Annales UMCS Informatica AI X, 2 (2010) 143-159
DOI: 10.2478/v10065-010-0060-6

Annales UMCS
Informatica
Lublin-Polonia
Sectio AI
http://www.annales.umcs.lublin.pl/

Modelling role hierarchy structure using the Formal Concept Analysis

Ścibor Sobieski[1]*, Bartosz Zieliński[1, 2]†

[1] *Department of Theoretical Physics II, University of Lódź,*
ul. Pomorska 149/153, 90-236 Łódź, Poland.
[2] *Institute of Mathematics, PAN,*
ul. Śniadeckich 8, 00-956 Warszawa, Poland.

Abstract – We demonstrate how one can use the formal concept analysis (FCA) to obtain the role hierarchy for the role based access control from the existing access control matrix. We also discuss assesed by means of FCA the quality of security system and finding users with excess permissions.

1 Introduction

The formal concept analysis emerged from the paper by Wille [12]. Mathematically it is part of the lattice theory. FCA found many applications as a method of nonstatistical data analysis and visualisation in the areas ranging from software engineering and data mining to psychology and sociology (see e.g. [10]). This work inspired by [7], demonstrates how to use the concept lattices to discover the modular structure of legacy code and assess its quality.

Role based access control (RBAC) [4] can be thought of as a modularisation of permissions. Instead of assigning directly to the user access rights to various objects in the system (e.g, files and folders in the operating system, tables or even parts of table rows in the case of the database system), one does it through roles, which can be thought of as subsets of permissions. Roles clearly simplify security the administration task. One can design a role hierarchy using the background knowledge about the structure of the organisation and the actual roles in the organisation which are implemented by users.

*scibor.sobieski@uni.lodz.pl
†bartosz.zielinski@uni.lodz.pl

In this paper we consider another scenario. Many organisations do not use RBAC in their database systems or file access systems, but assign permissions directly. When a system contains a small number of users and objects this is reasonable, but when a system grows the administration it can become unmanagable. We present a method, based on the formal concept analysis, which faciliates discovering the roles from the existing permission to the user assignements, i.e, we "modularise" an existing permission system (c.f. [7]). The assumption here is that most of the permissions are given correctly. On the other hand, our method assists in finding those few users who are given excess access rights.

The formal concept analysis was used in several papers in connection with roles and security (e.g., [3, 8, 9]). In [9] the authors used FCA to discover the hierarchy of security labels. In [3, 8] the authors considered a formal context of permissions to roles relation to find the full role hierarchy. (This context is different from that used here.)

2 Preliminaries

2.1 Lattices and order

(See e.g. [2] for further information.) Recall that a *partial order* on a set P is a binary relation " $\leq$ " $\subseteq P \times P$ satisfying all $p, q, r \in P$:

$$p \leq p, \quad p \leq q \text{ and } q \leq p \text{ implies } p = q, \quad p \leq q \text{ and } q \leq r \text{ implies } p \leq r.$$

We call a pair $(P, \leq)$, where P is a set and $\leq$ is a particular partial order on P, a *partially ordered set* (POSET). When no ambiguity is possible, we will often abuse the notation abbreviating $(P, \leq)$ to P. We will also write $p < q$ if $p \leq q$ and $p \neq q$. As an example of a partially ordered set take $(2^X, \subseteq)$ where X is any set. If $(P, \leq)$ is a poset and $A \subseteq P$ then $(A, \text{" } \leq \text{ "} \cap A \times A)$ is a poset as well (with the induced order).

We say that $p \in P$ *covers* $q \in Q$, which we denote by $q \prec p$, if $q < p$ and $q \leq r \leq p$ implies $r = q$ or $r = p$, for any $r \in P$. Observe that a finite poset is uniquely defined by a reflexive transitive closure of the covering relation.

Finite posets are visualised using *Hasse diagrams*. We draw a poset $(P, \leq)$ as a directed graph with P as the set of vertices and with the covering relation $\prec$ as the set of edges. However, instead of decorating edges with arrows to show direction, we draw vertex p strictly above vertex q if $q \prec p$. Fig. 1 is an example of a Hasse diagram of a five element poset $P = \{\bot, \top, u, w, v\}$, where the covering relation equals $\{\bot \prec w \prec u \prec \top, \bot \prec v \prec \top\}$:

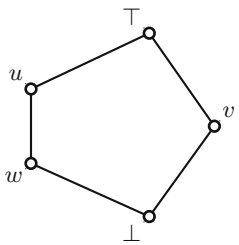

Fig. 1

An element $\top \in P$ (resp. $\bot \in P$) is called *top* or *the largest* (resp. *bottom* or *the smallest*) element of a POSET P if $p \leq \top$ (resp. $\bot \leq p$) for all $p \in P$. For a subset $A \subseteq P$ we define:

- The *set of maximal elements* $\mathbf{max}_{(P,\leq)} A := \{p \in A \mid p \leq q \Rightarrow p = q,\ \forall q \in A\}$.
- The *set of minimal elements* $\mathbf{min}_{(P,\leq)} A := \{p \in A \mid q \leq p \Rightarrow p = q,\ \forall q \in A\}$.
- The *supremum* (also called the *join*) $\bigvee_{(P,\leq)} A \in P$ defined by the conditions:
  (1) $p \leq \bigvee_{(P,\leq)} A$ for all $p \in A$.
  (2) For any $p \in \{q \in P \mid r \leq q,\ \forall r \in A\}$ we have $\bigvee_{(P,\leq)} A \leq p$.
- The *infimum* (also called the *meet*) $\bigwedge_{(P,\leq)} A \in P$ defined by the conditions:
  (1) $\bigwedge_{(P,\leq)} A \leq p$ for all $p \in A$.
  (2) For any $p \in \{q \in P \mid q \leq r,\ \forall r \in A\}$ we have $p \leq \bigwedge_{(P,\leq)} A$.

Top and bottom elements, as well as any join or meet are clearly unique if they exist. We will write $p \vee q$ instead of $\bigvee\{p, q\}$ and $p \wedge q$ instead of $\bigwedge\{p, q\}$. For brevity, when no ambiguity is possible, we will write **max**, **min**, $\bigvee$, $\bigwedge$ instead of $\mathbf{max}_{(P,\leq)}$, $\mathbf{min}_{(P,\leq)}$, $\bigvee_{(P,\leq)}$, $\bigwedge_{(P,\leq)}$. Note that $\bigwedge_{(P,\leq)} P = \bigvee_{(P,\leq)} \emptyset$ is the bottom element and $\bigvee_{(P,\leq)} P = \bigwedge_{(P,\leq)} \emptyset$ is the top element.

A POSET $(P, \leq)$ is called a *lattice* if for any finite and non-empty subset A of P the supremum $\bigvee A$ and infimum $\bigwedge A$ exist. It is called a *complete* lattice if $\bigvee A$ and $\bigwedge A$ exist for *any* subset A. Note that any finite lattice is complete. In particular, any finite lattice has a top and bottom elements. In this paper we are concerned with finite lattices only.

Let $(P, \leq)$ be a lattice. A subset $A \subseteq P$ is called *join dense* (resp. *meet dense*) in P if for all $p \in P$

$$p = \bigvee\{q \in A \mid q \leq p\} \quad (\text{resp. } p = \bigwedge\{q \in A \mid p \leq q\}). \tag{1}$$

Definition 3. Let X be any set. A map $C : 2^X \to 2^X$ is called a *closure operator* if

$$A \subseteq C(A), \quad C(A) = C(C(A)), \quad A \subseteq B \Rightarrow C(A) \subseteq C(B),$$

for any $A, B \subseteq X$. We define a set of closed subsets $X_C = \{A \subseteq X \mid C(A) = A\}$.

X_C with the inclusion order is a complete lattice. Explicitly, for any $\Psi \subseteq X_C$

$$\bigwedge_{(X_C,\subseteq)} \Psi = \bigcap \Psi, \qquad \bigvee_{(X_C,\subseteq)} \Psi = C\left(\bigcup \Psi\right). \tag{2}$$

2.2 Formal concept analysis

(See e.g. [1, 2, 6, 10] for further information.) Philosophically speaking a concept consists of the *extent*: the set of "things" representing the object and the *intent*: the set of attributes or properties shared by all objects from the extent, and distinguishing them from representatives of other concepts. Because each set of things can be an extent of a concept (trivially, the elements of this set have the distinguishing property of belonging to this set), to make concepts useful we limit ourselves to a specific *context*:

Definition 4. A *formal context* (G, M, R) consists of

- the set of objects G,
- the set of attributes M,
- the binary relation $R \subseteq G \times M$, where we read gRm as "the object g has the attribute m".

Formal contexts are usually written as tables (see e.g. Fig. 7) with the rows indexed by objects and columns indexed by attributes. One puts a cross (**x**) in the (g, m)'th entry if gRm, otherwise one leaves an empty space.

A binary relation $R \subseteq G \times M$ gives rise to a pair (called a *Galois connection*) of inclusion order reversing maps $2^G \underset{\triangleleft}{\overset{\triangleright}{\leftrightarrows}} 2^M$, defined for any $A \subseteq G$, $B \subseteq M$ by

$$A^{\triangleright} := \{m \in M \mid gRm,\ \forall g \in A\}, \quad B^{\triangleleft} := \{g \in G \mid gRm,\ \forall m \in B\}. \tag{3}$$

One can prove that $()^{\triangleright\triangleleft} : 2^G \to 2^G$ and $()^{\triangleleft\triangleright} : 2^M \to 2^M$ are closure operators (Def. 3). Moreover, the images of the subsets of G and M under $()^{\triangleright}$ and $()^{\triangleleft}$, respectively, are closed, and the restrictions $G_{\triangleright\triangleleft} \underset{\triangleleft}{\overset{\triangleright}{\leftrightarrows}} M_{\triangleleft\triangleright}$ are a pair of mutually inverse, order reversing bijections.

Definition 5. A *formal concept* of a formal context (G, M, R) is a pair $(A, B) \in 2^G \times 2^M$ such that

$$A^{\triangleright} = B, \quad B^{\triangleleft} = A.$$

We call A an *extent* of a concept and B an *intent*. We denote the set of formal concepts of a formal context (G, M, R) by $\mathcal{B}(G, M, R)$.

By the definition the remarks about about the extents and intents of formal concepts are closed subsets of G and M, respectively, and the maps

$$\textbf{ext} : \mathcal{B}(G, M, R) \longrightarrow G_{\triangleright\triangleleft}, \quad \textbf{int} : \mathcal{B}(G, M, R) \longrightarrow M_{\triangleleft\triangleright} \tag{4}$$

assigning to a concept its extent and intent, respectively, are bijective.

$\mathcal{B}(G, M, R)$ has a natural partial order relation of the generality:

$$(A, B) \leq (A', B') \quad \Leftrightarrow \quad A \subseteq A' \quad \Leftrightarrow \quad B \supseteq B', \tag{5}$$

i.e., the concept (A, B) is less general than (A', B') if and only if all objects belonging to its extent also belong to the extent of (A', B') or, equivalently, if and only if all the attributes from the intent of a more general concept (A', B') also belong to the intent of a more specific concept (A, B). With this order, the maps **ext** and **int** are the order and order reversing isomorphisms of posets, respectively. Moreover, $\mathcal{B}(G, M, R)$ is a complete lattice called the *concept lattice* of a context (G, M, R). The suprema and infima of a family $\{(A_j, B_j) \mid j \in J\}$ of concepts are expressed explicitly by

$$\bigvee\{(A_j, B_j) \mid j \in J\} = \left(\left(\bigcup\{A_j \mid j \in J\}\right)^{\triangleright\triangleleft}, \bigcap\{B_k \mid k \in J\}\right),$$
$$\bigwedge\{(A_j, B_j) \mid j \in J\} = \left(\bigcap\{A_j \mid j \in J\}, \left(\bigcup\{B_k \mid k \in J\}\right)^{\triangleleft\triangleright}\right). \tag{6}$$

One can prove that the maps

$$\gamma : G \longrightarrow \mathcal{B}(G, M, R), \qquad g \longmapsto (g^{\triangleright\triangleleft}, g^{\triangleright}),$$
$$\mu : M \longrightarrow \mathcal{B}(G, M, R), \qquad m \longmapsto (m^{\triangleleft}, m^{\triangleleft\triangleright})$$

have the following properties:

- $\gamma(G)$ is the join dense and $\mu(M)$ is the meet dense in $\mathcal{B}(G, M, R)$, i.e., each element of $\mathcal{B}(G, M, R)$ can be presented as the join of some subset of $\gamma(G)$ and the meet of some subset of $\mu(M)$,
- gRm if and only if $\gamma(g) \leq \mu(m)$.

The elements of $\gamma(G)$ (resp. $\mu(M)$) are called *object* (resp. *attribute*) concepts. In words $\gamma(g)$ is the smallest concept containing g in its extent and $\mu(m)$ is the largest concept with m in its intent. When drawing the Hasse diagrams of concept lattices, we use γ and μ to label the concepts (this is called *reduced labelling*):

- We write g below the concept $\gamma(g)$.
- We write m above $\mu(m)$.
- We leave any concept which is neither object nor attribute concept unlabelled.

There are many computer programs which generate concept lattices and their visualisations, in the form of Hasse diagrams, out of the formal contexts. The lattices in this paper (e.g., Fig 8) were created using *Concept Explorer* ([14]). The size of nodes is proportional to the number of new objects belonging to the concept's extent (i.e., for each concept $x \in \mathcal{B}(G, M, R)$ the number of objects $g \in G$ such that $x = \gamma(g)$. The blue (resp. black) halfdiscs indicate attribute (resp. object) concepts. The extent of a concept x can be recovered from the reduced labelling by collecting all object labels on all ascending paths from bottom to x. Similarily, one recovers the intent of x by collecting all attribute labels on all descending paths from top to x.

Note that while the formal context (G, M, R) can be recovered from the concept lattice and labelling maps γ and μ (using the property $\gamma(g) \leq \mu(m)$ if and only if gRm), the lattice structure alone is not sufficient for that. In fact, we have the following result:

Proposition 1. Let (G, M, R) be a formal context. A complete lattice L is order isomorphic to $\mathcal{B}(G, M, R)$ if and only if there exist two maps $\hat{\gamma} : G \to L$ and $\hat{\mu} : M \to L$ satisfying conditions

- $\hat{\gamma}(G)$ is the join dense and $\hat{\mu}(M)$ is the meet dense in L,
- gRm if and only if $\hat{\gamma}(g) \leq \hat{\mu}(m)$.

By the above proposition any complete lattice L is the lattice of concepts of a certain context, e.g., $L \simeq \mathcal{B}(L, L, \leq)$. Different contexts may give rise to isomorphic

concept lattices. For instance, usually $(G, M, R) \neq (\mathcal{B}(G, M, R), \mathcal{B}(G, M, R), \leq)$ but $\mathcal{B}(G, M, R) \simeq \mathcal{B}(\mathcal{B}(G, M, R), \mathcal{B}(G, M, R), \leq)$.

2.3 Access control matrix

Let $\mathcal{O}$ denote the set of all objects in our system accessible to the users. In the case of operating system, it may be the set of all files, folders and volumes, while in the case of a database system we may take for $\mathcal{O}$ the set of all tables, views and stored subprograms.

With each object $o \in \mathcal{O}$ we associate the set of potential permissions $\mathcal{P}_o$. For instance if o is a file then $\mathcal{P}_o = \{\text{READ}, \text{WRITE}, \text{EXECUTE}, \text{DELETE}\}$. Note that different kinds of objects from $\mathcal{O}$ may have different potential permissions. A folder will have a different potential permission set from that of an ordinary file.

Objects of the system are accessed by the programs executed on behalf of the users. The access rights of a program to various objects in the system are identical with those of the user. Note that we do not distinguish between users and subjects. In the literature the subject (usually associated with a particular execution of some program) can have a (sometimes proper) subset of rights of the user understood as a physical person. Here a physical person can correspond to many users (logins). We denote the set of all users by $\mathcal{U}$. We describe the access rights of users to objects by means of the *access control matrix*.

Definition 6. An *access control matrix* is a map

$$\mathcal{A} : \mathcal{U} \times \mathcal{O} \longrightarrow \bigsqcup \{2^{\mathcal{P}_o} \mid o \in \mathcal{O}\},$$

where by $\sqcup$ we denote a disjoint union of sets, such that

$$\mathcal{A}(u, o) \subseteq \mathcal{P}_o, \quad \text{for all } o \in \mathcal{O}, u \in \mathcal{U}.$$

We read $p \in \mathcal{A}(u, o)$ as "user u has permission p for object o".

In what follows we will assume that users are not objects, and neither is the access control matrix, i.e., $\mathcal{O} \cap \mathcal{U} = \emptyset$ and $\mathcal{A} \notin \mathcal{O}$. This means that our access control matrices will describe the mandatory access control (as opposed to the discretionary one), where all normal users are created by, and all permissions are assigned to by security officers (who are not counted as the users). The users do not own objects and so they cannot pass ownership of "their" objects, nor assign privileges to other users. While rare in the context of operating systems, this is rather typical in the case of corporate databases.

2.4 Roles and role based access control

A *role based access control* (RBAC) is a form of mandatory access policy in which permissions of a user are based on the roles played by the user within organisation ([4]). For example, a person employed by the bank may have the role of a teller or a loan officer. The roles need not be exclusive. A bank employee may pay out money or discuss loans with

customers depending on time and need. Instead of assigning permissions to the user directly, the user is assigned entitities called roles (which would ideally corespond to the particular real world functions performed by the user, like teller or loan officer), carrying with them collections of permissions. A user has the permission p for the object o whenever one of his roles carries permission p for object o.

RBAC clearly simplifies security administration. There are (usually) much fewer roles than objects in the system, and so it is much easier and less error prone to assign a few roles to a new user than it is to construct a row in the access control matrix. Also the role hierarchy should be fully determined by the structure of the organisation, which changes much more slowly than the employee list.

Here we use the modified version of RBAC_1 (e.g. [11, Def. 2]) without sessions. If a person has roles which do not need to (or must not) be executed at the same time (like teller and loan officer in the bank), we will assume that such a person has more than one login, i.e., to the computer system such a person will appear as several users. In this way we do not need to introduce sessions which simplifies the analysis. Another simplification is that we define role hierarchy to be a family of chosen subsets of permissions with the inclusion order, instead of introducing them as abstract entities with some order, and association with permissions determined by many to many relations.

Definition 7. The *role based access control model* consists of:

(1) The sets $\mathcal{U}$ (users), $\mathcal{O}$ (objects), and for each object $o \in \mathcal{O}$ the set $\mathcal{P}_o$ of possible permissions for the object o. We define the set of all permissions

$$\mathcal{P} := \bigcup \{ \{o\} \times \mathcal{P}_o \mid o \in \mathcal{O} \}.$$

(2) The set of roles $\mathcal{R} \subseteq 2^{\mathcal{P}}$ with the inclusion order.

(3) The user to roles assignement $\mathfrak{ur} : \mathcal{U} \longrightarrow 2^{\mathcal{R}}$ satisfying (in order to avoid redundancy) the condition that for all users $u \in \mathcal{U}$ elements of $\mathfrak{ur}(u)$ are incomparable, i.e.,

$$r_1, r_2 \in \mathfrak{ur}(u), r_1 \neq r_2 \quad \Rightarrow \quad r_1 \nsubseteq r_2.$$

The user $u \in \mathcal{U}$ has permissions

$$\bigcup \mathfrak{ur}(u).$$

Clearly the access control matrix can be reconstructed from RBAC as

$$\mathcal{A}(u, o) = \mathrm{pr}_2 \left(\{o\} \times \mathcal{P}_o \cap \bigcup \mathfrak{ur}(u) \right),$$

where pr_2 is the projection on the second component of an ordered pair.

3 Roles and formal concept analysis

Suppose that we are given the access control matrix $\mathcal{A} : \mathcal{U} \times \mathcal{O} \longrightarrow \bigsqcup\{2^{\mathcal{P}_o} \mid o \in \mathcal{O}\}$ of some existing system (for instance database system), where $\mathcal{U}$ is the set of users, $\mathcal{O}$ is the set of objects and $\mathcal{P}_o$ is the set of potential permissions for each object $o \in \mathcal{O}$ (see Def. 6). We want to use the formal concept analysis as a support in generating the role hierarchy in order to simplify the administration of our system. We might also want to judge the correctness and quality of our access control system by finding users who appear to have "too many duties" or to "know too much". Our task is analogous to the problem of modularisation of an old code and judging its quality (cf. [7]). We do not design the role hierarchy from the knowledge of the structure of an organisation, or functionalities which must be implemented by the system, but rather from the existing usage of objects and logins. Because of it, this aproach will work best for the large systems, under the assumption that most of the permissions had been assigned correctly. Large systems are also those which benefit most from introducing roles.

3.1 Role discovery

First we need to define a formal context (Def. 4) for the concept analysis:

Definition 8. Let $\mathcal{P} := \bigcup\{\ \{o\}\times\mathcal{P}_o \mid o \in \mathcal{O}\}$ be the set of permissions, and let $I_{\mathcal{A}} \subseteq \mathcal{U}\times\mathcal{P}$ be a binary relation defined by:

$$I_{\mathcal{A}} := \{(u,(o,p)) \in \mathcal{U} \times \mathcal{P} \mid p \in \mathcal{A}(u,o)\}.$$

We call $(\mathcal{U}, \mathcal{P}, I_{\mathcal{A}})$ a *security context* associated with the access control matrix $\mathcal{A}$. The concept lattice $\mathcal{B}(\mathcal{U}, \mathcal{P}, I_{\mathcal{A}})$ of this context will be called the *lattice of candidate roles* for the security context $(\mathcal{U}, \mathcal{P}, I_{\mathcal{A}})$ or simply a security concept lattice.

Henceforth, for simplicity we shall ignore the internal structure of permissions (each permission is a pair consisting of an object and an access right to this object), which is immaterial for the results presented in the remaining part of the paper. In particular, we will denote permissions by single letters $p \in \mathcal{P}$ instead of writing explicitly $p = (o, q)$.

We interpret the non-empty intents of the concepts of $\mathcal{B}(\mathcal{U}, \mathcal{P}, I_{\mathcal{A}})$ as possible roles. It is justified by the fact that intents of concepts of $\mathcal{B}(\mathcal{U}, \mathcal{P}, I_{\mathcal{A}})$ are the subsets $B \subseteq \mathcal{P}$, closed with respect to $()^{\triangleleft\triangleright}$ (see eq. (3)), i.e., if $p \in \mathcal{P}$ is such that every user who has all permissions from B has also permission p, then also $p \in B$. It is clear that creating a role, which is not closed in this sense, would be a bad design decision leading to unnecessary redundancy. A hidden assumption here, is of course that the set of users is large enough to ensure that any implication of type "all users who have permissions from B also have permission p", inferred from the access control matrix, reflects a real world rule as opposed to being a chance artefact of a small data sample.

In general, it is not recommended to interpret the whole lattice $\mathcal{B}(\mathcal{U}, \mathcal{P}, I_{\mathcal{A}})$ as a role hierarchy. The system designer has to choose a subposet of $\mathcal{B}(\mathcal{U}, \mathcal{P}, I_{\mathcal{A}})$. Unfortunately, we do not believe that such a choice can be fully automated. On the other hand, for realistic systems,

the set of candidate roles should be vastly smaller than the powerset $2^{\mathcal{P}}$, and hence using the concept lattice is really helpful.

Definition 9. We will call any subset $\mathcal{R} \subseteq \{\mathbf{int}(x) \mid x \in \mathcal{B}(\mathcal{U}, \mathcal{P}, I_{\mathcal{A}})\}$ with the inclusion order the *role hierarchy inferred from* $\mathcal{A}$. This role hierarchy defines the role based access control model (Def. 7), where the user to role assignement is given by

$$\mathfrak{ur} : \mathcal{U} \longrightarrow 2^{\mathcal{R}}, \quad u \longmapsto \mathbf{max}_{(\mathcal{R},\subseteq)}\{r \in \mathcal{R} \mid r \subseteq \{u\}^{\triangleright}\}.$$

The role hierarchy $\mathcal{R}$ will be called *complete* if $\bigcup \mathfrak{ur}(u) = \{u\}^{\triangleright}$ for all $u \in \mathcal{U}$.

Note that $\mathcal{R}$ with the opposite inclusion order is a subposet of $\mathcal{B}(\mathcal{U}, \mathcal{P}, I_{\mathcal{A}})$. Note also that if the role hierarchy is not complete (according to the above definition), then not all permissions of users are obtained through roles. Only complete role hierarchy corresponds to the pure role based access model as described in Section 2.4.

The examples below will demonstrate some of the choices which can be made when choosing roles from candidate ones.

Suppose that we have three users $U1$, $U2$, $U3$ and three permissions A, B, C. The context and the corresponding concept lattice is shown in Fig 2.

C1	A	B	C
U1	×		×
U2		×	×
U3	×	×	×

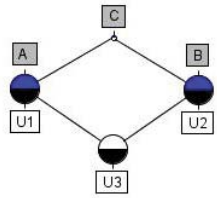

Fig. 2. The context and the corresponding concept lattice.

We read from the concept lattice diagram that there are the following possible roles (non-empty intents of concepts):

$$\{C\}, \quad \{A, C\}, \quad \{B, C\}, \quad \{A, B, C\}.$$

The first note, that if the role hierarchy $\mathcal{R}$ inferred from the above concept lattice is to be complete, it must include roles $\{A, C\}$ and $\{B, C\}$. The role candidates $\{A, B, C\}$ and $\{C\}$ are then optional. We will assume in what follows that $\mathcal{R}$ is complete unless explicitly stated otherwise.

Note, that by Def. 9, even if $\{C\} \in \mathcal{R}$, no user will have $\{C\}$ assigned directly (i.e., not through inheritance) as one of the roles. This situation is possible for roles r equal to one of the intents of non-object concepts. Strictly speaking, in general we have that no user is assigned directly the role $r \in \mathcal{R}$ (i.e., there exists no $u \in \mathcal{U}$ such that $r \in \mathfrak{ur}(u)$) if and only if for any $u \in r^{\triangleleft}$ there exists $r_1 \in \mathcal{R}$ such that

$$r \subsetneq r_1 \subseteq \{u\}^{\triangleright}. \tag{7}$$

If $\{A, B, C\} \in \mathcal{R}$ (resp. $\{A, B, C\} \notin \mathcal{R}$) we have

$$\mathfrak{ur}(U3) = \{\ \{A, B, C\}\ \}, \quad (\text{resp. } \mathfrak{ur}(U3) = \{\ \{A, C\}, \{B, C\}\ \}).$$

In other words we have the choice between creating the compound role through inheritance and assigning this single compound role to the user, or assigning the compounds as separate roles without extending the role hierarchy. Both choices are reasonable as the user may have many roles. Note that we have this choice for the user $u \in \mathcal{U}$ (while requiring completness) precisely when the object concept the user u belongs to is not an attribute concept, i.e., when $\gamma(u) \notin \mu(\mathcal{P})$, or, equivalently, when there does not exist the permission $p \in \mathcal{P}$ such that $\{u\}^{\triangleright} = \{p\}^{\triangleleft\triangleright}$. In the general situation, we can formalize our observation as follows (immediate proof is left to the reader):

Proposition 2. Suppose that $(\mathcal{U}, \mathcal{P}, I_{\mathcal{A}})$ is a *security context*. Any associated complete role hierarchy $\mathcal{R} \subseteq \mathbf{int}(\mathcal{B}(\mathcal{U}, \mathcal{P}, I_{\mathcal{A}}))$ necessarily contains $\gamma(\mathcal{U}) \cap \mu(\mathcal{P})$.

In particular, consider the example in Fig. 3: the user $U3$ is necessarily assigned the role $\{A, B, C\}$ which inherits from the roles $\{A\}$ and $\{B\}$ and in addition, contains a new permission C.

C2	A	B	C
U1	×		
U2		×	
U3	×	×	×

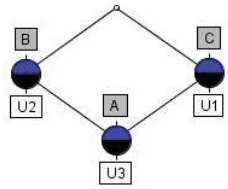

Fig. 3. Example.

The decision whether to include the role hierarchy intents of non-object concepts, or object concepts which are not attribute concepts, is largely left to the discretion of the system designer, as it is hard to formulate some strict rules. Designer's decision might be influenced by the background knowledge about the intended structure of the organisation for which he works, or by the cardinalities of extents and intents relative to the average number of users and permissions per concept. Suppose, for example, that A, B, C, $U1$, $U2$, $U3$ in Fig. 2 are really subsets of permissions and users instead of being single elements. If $U3$ contains a large number of users, it is simpler and less error prone to give each of them the single role $\{A, B, C\}$ instead of assigning the two separate roles. Also note, that the concept lattice helps us to infer the structure of real world facts from the data, and the concepts with large extents have a good chance of corresponding to real roles in the organisation instead of being random and transitory phenomena. On the other hand, if $U3$ contains only a small number of users, it

is possible that the fact that they "do two things" does not reflect any inherent organisational rule (like in the case of a loan officer doing also the teller job because there are now not enough customers asking for loans to fill the full eight hours). Then it is better to give each user in $U3$ the two separate roles instead of artificially extending the role hierarchy.

There is the special choice of the role hierarchy which seems to be the most natural one:

$$\mathcal{R} := \mathbf{int}(\mu(\mathcal{P})) = \{\ \{p\}^{\triangleleft\triangleright} \mid p \in \mathcal{P}\}, \tag{8}$$

i.e., the choice of all the closures of single permissions, or equivalently, all the intents of attribute objects. Note that the set of permissions of the user u is equal to $\bigcup\{\mathbf{int}(\mu(p)) \mid \gamma(u) \leq \mu(p)\}$, hence hierarchy (8) is complete. An advantage of (8) is that one can compute a common set of permissions for a subset of users on the level of roles. Namely, let us denote for any user u

$$\varkappa_{\mathcal{R}}(u) = \{r \in \mathcal{R} \mid \exists_{r' \in \mathfrak{ur}(u)} r \subseteq r'\}.$$

Then, for choice (8) of the role hierarchy, the common set of permissions for the subset of users $U \subseteq \mathcal{U}$ can be written as $\bigcap(\varkappa_{\mathcal{R}}(U))$. The other natural choice of the role hierarchy

$$\mathcal{R} = \{\ \{u\}^{\triangleright} \mid u \in \mathcal{U}\} \tag{9}$$

causes each user to have the single role. This choice amounts to sanctioning the consequent usage of the feature of many database and operating systems which allow to create a new user with the same access rights as those of an existing user.

C3	A	B	C
U1	×	×	
U2		×	×
U3	×		×

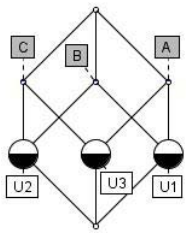

Fig. 4. Security system.

Consider the security system in Fig. 4. The hierarchies of types (9) and (8) are as follows:

- $\{\ \{A, B\}, \{B, C\}, \{A, C\}\ \}$ (role hierarchy (9)),
- $\{\ \{A\}, \{B\}, \{C\}\ \}$ (role hierarchy (8)).

Hence the role hierarchy (8) (unlike (9)) indeed reflects the structure of access control matrix on the level of roles. Note, however, that the example in Fig. 4 is the extreme one, in the sense, that no concept is at the same time the attribute and the object (i.e.,$\gamma(\mathcal{U}) \cap \mu(\mathcal{P}) = \emptyset$). On the other hand, it is well known that if the concept lattice is *meet free*, i.e., the meet of any two incomparable elements equals the bottom element:

C4	HR Zatrud.	Fin	Payroll	Stud Oceny	Stud Styp	HR Ocena
John	×					
Eve	×		×	×		×
Bob			×			
Jane		×	×			×
Joe	×	×			×	
Alec		×			×	
Alice				×		×

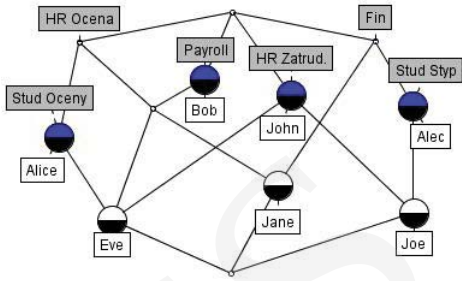

Fig. 5. More complicated example.

$$a \not\leqslant b \text{ and } b \not\leqslant a \text{ implies } a \wedge b = \bot,$$

then each concept different from top and bottom is necessarily an attribute concept. Indeed, if a lattice element $(U, P) \neq \top$ is not a meet of two or more elements, it is covered by the unique element, say (U', P') (i.e., $(U, P) \prec (U', P')$). It follows, that any element of $P \setminus P'$ (which is non-empty because the map **int** (4) is an order reversing isomorphism) must appear first in the concept (U, P), when going from the top. Hence (U, P) is an attribute concept. The role hierarchy based on the object concepts might be much smaller and simpler in the case of the meet free concept lattice.

We conjecture, however, that in practical applications, large security concept lattices will have at least as much non-bottom meets as non-top joins, i.e., if they are asymmetric at all with respect to a number of joins and meets, they will be closer to join free lattices (defined dually to meet free lattices) rather than to meet free. Hence the role hierarchy based on the attribute concepts will tend to be simpler. It is because in a meet free security concept lattice, the users appearing first near the bottom of the lattice (that is the users with more permissions) simply extend the sets of permissions of less privileged users, whereas one of the basic reasons for the user to have more permissions than the other one is that he needs to combine several sources of data. Hence such a user should appear first in the extent of the concept which is the meet of several other concepts. Consequently, we expect many non-bottom meets. On the other hand, if the common set of permissions of two users $u_1, u_2 \in \mathcal{U}$, belonging to incomparable concepts, contains non-public permissions, i.e., if

$$\gamma(u_1) \not\leqslant \gamma(u_2) \text{ and } \gamma(u_2) \not\leqslant \gamma(u_1) \text{ and } \gamma(u_1) \vee \gamma(u_2) \neq \top,$$

this might possibly indicate poor separation of duties.

Consider a more complicated example in Fig. 5. The attribute based on role hierarchy (8) equals

$$\mathcal{R}_1 = \{\ \{\mathsf{HR\ Ocena}\}, \{\mathsf{HR\ Ocena}, \mathsf{Stud\ Oceny}\}, \{\mathsf{Payroll}\},$$
$$\{\mathsf{HR\ Zatrud}\}, \{\mathsf{Fin}\}, \{\mathsf{Fin}, \mathsf{Stud\ Styp}\}\ \},$$

so, e.g., the roles for a user Joe are $\mathfrak{ur}(\mathsf{Joe}) = \{\ \{\mathsf{HR\ Zatrud}\}, \{\mathsf{Fin}, \mathsf{Stud\ Styp}\}\ \}$. Observe that a set $\{\mathsf{Stud\ Styp}\}$ should not be a role because, as evident from the Hasse diagram, each user which has the permission Stud Styp also has the permission Fin.

Note that each user u which is below some other user u' (i.e., $\gamma(u) < \gamma(u')$), first appears in a concept which is a meet of other concepts. It follows that (assuming that each user is given only as many access rights as necessary) each of those users integrates data from many sources. For instance, the user Joe integrates data from Stud Styp and HR Zatrud.

3.2 Finding incorrect roles

The security concept lattice can be used not only to discover roles but also to judge the quality of the security system, e.g., to find users with more access rights than necessary. There are two cases to consider. In the first one the user simply has more permissions than he needs. In the second, the user has just as many access rights as he needs but he does not need them all at the same time. For example, the user might need to access files A and B, but none of the programs executed on behalf of him combines the data from the two files. In the first case, one corrects the access control matrix by deleting the excess permissions. In the second case, one divides the user into several new logins.

A good indication of quality of the security system is that the extent of bottom elements of the security concept lattice is empty, i.e., there are no users which can do everything, and the concept lattice decomposes into many connected components after removing the top and the bottom element (i.e., it has a horizontal decomposition). In other words our security system consists of many independent and isolated subsystems.

Note that the non-empty intent of the top element is not a problem: it is simply interpreted as the set of public access rights. If the top element is an object concept then the users $u \in \mathcal{U}$ such that $\gamma(u) = \top$ are public users.

In practice, the demand for a horizontal decomposition is too strong. One needs to consider the broad (block) structure of the security concept lattice. There exists a strict mathematical definition of blocks and algorithms for recognizing such a broad structure (see e.g. [6]) but the details are beyond the scope of this paper.

Consider for example Fig. 6.
The two blocks on the right (the green and the pink ones) do not form separate horizontal components of the lattice because they are connected by supremum $\mu(A)$, infimum $\gamma(U1)$, and internal bridges through $\mu(B)$ and $\gamma(U2)$.

The supremum $\mu(A)$ and infimum $\gamma(U1)$ are probably legitimate: A is the common permission (or set of permissions) for the two blocks, and the user $U1$ may be integrating data from the two blocks. On the other hand, permission B and the user $U2$ look suspicious, because, if not for them, both blocks would separate neatly. In short, if, after removal of a few users or permissions, our lattice splits into clean blocks which "connect" only as a whole, i.e.,

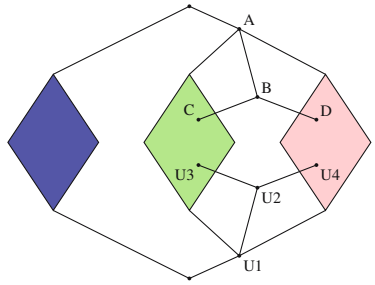

Fig. 6. Example.

through suprema and infima of sets of whole blocks, then those few users and permissions are obvious candidates for checking.

If the connections between blocks are illegitimate or unnecessary, we should correct the access control matrix. The situation is the simplest when, say, the user $U2$ does not really need the permissions of the user $U4$, or the users below $\mu(C)$ do not need permission B. Then we can simply delete unnecessary access rights from the users. If the user $U2$ needs access rights of the users $U3$ and $U4$, but not at the same time, we can split this user into two logins – one with access rights of $U3$, the other with access rights of $U4$. Such splitting removes the bridging concept from the lattice. It is most difficult to deal with bridging supremum-type concepts like $\mu(B)$. In principle, it might require splitting all users below $\mu(C)$ or $\mu(D)$.

Let us consider the security context in Fig. 7 and the corresponding concept lattice in Fig. 8. It splits into two horizontal components. One notices, that all object concepts except one are covered by one or two other concepts. Only γ(P06) is covered by three concepts. Splitting the user P06 (Fig. 9) yields the concept lattice in Fig. 10 which decomposes into four components. Also notice, that $\gamma(P06)$ was not a meet of the three subblocks of the left block in Fig. 8. This indicates, according to the above discussion, that P06's permissions are possibly incorrect.

Q1	HR Main	HR GUS	HR ZUS	KIOD	PY Main	PY ZUS	Podatki	BHP	UNI	BKZ	BWZ	Rekrutacja	Wydawnictwo	KZP
P01					×		×							
P02	×			×										
P03		×	×			×								
P04			×			×								
P05										×	×			
P06		×										×	×	×
P07												×	×	
P08									×				×	
P09										×				
P10									×			×		
P11				×				×						×
P12					×		×			×				

Fig. 7. Security context.

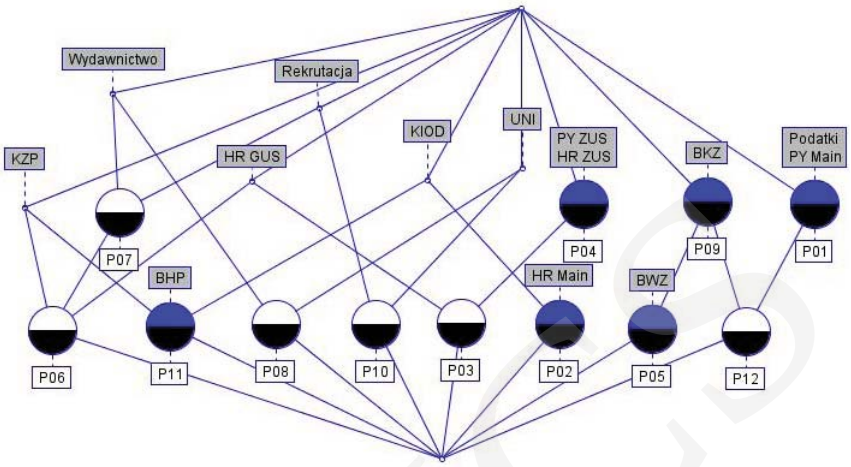

Fig. 8. Concept lattice.

Q2	HR Main	HR GUS	HR ZUS	KIOD	PY Main	PY ZUS	Podatki	BHP	UNI	BKZ	BWZ	Rekrutacja	Wydawnictwo	KZP
P01					×		×							
P02	×			×										
P03		×	×			×								
P04			×			×								
P05										×	×			
P06_1		×												
P07												×	×	
P08									×				×	
P09										×				
P10									×			×		
P11				×				×						×
P12					×		×			×				
P06_2												×	×	
P06_3														×

Fig. 9. Security context.

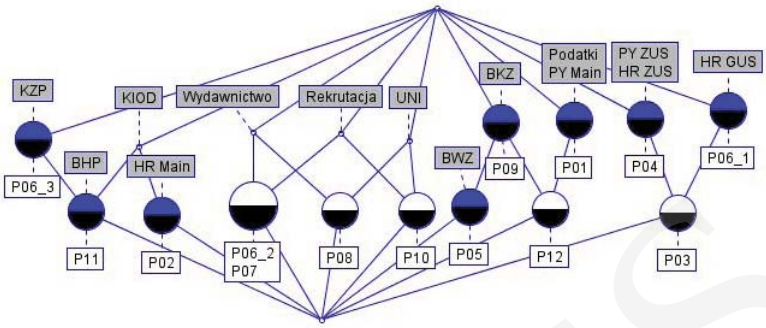

Fig. 10. Concept lattice.

4 Conclusions

We have presented a method of role discovery from the existing permission system using the formal concept analysis, where the possible roles are chosen from the intents of concepts of the concept lattice associated with the access control matrix. The method also allows to judge the quality of security system (whether it splits into well defined and isolated blocks) and to find suspicious logins which might have excess rights. The examples given were rather small, both with respect to the number of permissions and users. On the other hand, this method works best for very large ACL's where inferences of the form: all users which have permission A also have permission B, are less likely to be the result of chance, but rather genuinely reflect the rules of the organisation. We plan in the future to perform a case study of the existing large database system.

In this paper we have not considered the consequences of internal structure of permissions. We conjecture that for a real system, where there are only a few types of objects, and so permissions $\mathcal{P}$ are almost a cartesian product (like $\mathcal{P} = \{\text{ALL FILES}\} \times \{\text{READ}, \text{WRITE}, \text{EXECUTE}\}$), the concept lattice of role candidates may be susceptible to decomposition (e.g [5]), especially the tensorial decomposition [13].

References

[1] Carpineto C., Romano G., Concept Data Analysis. Theory and Applications (John Wiley & Sons, Chichester, 2004): 3–24.

[2] Davey B. A., Priestley H. A., Introduction to Lattices and Order (Cambridge University Press, Cambridge, 1990).

[3] Dai J., Mueller R., Szymański J., Zhang G. Q., Towards "WYDIWYS" for MIMI using concept analysis, Shin S. Y., Ossowski S., Proc. of the 2009 ACM Symposium on Applied Computing (ACM, Honolulu, 2009): 91–97.

[4] Ferraiolo D., Kuhn R., Role-based access controls, 15th NIST-NCSC National Computer Security Conference (Gaithersburg, Md., 1992): 554–563.

[5] Funk P., Lewien A., Snelting G., Algorithms for concept lattice decomposition and their application, Report 95–09 (Computer Science Department, Technische University at Braunschweig, 1995).

[6] Ganter B., Wille R., Formal Concept Analysis. Mathematical Foundations (Springer, Berlin 1999).

[7] Lindig C., Snelting G., Assessing Modular Structure of Legacy Code Based on Mathematical Concept Analysis, Proc. International Conference on Software Engineering (Boston, 1997): 349–359.

[8] Mueller R., Tran V. A., Zhang G. Q., A Scalable Parametric-RBAC Architecture for the Propagation of a Multi-modality, Multi-resource Informatics System, Enterprise Information Systems, Will A. at al. (Springer, Berlin, 2009): 114–124.

[9] Obiedkov S., Kourie D. G., Eloffa J. H. P., Building access control models with attribute exploration, Computers & Security 28(1–2) (2009): 2–7.

[10] Priss U., Formal Concept Analysis in Information Science, Cronin, Blaise, Annual Review of Information Science and Technology 40 (2005).

[11] Sandhu R. S., Coyne E. J., Feinstein H. L., Youman C. E., Role-based access control models, IEEE Computer 29(2) (1996): 38–47.

[12] Wille R., Restructuring lattice theory: an approach based on hierarchies of conceptes, Ordered Sets, Rival I. (1982).

[13] Wille R., Tensorial decomposition of concept lattices, Order 2 (1985): 81–95.

[14] Yevtushenko S. A., System of data analysis "Concept Explorer" (in Russian), Proc. of the 7th National Conference on Artificial Intelligence KII-2000 (Russia, 2000): 127–134.