that thanks to these pointers, an update to an attribute value at a low level would be automatically propagated to the higher levels.
- Attributes and methods are automatically replicated via inheritance links *Isa1*, *Isa2*, *Isa3*. Notice that thanks to these inheritance links, an insertion of a new class attribute (or method) at a low level would be automatically propagated to the higher levels.

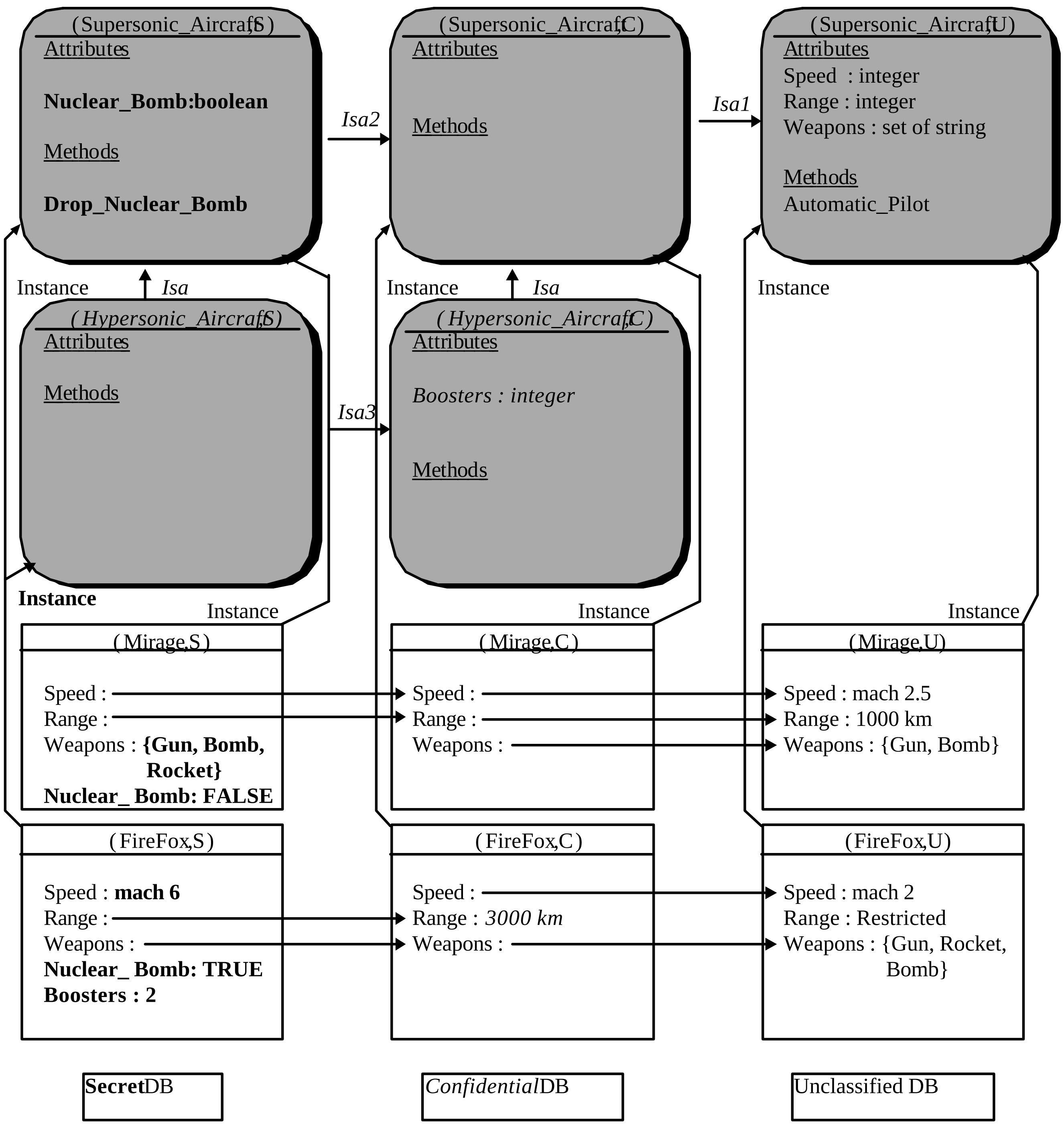

Figure 5: Example of implementation of a MultiView object-oriented database

9. Conclusion

In this paper, our objective was to propose a complete and formal model for any type of multilevel database. Definition of this model was done in three phases:

- Non-protected database model: We defined a language based on first order logic. This language allowed us to represent a database content including integrity constraints.
- Multilevel database model: We extended our language in order to protect each piece of information represented by an atomic formula. We also derived some general theorems that must be enforced when classifying the database content. Finally we provided a means to introduce cover stories in the multilevel database.
- MultiView database model: We extended again our language in order to decompose the multilevel database into a collection of single-level Views.

Many multilevel security models for database (relational or object-oriented) have already been presented by different authors [DLSSH88,Wil89,HOST90,SW92,CS95,QL96,JK90,Lun90, ML92,KTT89]. However, each of these models is specific to a particular type of database, relational [DLSSH88,Wil89,HOST90,SW92,CS95,QL96] or object-oriented [JK90,Lun90,ML92,KTT89]. None of them proposes a methodology to formalize a multilevel database in general. Moreover the use of polyinstantiation generates several ambiguities in most of these models.

Our model is complete, consistent and free of ambiguities:

- It is complete in the sense that given a set of integrity constraints of *Type 1* or *2*, we derive from *Rules 1* and *2* all and every possible inference control theorems.
- It is consistent since it is possible to build a model of our logical theory. For example, Figure 2 is a model of the logical theory representing a non protected object-oriented database, Figure 3 is a model of the logical theory representing a multilevel object-oriented database and Figure 4 is a model of the logical theory representing a MultiView object-oriented database.
- It is free of ambiguities since each concept used in this paper is formally defined using mathematical logic. In particular, giving a formal definition for the concept of cover stories certainly contributed to the clarification of this notion.

In the last sections of this paper, we applied our general 3-layer approach to represent a MultiView object-oriented database. This particular MultiView model for an object-oriented database has already been presented in several papers [BCGY93a,BCGY93b,BCGY94a,CG96a] and some of its results have been reused in other security models [SMKL95]. Compared to the previous versions of this model, the new version presented in this paper shows two main improvements:

- It includes explicit management of cover stories.
- It includes the possibility to have a partial order in the set of security levels.

The main use of our model is to *build* a multilevel database. However, we can also apply this model to *evaluate* an existing multilevel database. For instance, we can check whether an existing multilevel database respects the inference control rules *Rule 1* and *Rule 2*. A useful extension to this model could be then to define some guidelines that could be used to conform the existing database to these inference control rules.

Our work is purely theoretical. In this paper, we did not address the problem of implementation of our model. However, we gave some basic implementation principles in the last section of this paper for the particular case of a MultiView object-oriented database. More can be found about this topic in [BCGY93a,Gab95]. Another issue we did not consider here, is the problem of updating a MultiView database. This issue remains to be investigated although some results are presented in [Gab95].

References:

[BCGY93a] N. and F. Cuppens, A. Gabillon, K. Yazdanian. *"MultiView Model for Multilevel Object-Oriented DataBases"*. Proc. of the Ninth Annual Computer Security Applications Conference (ACSAC). Orlando, USA 1993.

[BCGY93b] N. and F. Cuppens, A. Gabillon, K. Yazdanian. *"Techniques to Handle MultiLevel Objects in Secure Object-Oriented DataBases"*. Proc. of the OOPSLA-93 Workshop on "Security for Object-Oriented Systems". Washington, USA 1993.

[BCGY94a] N. Boulahia-Cuppens, F. Cuppens, A. Gabillon and K. Yazdanian. *Decomposition of Multilevel Objects in an Object-Oriented Database*. In European Symposium on Research in Computer Security. UK 1994. Springer Verlag.

[BCGY94b] N. and F. Cuppens, A. Gabillon, K. Yazdanian. *"Virtual View Model to Design A Secure Object-Oriented DataBase"*. Proc. of the 17th National Computer Security Conference (NCSC). Baltimore, USA 1994.

[CG96a] F. Cuppens and A. Gabillon. *A Logical Approach to Model a Multilevel Object-Oriented Database*. Proc. of the 10th Annual IFIP WG 11.3 Working Conference on Database Security. Como Italy.1996.

[CG96b] F. Cuppens, A. Gabillon. *"A Query Language For a Multilevel Object-Oriented Database"*. Proc. of the XI International Symposium on Computer and Information Sciences (ISCIS). Antalya, Turkey. November 1996.

[CG97] F. Cuppens and A. Gabillon. *Design of multilevel object-oriented databases*. Proc. of the twelfth International Symposium on Computer and Information Science (ISCIS). Turkey.

[CS95] F. Chen and R. S. Sandhu. *The Semantics and Expressive Power of the MLR Data Model*. IEEE Symposium on Security and Privacy. Oakland 1995.

[Cup96] F. Cuppens. *Querying a Multilevel Database: A Logical Analysis*. Proceedings of the 22nd International Conference on Very Large Databases. Bombay, India 1996.

[DLSSH88] D. Denning, T. Lunt, R Schell, W Shockley and M. Heckman. *The Sea View Security Model*. Proc. of the 1988 IEEE Symposium on Research in Security and Privacy.

[Gab95] A. Gabillon. *Sécurite multi-niveaux dans les bases de données a objets*. Ph.D. dissertation. ENSAE 1995.

[HOST90] J. Haig, R, O'Brien, P Stachour and D. Toups. *The LDV approach to database security*. In Database Security III, Status and Prospect. North Holland.

[JK90] S. Jajodia and B. Kogan. *Integrating an Object-Oriented Data Model with Multilevel Security*. In IEEE Symposium on Security and Privacy. Oakland 1990.

[KTT89] T. Keefe, W. Tsai and B Thuraisingham. *SODA: A Secure Object-Oriented Database System*. Computer and Security, 8(6), 1989.

[Lun90] T.F. Lunt. *Multilevel Security for Object-Oriented Database Systems*. In D.L. Spooner and C. Landwehr editors. Database Security III: Status and Prospects. North-Holland 1990. Result of the IFIP WG 11.3 Workshop on Database Security.

[ML92] J.K. Millen and T.F. Lunt. *Security for Object-Oriented Database Systems*. Proc. of the 1992 IEEE Symposium on Research in Security and Privacy.

[QL96] X. Qian and T. F. Lunt. *A Mac Policy Framework for Multilevel Databases*. IEEE Transactions on Knowledge and Data Engineering. Vol 8, Number 1, February 1996.

[OS94] M. S. Olivier and S. H. Von Solms. *A Taxonomy for Secure Object-Oriented Databases*. ACM Transactions on Database Systems. Vol 19, Number 1. March 1994.

[Rei78] R. Reiter. *On Closed World Databases*. Logic and Database. H.Gallaire and J. Mincker. New-York: Plenum. 1978.

[Rei83] R. Reiter. *Toward a logical reconstruction of relational database theory*. In On Conceptual Modelling: Perspectives from Artificial Intelligence, Databases and Programming Languages. Springer Verlag, 1983.

[SJ92] R. Sandhu and S. Jajodia. *Polyinstantiation for cover stories*. In European Symposium on Research in Computer Security. Toulouse, France.1992. Springer Verlag.

[SJ93] R. S. Sandhu and S. Jajodia. *Referential Integrity in Multilevel Secure Databases.* Proceedings of the 16th National Computer Security Conference. 1993.

[SMLK95] M. Schaefer, P Martel, T. Kanawan and V. Lyons. *Multilevel data model for the trusted ONTOS prototype*. In Ninth Annual IFIP WG 11.3 Working Conference on Database Security, Rensselaerville, USA, 1995.

[SW92] K. Smith and M. Winslett. *Entity Modeling in the MLS Relational Model*. Proceedings of the 18th International Conference on Very Large Data Bases. Vancouver, Canada 1992.

[Wie91] R. Wieringa. *A formalisation of objects using equational dynamic logic.* Second international DOOD'91 conference. Germany 1991.

[Wil89] J. Wilson. *A Security Policy for an A1 DBMS (A Trusted Subject}*. IEEE Symposium on Security and Privacy. Oakland 1989.