On Lattices in Access Control Models

Sergei Obiedkov, Derrick G. Kourie, and J.H.P. Eloff

Department of Computer Science, University of Pretoria, Pretoria 0002, South Africa

Abstract. Lattices have been extensively used for implementing mandatory access control policies. Typically, only a small sublattice of the subset lattice of a certain alphabet is used in applications. We argue that attribute exploration from formal concept analysis is an appropriate tool for generating this sublattice in a semiautomatic fashion. We discuss how two access control models addressing different (in a sense, opposite) requirements can be incorporated within one model. In this regard, we propose two operations that combine contexts of the form (G, M, I) and (N, G, J). The resulting concept lattices provide most of the required structure.

1 Introduction

Multiuser computer systems must ensure that information they contain is accessible only to those who are authorized to use it. Therefore, choosing and implementing an access control model suitable for a particular computer system is an important part of its development.

Some of the most well-know access control models make heavy use of lattices [1]. They are based on Denning's Axioms formulating assumptions under which an information flow policy can be regarded as a lattice [2]. Due to relative complexity of their creation and maintenance, these models have enjoyed only limited use in practice.

In the access control setting, one speaks about active subjects (such as users or processes) accessing passive objects (such as files or resources). This separation is not absolute: an object such as a program can become a subject when trying to access another object such as a file.

In lattice-based control models, security labels are assigned to entities (subjects and objects). These security labels are partially ordered and, in fact, form a lattice. There are good reasons for it to be a lattice rather than just a partial order: the supremum and infimum operations play a role in, for instance, some versions of the Biba model [3, 4]. Information is allowed to flow only in one direction, e.g., from entities with security labels that are lower in the lattice to entities with security labels that are higher. Section 2 describes the lattice-related aspects of the Bell-LaPadula model [5].

A security label is usually a combination of a security level (*secret*, *confidential*, etc.) and a subset of categories (project names, academic departments, etc.). In practice, only a small fraction of all possible labels is used. In Section 3, we argue that attribute exploration from formal concept analysis (FCA) can help effectively identify such useful labels. We describe attribute exploration rather informally (since formal definition is available elsewhere [8]) and, perhaps, more than necessary for the general ICCS audience, but this is to give security experts without FCA background an idea of how they can benefit from using this technique.

A lattice-based access control model typically addresses some particular security concerns, such as confidentiality (the Bell-LaPadula model [5]) or integrity (the Biba model [3]). In terms of formal concept analysis, the two models can be expressed by two formal contexts such that the object set of one context is the attribute set of the other: (G, M, I) and (N, G, J). Some researchers combine the lattices of the two models into one lattice [6] to obtain a unified information flow policy. In Section 4, we discuss the resulting structure, which arises from a combination of the contexts mentioned above. Then, we outline an attribute exploration procedure to get the necessary components of this structure.

2 Lattices in Access Control Models

Definition 1. *Let H denote a totally ordered set of* classifications *or* security levels *(we use standard notation for this order: e.g., $<$ and $\leq$) and let C denote a set of* categories *such as names of departments within a company, project names, etc. Then, a* compartment *is defined as a subset of categories, and a* security label *is a pair (h, D) consisting of a security level $h \in H$ and a compartment $D \subseteq C$.*

Security labels are partially ordered: $(h, D) \leq (g, E)$ if and only if $h \leq g$ and $D \subseteq E$. It is easy to see that this partial order is a lattice: it is the product of two lattices, $(H, \leq)$ and $(\mathfrak{P}(C), \subseteq)$. Such lattices are common in military security models. Sometimes, only a subset of the product is used in practice.

Example 1. Suppose that there are three security levels: *unclassified*, *secret*, *top secret*. Let $C = \{a, b, c\}$ contain the names of three different projects in the system. Figure 1 presents a security lattice built from H and C. Subjects with the label $(secret, \{a\})$ can access objects with the same label and objects labeled $(unclassified, \emptyset)$, but cannot access objects labeled, e.g., $(top\ secret, \{a, b\})$ or $(secret, \{b\})$.

It should be noted that by accessing we mean essentially reading (rather than writing or modifying). Thus, an access control model, such as the one from Example 1, addresses the *confidentiality* requirement [4]:

Confidentiality: prevention of unauthorized disclosure of information.

One of the most popular models dealing with confidentiality issues is the Bell-LaPadula model [5]. There exist many variants of this model; we concentrate only on its lattice-related aspects and follow the presentation in [1].

Assuming that $\lambda(e)$ is the security label of the subject or object e, the *simple-security property* is formulated as follows:

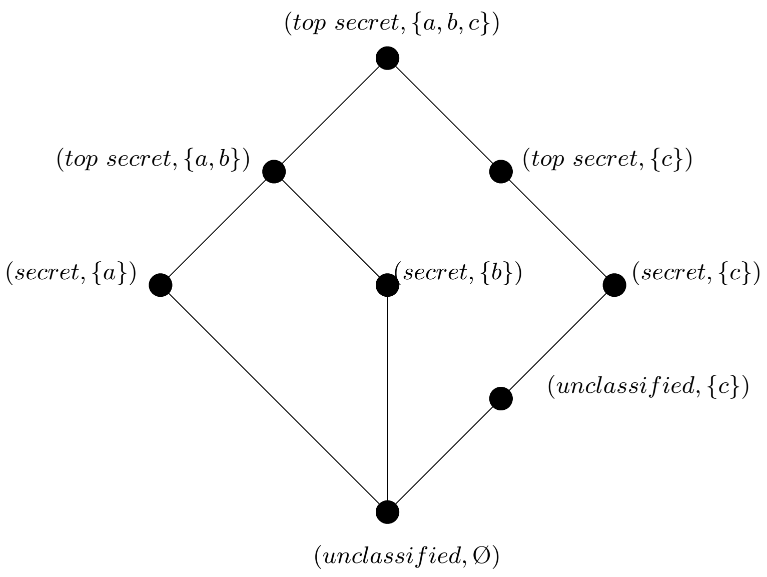

Fig. 1. An example of a security lattice.

– Subject s can read object o only if $\lambda(o) \leq \lambda(s)$.

With security labels defined as above, a subject can read an object only if the subject is at the same security level as the object or higher and the subject has access to all categories associated with the object. However, the simple-security property is not enough: some restrictions are necessary on how subjects can write or modify objects. If there are no restrictions, a secret subject can read a secret document and create an unclassified copy of it, thus providing unclassified subjects with access to secret information. To avoid this, the $\star$-*property* is introduced:

– Subject s can write object o only if $\lambda(s) \leq \lambda(o)$.

According to these two rules, information can flow only upwards: from less secure objects to more secure subjects and from less secure subjects to more secure objects.

The $\star$-property implies that higher-level subjects cannot send messages to lower-level subjects. This is not always acceptable. To avoid this problem, subjects are sometimes granted *downgrading* capabilities: they are allowed to temporarily change their security label to one that is lower in the lattice. For example, a secret user may be associated with a secret subject and an unclassified subject. Then, the user can write an unclassified document by logging in as an unclassified subject. During this session, the user will not be able to read any secret documents. It is trusted that the user will not betray the information she got during a previous session when logged in as a secret subject. In the case of a subject with downgrading capabilities being a program rather than a human, previous sessions are not an issue.

From Example 1, it is clear that not all possible combinations of security levels and categories make sense as security labels. For instance, Smith describes a lattice from military practice based on four security levels and eight categories, which potentially gives rise to 1024 labels [7]. However, the actual lattice contains only 21 elements: there are no compartments consisting of more than three categories (apart from the compartment associated with the top element) and compartments are used only in combination with two top-most security levels. Hence, development of an access control model would involve identification of meaningful combinations of security levels and categories. In the next section, we argue that attribute exploration from formal concept analysis can help organize this process in a semiautomatic fashion and, in some sense, ensure the validity of its results.

3 Building Access Control Models by Attribute Exploration

First, we recall some basic notions of formal concept analysis (FCA) [8].

Definition 2. *A* formal context *is a triple* (G, M, I)*, where* G *is a set of* objects*,* M *is a set of* attributes*, and* $I \subseteq G \times M$ *is the* incidence relation *providing information on which objects have which attributes.*

Formal contexts are naturally represented by cross tables, where a cross for a pair (g, m) means that this pair belongs to the relation I.

Definition 3. *For* $A \subseteq G$*,* $B \subseteq M$*, the following* derivation operators *are defined:*

$$A^I := \{m \in M \mid gIm \text{ for all } g \in A\}$$

$$B^I := \{g \in G \mid gIm \text{ for all } m \in B\}$$

If the relation I *is clear from context, one writes* A' *and* B' *instead of* A^I *and* B^I*.*

Derivation operators are used to define concepts:

Definition 4. *The pair* (A, B)*, where* $A \subseteq G$*,* $B \subseteq M$*,* $A' = B$*, and* $B' = A$ *is called a* (formal) concept *(of the context* K*) with* extent A *and* intent B*. A concept* (A, B) *is* more general *(*less specific*, etc.) than a concept* (C, D)*, if* $C \subseteq A$ *(equivalently,* $B \subseteq D$*).*

For $g \in G$ and $m \in M$ the sets $\{g\}'$ and $\{m\}'$ are called *object intent* and *attribute extent*, respectively.

The operation $(\cdot)''$ is a closure operator [8], i.e., it is idempotent ($X'''' = X''$), extensive ($X \subseteq X''$), and monotone ($X \subseteq Y \Rightarrow X'' \subseteq Y''$). Sets $A \subseteq G$, $B \subseteq M$ are called *closed* if $A'' = A$ and $B'' = B$. Obviously, extents and intents are closed sets. Since the closed sets form a closure system [9], the set of all formal concepts of the context $\mathbb{K}$ forms a lattice called a *concept lattice* and usually denoted by $\underline{\mathfrak{B}}(\mathbb{K})$ in FCA literature.

Definition 5. *A* many-valued context *is a quadruple (G, M, W, I), where G and M are object and attribute sets, respectively; W is a set of attribute values; and $I \subseteq G \times M \times W$ is a ternary relation satisfying the following condition:*

$$(g, m, w) \in I \text{ and } (g, m, v) \in I \Rightarrow w = v.$$

Thus, every object has *at most* one value for every attribute. For our purposes, it is convenient to assume that every object has *exactly* one value for every attribute.

To apply FCA methods to many-valued contexts, one first needs to transform them into one-valued contexts (i.e., contexts in the sense of Definition 2). A standard transformation technique here is *plain conceptual scaling.* We replace every many-valued attribute m with a set of one-valued attributes M_m by building a one-valued context (G_m, M_m, I_m), where G_m is the set of all possible values of m, i.e., $\{w \mid w \in W \text{ and } \exists g \in G : (g, m, w) \in I)\} \subseteq G_m$. The relation I_m translates every value of m into a subset of M_m. Then, the many-valued context (G, M, W, I) is replaced by a one-valued context

$$(G, \bigcup_{m \in M} \{m\} \times M_m, J),$$

where $(g, (m, n)) \in J$ if and only if there is $w \in W$ such that $(g, m, w) \in I$ and $(w, n) \in I_m$.

Now, we can easily express the Bell-LaPadula model in terms of FCA. The set H of security levels and the set C of categories (see Definition 1) constitute our attribute set. In fact, security level is a many-valued attribute, but the scaling is pretty straightforward: we use the context $(H, H, \geq)$ as a scale. In the case of Example 1, we get the following (ordinal) scale:

	unclassified	*secret*	*top secret*
unclassified	×		
secret	×	×	
top secret	×	×	×

The attribute *unclassified* is clearly redundant; so, in principle, we do not have to include the smallest security level as an attribute in the one-valued context.

The elements of G in our context are subjects and objects and the incidence relation should assign categories and levels to them. Then, security labels correspond to concept intents, and the security label of an entity (subject or object) is the intent of the least general concept covering the entity (i.e., containing the entity in its extent). The concept lattice we get is the reverse of the Bell-

LaPadula lattice, since larger intents correspond to less general concepts; the set of concept intents with usual subset order is precisely the Bell-LaPadula lattice[1].

The problem here is that a complete list of subjects and objects is usually unknown at the moment when the access control model is being developed (and, if a system is open to new users or new documents, a complete list never becomes available). To construct the lattice of security labels, the developer of the system must envision all types of potential subjects and objects and describe them in terms of security levels and compartments. It would be good to have a way of verifying that all possible cases have been considered. Besides, it would not make harm to bring some order to the process of selecting these relevant combinations. Attribute exploration is a technique that does precisely this.

Definition 6. *An expression $D \to B$, where $D \subseteq M$ and $B \subseteq M$, is called an* (attribute) implication. *An implication $D \to B$* holds *in the context (G, M, I) if all objects from G that have all attributes from the set D also have all attributes from the set B, i.e., $D' \subseteq B'$.*

There is a connection between implication sets and closure operators.

Definition 7. *An attribute set $F \subseteq M$* respects an implication *$D \to C$ if $D \nsubseteq F$ or $C \subseteq F$. A set F* respects an implication set *Σ if it respects all implications from Σ. If every set that respects an implication set Σ also respects the implication $D \to C$, then we say that $D \to C$* (semantically) follows from *Σ.*

All sets that respect some implication set form a closure system (and, hence, there is a closure operator corresponding to every implication set). A minimal (in terms of size) set of implications from which all other implications of a context semantically follow was characterized in [10]. It is called the *Duquenne-Guigues basis* or *stem base* in the literature.

Note that, having the Duquenne-Guigues basis of the context, we are able to construct the lattice of concept intents even without knowing the actual objects of the context. The join-irreducible[2] elements of this lattice correspond to object intents that have to be in the context. Such necessary object intents form the *representation context* of the concept lattice.

The goal of attribute exploration is to find this representation context and construct its lattice. The attribute exploration process is quite standard [8] and, perhaps, does not have to be formally explained here. In its simplest version, it can be outlined as follows. Given some initial (possibly empty) set of objects of a subject domain, which is known to have considerably more (perhaps, an infinite number of) such objects, and their intents, attribute exploration aims to build

[1] To get the same order in the concept lattice, we can also consider categories as objects of the context and entities (subjects and objects of the access control model), as attributes. Then, security labels are concept extents.

[2] We are working under the assumption that the order is that of concept generality, i.e., the reverse of the intent subset order. Therefore, join-irreducible intents are those that cannot be presented as intersections of other intents.

an implicational theory of the entire domain (summarized by the Duquenne–Guigues basis) and a representation context. Obviously, an object of the representation context must respect all implications from the generated implication basis and provide a counterexample for every implication that does not follow from the basis. It means, in particular, that the concept lattice of the domain is isomorphic to the concept lattice of this relatively small representation context.

The process of attribute exploration is interactive. In the general setting, the process is as follows: the computer suggests implications one by one; the user (the expert) accepts them or provides counterexamples. Attribute exploration is designed to be as efficient as possible, i.e., to suggest as few implications as possible without any loss in completeness of the result. This is achieved by generating implications from the Duquenne-Guigues basis in the order consistent with the subset ordering of implication premises (from smaller to larger premises). Then, if the user rejects an implication $D \to B$ at some step, it does not affect implications whose premise is not a superset of D: if such implications were in the basis, they will remain there.

Advanced versions of attribute exploration allow the user to input background knowledge, e.g, in form of implications the user knows to be true. Note that background knowledge is not limited to implications [11]. The presence of background knowledge usually decreases the number of questions the user has to answer, since if the answer to a certain question follows from the background knowledge (combined with already accepted implications), this question is not asked.

Background knowledge is particularly useful in the case of many-valued attributes. Moreover, it is readily available in this case: it can be automatically generated from scales. Indeed, a scale completely specifies all possible combinations of the corresponding new one-valued attributes. If we want to limit ourselves to implicational background knowledge, all that is necessary is to generate the Duquenne-Guigues basis for each scale. A method for generating the complete propositional axiomatization of the scale (with attributes interpreted as propositional variables) also exists [12] and is surprisingly similar to a method for generating the implication basis [13].

In our case, there is only one many-valued attribute: security level. It can be shown that the Duquenne–Guigues basis provides the axiomatization for the whole propositional theory of an ordinal scale, which is the type of the scale we used above for this attribute. If H is the set of security levels and $l \in H$ is the lowest level, the Duquenne–Guigues basis consists of the following implications:

$$\emptyset \to \{l\}$$

and

$$\{l, h\} \to \{k \mid k \leq h\}$$

for all $h \in H$ such that $\exists j \in H(l < j < h)$. Therefore, in our case, we can do with only background implications. The basis for the context in Example 1 is as follows:

$$\emptyset \to \{unclassified\};$$

$$\{unclassified, top\ secret\} \to \{unclassified, secret, top\ secret\}.$$

By entering these implications as background knowledge we avoid questions on implications between these attributes. In the case of Example 1, we would start

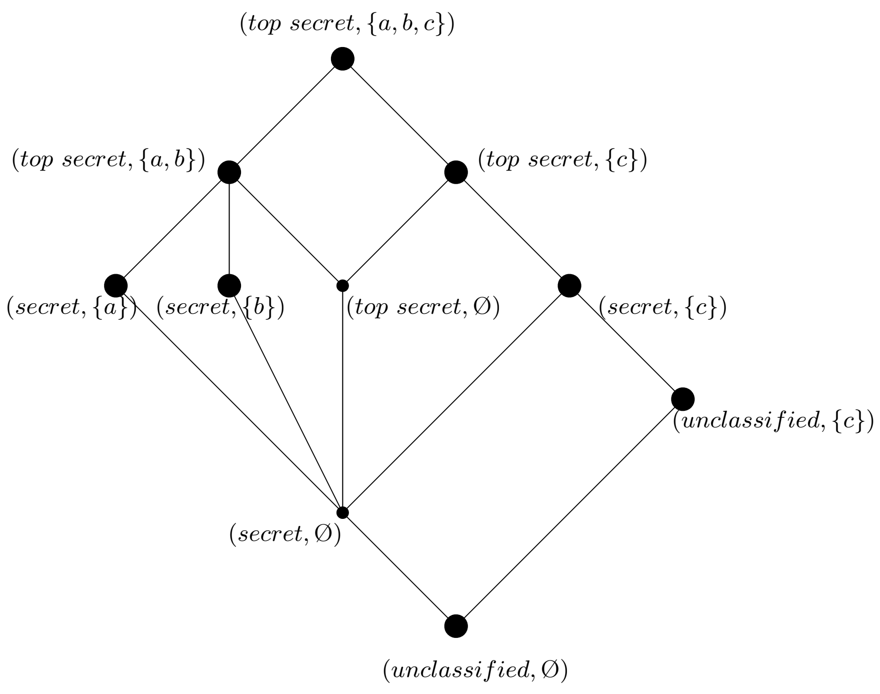

Fig. 2. The lattice obtained by attribute exploration based on Example 1. The order is the subset order of intents. Smaller nodes correspond to nodes absent from the lattice in Figure 1.

with the context $(\emptyset, M, \emptyset)$, where $M = \{top\ secret, secret, unclassified, a, b, c\}$ and the two background implications above. The first question asked by the system would be:

Is it true that all objects must have all attributes?

This is the most strong implication, and there are no counterexamples to it in our so far empty context. However, the answer is clearly "no", and we have to enter an object that lacks some attributes. Suppose that we enter an object with the intent $\{top\ secret, secret, unclassified, a, b\}$. Since this will be the only object of our context, the system assumes that all objects from the domain are like that and asks if it is true that every entity is labeled as at least $(top\ secret, a, b)$, to which we have to enter another counterexample, say, $\{top\ secret, secret, unclassified, c\}$. Then, we may have to enter object intents

$\{secret, unclassified, a\}$ and $\{unclassified, c\}$. The system is not going to ask us whether every object is labeled as at least $unclassified$, as it knows from the background implications that this is so.

It is obvious that the process always stops at some point. In fact, the number of questions we are going to answer is equal to the sum of the sizes of the implication basis and representation context. The latter, however, depends on what objects we enter as counterexamples.

The resulting lattice may (and in the case of Example 1, will) contain more elements than it is necessary, that is, there may be some concepts that do not correspond to any realistic security labels (see Figure 2). This is because the concept lattice is closed under intersection of intents, whereas the security lattice, generally speaking, does not have to be closed under intersection of security labels. For instance, the lattice we obtain from attribute exploration on Example 1 will contain an intent $\{top\ secret, secret, unclassified\}$ obtained as the intersection of $\{top\ secret, secret, unclassified, a, b\}$ and $\{top\ secret, secret, unclassified, c\}$. In the lattice of Figure 1, the meet of the labels corresponding to these two intents is the bottom element. Some additional effort on the side of the system developer is required if they want to decrease the size of the lattice. On the other hand, the existence of such extra concepts suggests that the choice of security levels and categories or their distribution among security labels might not be optimal. For example, all labels in Figure 1 that do not contain c can easily be marked as $unclassified$: this would not affect the information flow policy, but would simplify the model.

As a matter of fact, we can even argue that the labeling is optimal only if there is a one-to-one correspondence between security categories and levels, on the one hand, and join-irreducible[3] elements of the lattice, on the other hand, and the lattice is closed under intersection. In other words, a node should contain a category or a level if and only if it is the one that corresponds to this category or level or if it is above such unique corresponding node. In the case of Example 1, the level *secret* corresponds to the node $(secret, \{c\})$ and, consequently, should not co-occur with compartments $\{a\}$, $\{b\}$, and $\{a, b\}$. Such approach ensures that the number of categories and levels is minimal and that the labels are as small as possible.

4 Combining Confidentiality and Integrity Models

As said above, the Bell-LaPadula model is concerned with confidentiality. Another issue that requires attention is *integrity*:

Integrity: prevention of unauthorized modification of information.

One of the most common models dealing with integrity is the Biba model [3]. Biba proposed several integrity models, of which the best known is *strict*

[3] Assuming the subset order as in Figure 1.

integrity. This model is dual to the Bell-LaPadula model: again we have a lattice of (integrity) labels, but this time the information is allowed to flow only downwards—from entities with higher integrity to entities with lower integrity—this is to prevent corruption of "clean" high-level entities by "dirty" low-level entities [4]. Assuming that $\omega(e)$ is the integrity level of the entity e, the rules of the model are formulated as follows [1]:

Simple-integrity property: Subject s can read object o only if $\omega(s) \leq \omega(o)$
Integrity $\star$-property: Subject s can write object o only if $\omega(o) \leq \omega(s)$

Of course, it is generally not very important whether the information flows only upwards or only downwards. However, it becomes important if we want to combine the Bell-LaPadula and Biba models in order to address both confidentiality and integrity issues. In the combination of these two models, the rules of information flow are as follows [1]:

– Subject s can read object o only if $\lambda(o) \leq \lambda(s)$ and $\omega(s) \leq \omega(o)$.
– Subject s can write object o only if $\lambda(s) \leq \lambda(o)$ and $\omega(o) \leq \omega(s)$.

If the same security labels are used both for Bell-LaPadula and Biba models, then the information flow policy boils down to allowing subjects to read and write only objects from their own security level and compartment. The case when labels are different for the two models is more interesting (and useful).

Let $\mathbb{K}_1 = (G, M, I)$ and $\mathbb{K}_2 = (N, G, J)$ be formal contexts. By using G as the object set of $\mathbb{K}_1$ and as the attribute set of $\mathbb{K}_2$, we address the difference in the information flow direction of the two models. We want to combine these contexts into one structure in a way that preserves both orders of the corresponding concept lattices. The largest possible combination of the concept lattices is their product: $(\underline{\mathfrak{B}}(\mathbb{K}_1) \times \underline{\mathfrak{B}}(\mathbb{K}_2), \leq)$, where $(c_1, c_2) \leq (d_1, d_2)$ if and only if $c_1 \leq d_1$ and $c_2 \leq d_2$ (with respect to the usual "generality" order on concepts). The join and meet operations of the lattice $(\underline{\mathfrak{B}}(\mathbb{K}_1) \times \underline{\mathfrak{B}}(\mathbb{K}_2), \leq)$ are obvious:

$$(c_1, d_1) \vee (c_2, d_2) = (c_1 \vee c_2, d_1 \vee d_2)$$

$$(c_1, d_1) \wedge (c_2, d_2) = (c_1 \wedge c_2, d_1 \wedge d_2)$$

Here, joins and meets of concept pairs are taken in their respective lattices.

However, the product is too large for our purposes: it reflects the structure of each of the component lattices, but it fails to capture the dependencies between the elements of different lattices. These dependencies are given via the relations between the set G and corresponding sets (M and N) in the two contexts.

So, we are looking for an adequate subset $\underline{\mathfrak{B}}_{\Diamond}(\mathbb{K}_1, \mathbb{K}_2)$ of $\underline{\mathfrak{B}}(\mathbb{K}_1) \times \underline{\mathfrak{B}}(\mathbb{K}_2)$. The first observation is that $\underline{\mathfrak{B}}_{\Diamond}(\mathbb{K}_1, \mathbb{K}_2)$ must contain concept pairs of the form

$$((\{g\}^{II}, \{g\}^{I}), (\{g\}^{J}, \{g\}^{JJ})) \qquad (1)$$

for every $g \in G$. That is, every element of G must get exactly the same description as it is given by the two initial contexts.

Then, we have several options for how to proceed. We can adopt a minimalist approach and add to $\underline{\mathfrak{B}}_\Diamond(\mathbb{K}_1, \mathbb{K}_2)$ only those concepts from $\underline{\mathfrak{B}}(\mathbb{K}_1) \times \underline{\mathfrak{B}}(\mathbb{K}_2)$ that are required to make our initial set of concepts a lattice. This makes sense if we know that G is the entire object set of our domain (rather than a set that gives rise to representation contexts for $\mathbb{K}_1$ and $\mathbb{K}_2$) and if all we need is to properly order elements from G. In the case of combining access control models, this is indeed all we need (ideally, every concept of the lattice should be a meaningful security label for some subject or object), but, unfortunately, the set G does not necessarily contain all possible entities, but only those enough to get a representation context for each of the two models. Therefore, we choose a different approach.

The (almost) maximalist approach we choose is to take the sublattice of $\underline{\mathfrak{B}}(\mathbb{K}_1) \times \underline{\mathfrak{B}}(\mathbb{K}_2)$ generated by concept pairs from (1), i.e., the smallest subset of $\underline{\mathfrak{B}}(\mathbb{K}_1) \times \underline{\mathfrak{B}}(\mathbb{K}_2)$ containing all object pairs (1) and closed under the join and meet operations. From now on, $\underline{\mathfrak{B}}_\Diamond(\mathbb{K}_1, \mathbb{K}_2)$ denotes this lattice. This approach ensures, in particular, that every concept of each component has a counterpart in the resulting lattice:

1. For every $c \in \underline{\mathfrak{B}}(\mathbb{K}_1)$, there is $d \in \underline{\mathfrak{B}}(\mathbb{K}_2)$ such that $(c, d) \in \underline{\mathfrak{B}}_\Diamond(\mathbb{K}_1, \mathbb{K}_2)$;
2. For every $d \in \underline{\mathfrak{B}}(\mathbb{K}_2)$, there is $c \in \underline{\mathfrak{B}}(\mathbb{K}_1)$ such that $(c, d) \in \underline{\mathfrak{B}}_\Diamond(\mathbb{K}_1, \mathbb{K}_2)$;

This is useful if we believe that every concept of the initial lattices corresponds to a (complete w.r.t. the given attributes) description of some entity (for the combination of the Bell-LaPadula and Biba models, to a security label to be attached to some subject or object)—then, we cannot discard any concept.

We consider two important subsets of $\underline{\mathfrak{B}}_\Diamond(\mathbb{K}_1, \mathbb{K}_2)$:

$$\underline{\mathfrak{B}}_\triangle(\mathbb{K}_1, \mathbb{K}_2) := \{((A^{II}, A^I), (B^{JJ}, B^J)) \mid A \subseteq G, B = \bigcup_{a \in A} \{a\}^J\}$$

and

$$\underline{\mathfrak{B}}_\bigtriangledown(\mathbb{K}_1, \mathbb{K}_2) := \{((B^I, B^{II}), (A^J, A^{JJ})) \mid A \subseteq G, B = \bigcup_{a \in A} \{a\}^I\}$$

It can be easily seen that each of the sets $\underline{\mathfrak{B}}_\triangle(\mathbb{K}_1, \mathbb{K}_2)$ and $\underline{\mathfrak{B}}_\bigtriangledown(\mathbb{K}_1, \mathbb{K}_2)$ is a lattice, and (possibly, without the bottom and top elements, respectively) they are the set of all joins and the set of all meets of subsets of pairs (1), respectively.

We now define two contexts whose concept lattices are isomorphic to the lattices $\underline{\mathfrak{B}}_\triangle(\mathbb{K}_1, \mathbb{K}_2)$ and $\underline{\mathfrak{B}}_\bigtriangledown(\mathbb{K}_1, \mathbb{K}_2)$.

Definition 8. *Let $\mathbb{K}_1 = (G, M, I)$ and $\mathbb{K}_2 = (N, G, J)$ be formal contexts. Then,*

$$\mathbb{K}_1 \triangle \mathbb{K}_2 := (G, G \cup M, I \cup I_\triangle),$$

where $I_\triangle = \{(g, h) \mid g \in G, h \in G,$ and $\{g\}^J \subseteq \{h\}^J\}$, and

$$\mathbb{K}_1 \bigtriangledown \mathbb{K}_2 := (G \cup N, G, J \cup J_\bigtriangledown),$$

where $J_\bigtriangledown = \{(g, h) \mid g \in G, h \in G,$ and $\{h\}^I \subseteq \{g\}^I\}$.

Proposition 1. *The concept lattice* $\underline{\mathfrak{B}}(\mathbb{K}_1 \bigtriangleup \mathbb{K}_2)$ *is isomorphic to the lattice* $\underline{\mathfrak{B}}_{\bigtriangleup}(\mathbb{K}_1, \mathbb{K}_2)$ *and the concept lattice* $\underline{\mathfrak{B}}(\mathbb{K}_1 \bigtriangledown \mathbb{K}_2)$ *is isomorphic to the lattice* $\underline{\mathfrak{B}}_{\bigtriangledown}(\mathbb{K}_1, \mathbb{K}_2)$.

Proof. We define a mapping $f_{\bigtriangleup} : \underline{\mathfrak{B}}(\mathbb{K}_1 \bigtriangleup \mathbb{K}_2) \to \underline{\mathfrak{B}}_{\bigtriangleup}(\mathbb{K}_1, \mathbb{K}_2)$ as follows. For $A, C \subseteq G$,

$$f_{\bigtriangleup}((A, A^I \cup C)) = ((A^{II}, A^I), (C^J, C)).$$

To show that this mapping is well-defined we need to prove that, for every concept (A, B) of $\underline{\mathfrak{B}}(\mathbb{K}_1 \bigtriangleup \mathbb{K}_2)$, there is $C \subseteq G$ such that $B = A^I \cup C^{JJ}$. It is obvious that $B \cap M = A^I$. One can see that $B \cap G$ is $(\cdot)^J$-closed, as $B \cap G = \{h \mid h \in G \text{ and } \forall g \in A(\{g\}^J \subseteq \{h\}^J)\} = (\bigcup_{g \in A} \{g\}^J)^J$. Note that $((A^{II}, A^I), (C^J, C)) \in \underline{\mathfrak{B}}_{\bigtriangleup}(\mathbb{K}_1, \mathbb{K}_2)$, since $C = (\bigcup_{g \in A} \{g\}^J)^J$.

Clearly, $f_{\bigtriangleup}$ is order-preserving. The inverse mapping is as follows:

$$f_{\bigtriangleup}^{-1}((A^{II}, A^I), (B^{JJ}, B^J)) = (\{a \in A^{II} \mid \{a\}^J \subseteq B^{JJ}\}, A^I \cup B^J).$$

Without loss of generality, we may assume that $B = \bigcup_{a \in A} \{a\}^J$. Then, $A \subseteq \{a \in A^{II} \mid \{a\}^J \subseteq B^{JJ}\}$, and it is easy to see that $f_{\bigtriangleup}^{-1}((A^{II}, A^I), (B^{JJ}, B^J))$ is indeed a concept of $\mathbb{K}_1 \bigtriangleup \mathbb{K}_2$.

The mapping $f_{\bigtriangledown} : \underline{\mathfrak{B}}(\mathbb{K}_1 \bigtriangledown \mathbb{K}_2) \to \underline{\mathfrak{B}}_{\bigtriangledown}(\mathbb{K}_1, \mathbb{K}_2)$ and the inverse mapping are given below:

$$f_{\bigtriangledown}((A^J \cup C, A)) = ((C, C^I), (A^J, A^{JJ}));$$

$$f_{\bigtriangledown}^{-1}((B^I, B^{II}), (A^J, A^{JJ})) = (A^J \cup B^I, \{a \in A^{JJ} \mid \{a\}^I \subseteq B^{II}\}).$$

We omit the rest of the proof. □

Let us consider implications in these contexts. How should they be interpreted? An implication of $\mathbb{K}_1 \bigtriangleup \mathbb{K}_2$ may contain elements of G, as well as elements of M. An implication $A \to B$ holds in $\mathbb{K}_1 \bigtriangleup \mathbb{K}_2$ if and only if, for all $g \in G$, $B \cap M \subseteq g^I$ and $g^J \subseteq (B \cap G)^J$ whenever $A \cap M \subseteq g^I$ and $g^J \subseteq (A \cap G)^J$. In words:

> If (in the two initial contexts) an element of G is related to all elements from $A \cap M$ and no elements from $N \setminus (A \cap G)^J$, then it is related to all elements from $B \cap M$ and no elements from $N \setminus (B \cap G)^J$.

An (object) implication $C \to D$ over $G \cup N$ in the context $\mathbb{K}_1 \bigtriangledown \mathbb{K}_2$ reads as follows:

> If (in the two initial contexts) an element from G is related to all elements from $C \cap N$ and no elements from $M \setminus (C \cap G)^I$, then it is related to all elements from $D \cap N$ and no elements from $M \setminus (D \cap G)^I$.

These implications express the relation between the attributes of M and negations of attributes from N (and vice versa). Note however that the implication system of, e.g., $\mathbb{K}_1 \bigtriangleup \mathbb{K}_2$ is different from the implication system of $(G, M \cup N, I \cup (G \times N) \setminus J^{-1})$, the context obtained by combining attributes from M

and negations of attributes from N. The difference is due to the fact that, in the case of $\mathbb{K}_1 \bigtriangleup \mathbb{K}_2$, our additional attributes are only certain conjunctions of negated attributes from N.

Now, we outline how attribute exploration can be organized in the case when the set G described by two contexts (G, M, I) and (N, G, J) is not completely available. We take the problem of combining Bell-LaPadula and Biba model as an example.

Suppose that the elements of M are confidentiality categories and confidentiality levels (of the Bell-LaPadula model) and the elements of N are integrity categories and levels (of the Biba model).

We start by constructing $\mathbb{K}_1 = (G, M, I)$ and $\mathbb{K}_2 = (N, G, J)$ using standard attribute exploration and object exploration[4], respectively. The user is asked to verify implications over M and implications over N (but not implications over $M \cup N$). However, when providing a counterexample to one of these implications, the user must enter its complete description in terms of both M and N.

Now, we know all confidentiality labels and all integrity labels. What we do not know is what combinations of confidentiality and integrity labels are possible. We construct contexts $\mathbb{K}_1 \bigtriangleup \mathbb{K}_2$ and $\mathbb{K}_1 \bigtriangledown \mathbb{K}_2$. At this stage, the concept lattices of these contexts might not be exactly what we need, but they do contain every combination of confidentiality and security labels attached to at least one element from G. To explore other possibilities, we use attribute exploration on $\mathbb{K}_1 \bigtriangleup \mathbb{K}_2$ and object exploration on $\mathbb{K}_1 \bigtriangledown \mathbb{K}_2$ to build appropriate lattices.

Questions asked during attribute exploration of $\mathbb{K}_1 \bigtriangleup \mathbb{K}_2$ sound more natural than one could imagine by looking at the previously given formulas. If $M = \{a, b, c\}$ and $N = \{d, e, f\}$, we may be asked to verify an implication $\{\neg d\} \to \{b, \neg e\}$, which can be understood as "if a subject may not write to d, it may not write to e, either, but may read from b instead".

Having built $\mathbb{K}_1 \bigtriangleup \mathbb{K}_2$ and $\mathbb{K}_1 \bigtriangledown \mathbb{K}_2$, we can construct $\underline{\mathfrak{B}}_{\Diamond}(\mathbb{K}_1, \mathbb{K}_2)$ by applying the join and meet operations to $f_{\bigtriangleup}(\underline{\mathfrak{B}}(\mathbb{K}_1 \bigtriangleup \mathbb{K}_2)) \cup f_{\bigtriangledown}(\underline{\mathfrak{B}}(\mathbb{K}_1 \bigtriangledown \mathbb{K}_2))$.

5 Conclusion

We have discussed some lattice-related aspects of access control models. It seems likely that attribute exploration can be useful in their construction. The process may take long, but it is worth the effort in serious applications: attribute exploration explicitly forces the system developer to consider issues that can be overlooked otherwise. Although a lattice produced by attribute exploration can contain more elements than it is necessary for a given set of security labels and categories, it is a better starting point than the lattice of all subsets and it

[4] Object exploration is a process dual to attribute exploration: the user is asked to confirm an implication between objects (also defined dually to the attribute implication) or enter a new attribute as a counterexample. The only reason we are talking about object exploration is that the set N is the object set of $\mathbb{K}_2$. One can think of object exploration as attribute exploration in the transposed context (where objects and attributes change places).

still contains all necessary elements. In fact, the presence of extra elements may indicate that the choice of security labels and categories is not optimal.

We have also shown how a model addressing confidentiality and a model addressing integrity can be combined within one lattice and how this lattice can be obtained with the help of attribute exploration. This is only a step towards formalizing the combined model and further research is necessary to estimate the benefits of the proposed approach and to evaluate other possibilities.

References

1. Sandhu, R.: Lattice-based access control models. IEEE Computer **26**(11) (1993) 9–19
2. Denning, D.: A lattice model of secure information flow. Comm. ACM **19**(5) (1976) 236–243
3. Biba, K.: Integrity considerations for secure computer systems. Report TR-3153, Mitre Corporation, Bedford, Mass. (1977)
4. Gollmann, D.: Computer Security. John Wiley & Sons Ltd, Chichester (1999)
5. Bell, D., LaPadula, L.: Secure computer systems: Mathematical foundations and model. Report M74-244, Mitre Corporation, Bedford, Mass. (1975)
6. Lipner, S.: Nondiscretionary controls for commercial applications. In: Proc. IEEE Symp. Security and Privacy, Los Alamitos, Calif., IEEE CS Press (1982) 2–10
7. Smith, G.: The Modeling and Representation of Security Semantics for Database Applications. PhD thesis, George Mason Univ., Fairfax, Va. (1990)
8. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer, Berlin (1999)
9. Birkhoff, G.: Lattice Theory. Amer. Math. Soc. Coll. Publ., Providence, R.I. (1973)
10. Guigues, J.L., Duquenne, V.: Familles minimales d'implications informatives resultant d'un tableau de données binaires. Math. Sci. Humaines **95** (1986) 5–18
11. Ganter, B.: Attribute exploration with background knowledge. Theoretical Computer Science **217** (1999) 215–233
12. Ganter, B., Krausse, R.: Pseudo models and propositional horn inference. Technical Report MATH-AL-15-1999, Technische Universität Dresden, Germany (1999)
13. Ganter, B.: Two basic algorithms in concept analysis. Preprint 831, Technische Hochschule Darmstadt (1984)