Leveraging Lattices to Improve Role Mining

Alessandro Colantonio, Roberto Di Pietro, Alberto Ocello

Abstract In this paper we provide a new formal framework applicable to role mining algorithms. This framework is based on a rigorous analysis of identifiable patterns in access permission data. In particular, it is possible to derive a *lattice* of candidate roles from the permission powerset. We formally prove some interesting properties about such lattices. These properties, a contribution on their own, can be applied practically to optimize role mining algorithms. Data redundancies associated with co-occurrences of permissions among users can be easily identified and eliminated, allowing for increased output quality and reduced processing time. To prove the effectiveness of our proposal, we have applied our results to two existing role mining algorithms: Apriori and RBAM. Application of these modified algorithms to a realistic data set consistently reduced running time and, in some cases, also greatly improved output quality; all of which confirmed our analytical findings.

1 Introduction

In recent years *role-based access control* (RBAC, [3]) has been spreading within organizations, greatly due to simplicity of the model: a role is just a set of access permissions, while users are assigned to roles based on duties to fulfill. However, companies still have considerable difficulty migrating to this model due to the com-

Alessandro Colantonio
Engiweb Security, Roma, Italy, e-mail: alessandro.colantonio@eng.it
Università di Roma Tre, Roma, Italy, e-mail: colanton@mat.uniroma3.it

Roberto Di Pietro
Università di Roma Tre, Roma, Italy, e-mail: dipietro@mat.uniroma3.it
Universitat Rovira i Virgili, UNESCO Chair in Data Privacy, Dept. of Computer Engineering and Maths, Av. Països Catalans 26, E-43007 Tarragona, Catalonia, e-mail: roberto.dipietro@urv.cat

Alberto Ocello
Engiweb Security, Roma, Italy, e-mail: alberto.ocello@eng.it

plexity involved in identifying a set of roles fitting the real needs of the company. Thus was born *role engineering*, the discipline of role definition based on actual company needs [5]. Various role engineering approaches proposed in literature are typically classified as: *top-down* or *bottom-up* [7,8,13]. The former carefully deconstructs business processes into elementary components, identifying system features necessary to carry out specific tasks. This activity is mainly manual, requiring a high level analysis of the business [10–12]. The latter class searches legacy access control systems to find *de facto* roles embedded in existing permissions. Automating this process with data mining techniques [4,9,13–16] is called *role mining*. All role mining techniques proposed to date in literature seek to derive candidate roles through the identification of data patterns in currently existing access rights. Despite important differences among the various techniques, almost all take advantage of some common principles summarized by the following:

- *If two access permissions always occur together among users, these should simultaneously belong to the same candidate roles*. Without further access data semantics, a bottom-up approach cannot differentiate between a role made up of two permissions and two roles containing individual permissions [15]. Moreover, defining roles made up of as many permissions as possible minimizes the administration cost by reducing the number of role-user assignments [4].
- *If no user possesses a given combination of access permissions, it makes no sense to define a role containing such combination*. Similar to the previous point, if no user actually performs a task for which a certain permission set is necessary, it is usually better not to define a role containing such an unassignable set.
- *It is quite common within an organization to have many users possessing the same set of access permissions*. This is one of the main justifications that brought about the RBAC model. The creation of a role in connection with a set of co-occurring permissions is typically more advantageous since the number of relationships to be managed is reduced [4].

The following example clarifies the assertions just made, particularly that of the first point presented. If of the given four permissions p_1, p_2, p_3, p_4, the pair p_1, p_2 is always found together with p_3, p_4, it is advisable not to define two distinct roles $\{p_1, p_2\}$ and $\{p_3, p_4\}$ but, rather, a single role $\{p_1, p_2, p_3, p_4\}$. This is different from saying that no user possesses only p_1, p_2 without also having some other permission. Suppose some users possess only p_3, others only p_4, others p_1, p_2, p_3 and still others p_1, p_2, p_4. In this case, even if p_1, p_2 never occur "by themselves", it could be convenient to define the role $\{p_1, p_2\}$ since roles $\{p_3\}$ and $\{p_4\}$ will certainly already exist individually. Thus, avoiding roles $\{p_1, p_2, p_3\}$ and $\{p_1, p_2, p_4\}$.

The cited role mining techniques do not always exploit the above-mentioned observations, even though analyzing such data "recurrences" could improve the quality of proposed candidate roles or increase computational efficiency of the algorithms.

Contributions. The mathematical analysis introduced in this paper provides a new model capable of increasing output quality and reducing process time of role mining algorithms. The model revolves around identifiable patterns in access permissions

data. Through analysis of user permissions, a *lattice* [6] of candidate roles can be constructed from the permission powerset. Notable properties of this lattice will be discussed to substantiate their effectiveness in optimizing role mining algorithms. Leveraging our results, data redundancies associated with co-occurrence of permissions among users can be easily identified and eliminated, thus improving the role mining output.

To prove the merit of our proposal, we have applied our results to two algorithms: Apriori [1] and RBAM [4]. Applying them to a realistic data set yielded drastic reductions in running time and often provided significant redundancy elimination.

Roadmap. This paper is organized as follows: Section 2 cites the main related works. Section 3 reviews mathematical tools and RBAC concepts required for the analysis. Section 4 provides a description of how to define roles based on the permission-powerset lattice, and then Section 5 further analyzes this lattice by introducing the concept of equivalent sublattice and a few of its properties. Section 6 shows how to apply permission-powerset lattice properties to existing role mining techniques. We implement and test, over a real data set, our proposed framework with reference to two role mining algorithms, obtaining support for our theoretical findings. Finally, Section 7 reports concluding remarks and indicates further research directions.

2 Related Work

The proposed mathematical formalism is based on some well-known concepts such as the lattice, powerset, partial order, Hasse diagrams and directed acyclic graphs. The following section introduce these subjects; further details can be found in [6].

Various role mining techniques can benefit from this analysis. Due to space constraints, only a few of them will be summarized. The first improved algorithm is Apriori [1]. It is used in *Market Basket Analysis* (MBA, also known as *association-rule mining*), a method for discovering customer purchasing patterns by extracting associations or recurrences from store transaction databases. Role mining can be seen as a particular application of MBA, simply considering permissions, roles and users instead of products, transactions and customers, respectively. The RBAM algorithm [4] also benefits from the present analysis. It is a specialized implementation of Apriori in which any permission combinations increasing the RBAC model administration cost are rejected. This paper shows how data pruning operations can be conducted to improve RBAM efficiency without detracting from output quality.

There also exist some role mining techniques that take into account some properties described in the previous section. The most important is probably *subset enumeration* [15]. This algorithm starts from permission sets possessed by users and identifies potential candidate roles from all possible intersections among these sets. The resulting candidate role set presents analogies to a lattice of roles where all redundancies related to permission co-occurrence among users are eliminated.

3 Background and Preliminaries

3.1 Posets, Lattices, Hasse Diagrams and Graphs

In computer science and mathematics, a *directed acyclic graph* (DAG) is a directed graph with no directed cycles. For any vertex v, there is no non-empty directed path starting and ending on v, thus DAG "flows" in a single direction. Each DAG provides a *partial order* to its vertices. We write $u \succeq v$ when there exists a directed path from v to u. The *transitive closure* is the reachability order "$\succeq$". A *partially ordered set* (or *poset*) formalizes the concept of element ordering [6]. A poset $\langle S, \succeq \rangle$ consists of a set S and a binary relation "$\succeq$" that indicates, for certain element pairs in the set, which element precedes the other. A partial order differs from a total order in that some pairs of elements may not be comparable. The symbol "$\succeq$" often indicates a *non-strict* (or *reflexive*) partial order. A *strict* (or *irreflexive*) partial order "$\succ$" is a binary relation that is irreflexive and transitive, and therefore asymmetric. If "$\succeq$" is a non-strict partial order, then the corresponding strict partial order "$\succ$" is the reflexive reduction given by: $a \succ b \;\Leftrightarrow\; a \succeq b \wedge a \neq b$. Conversely, if "$\succ$" is a strict partial order, then the corresponding non-strict partial order "$\succeq$" is the reflexive closure given by: $a \succeq b \;\Leftrightarrow\; a \succ b \vee a = b$. An *antichain* of $\langle S, \succeq \rangle$ is a subset $A \subseteq S$ such that $\forall x, y \in A : x \succeq y \;\Rightarrow\; x = y$. We write $x \parallel y$ if $x \not\succeq y \wedge y \not\succeq x$. A *chain* is a subset $C \subseteq S$ such that $\forall x, y \in C : x \succeq y \vee y \succeq x$. Given a poset $\langle S, \succeq \rangle$, the *down-set* of $x \in S$ is $\downarrow x \triangleq \{y \in S \mid x \succeq y\}$, while the *up-set* of $x \in S$ is $\uparrow x \triangleq \{y \in S \mid y \succeq x\}$. Given $a \succeq b$, the interval $[a, b]$ is the set of points x satisfying $a \succeq x \wedge x \succeq b$. Similarly, the interval (a, b) is set of points x satisfying $a \succ x \wedge x \succ b$.

The *transitive reduction* of a binary relation R on a set S is the smallest relation R' on S such that the transitive closure of R' is the same as the transitive closure of R. If the transitive closure of R is antisymmetric and finite, then R' is unique. Given a graph where R is the set of arcs and S the set of vertices, its transitive reduction is referred to as its *minimal representation*. The transitive reduction of a finite acyclic graph is unique and algorithms for finding it have the same time complexity as algorithms for transitive closure [2]. A *Hasse diagram* is a picture of a poset, representing the transitive reduction of the partial order. Each element of S is a vertex. A line from x to y is drawn if $y \succ x$, and there is no z such that $y \succ z \succ x$. In this case, we say y *covers* x, or y is an *immediate successor* of x, also written $y \succcurlyeq x$. A *lattice* is a poset in which every pair of elements has a unique *join* (the least upper bound, or *lub*) and a *meet* (the greatest lower bound, or *glb*). The name "lattice" is suggested by the Hasse diagram depicting it. Given a poset $\langle L, \succeq \rangle$, L is a lattice if $\forall x, y \in L$ the element pair has both a join, denoted by $x \curlyvee y$, and a meet, denoted by $x \curlywedge y$ within L. Let $\langle L, \succeq, \curlyvee, \curlywedge \rangle$ be a lattice. We say that $\langle \Lambda, \succeq, \curlyvee, \curlywedge \rangle : \Lambda \subseteq L$ is a *sublattice* if and only if $\forall x, y \in \Lambda : x \curlyvee y \in \Lambda \;\wedge\; x \curlywedge y \in \Lambda$. In general, we define:

- $\curlyvee \Lambda \triangleq \{x \in L \mid \forall \ell \in L, \forall \lambda \in \Lambda : \ell \succeq \lambda \;\Rightarrow\; \ell \succeq x\}$, the join of Λ (lub);
- $\curlywedge \Lambda \triangleq \{x \in L \mid \forall \ell \in L, \forall \lambda \in \Lambda : \lambda \succeq \ell \;\Rightarrow\; x \succeq \ell\}$, the meet of Λ (glb).

In particular, $x \curlyvee y \triangleq \curlyvee \{x, y\}$ and $x \curlywedge y \triangleq \curlywedge \{x, y\}$. Both $\curlyvee \Lambda$ and $\curlywedge \Lambda$ are unique.

3.2 RBAC Model

We shall now review some of the concepts in the RBAC model according to the ANSI/INCITS standard [3]. The entities of interest for the present analysis are:

- $PERMS$, the set of all possible access permissions; $USERS$, the set of all system users; $ROLES \subseteq 2^{PERMS}$, the set of all roles.
- $UA \subseteq USERS \times ROLES$, the set of user-role assignments. Given a role, the function $ass\_users\colon ROLES \to 2^{USERS}$ identifies all the assigned users.
- $PA \subseteq PERMS \times ROLES$, the set of permission-role assignments. Given a role, the function $ass\_perms\colon ROLES \to 2^{PERMS}$ identifies all the assigned perms.
- $RH \subseteq ROLES \times ROLES$, the set of hierarchical relationships between pairs of roles. $\langle r_1, r_2 \rangle \in RH$ indicates that all the permissions assigned to r_1 are also assigned to r_2, and some more permissions are assigned to r_2.

The symbol "$\succeq$" indicates a partial order based on the role hierarchy. If $r_1 \succeq r_2$, then r_1 is referred to as the *senior* of r_2, while r_2 as the *junior* of r_1. If $r_1 \succcurlyeq r_2$ then r_1 is an *immediate senior* of r_2, while r_2 is an *immediate junior* of r_1. For the sake of simplicity, we define the functions $ass\_users()$ and $ass\_perms()$ indicating respectively the users and permissions *authorized* by a role—what a role inherits along the hierarchical path. This is slightly different from the definition given by the NIST standard: $ass\_users(r)$ thus indicates users possessing permissions assigned to role r instead of users assigned to role r but not to its seniors. In particular:

$$r_1 \succeq r_2 \;\Rightarrow\; ass\_users(r_1) \subseteq ass\_users(r_2) \;\wedge\; ass\_perms(r_1) \supseteq ass\_perms(r_2). \quad (1)$$

In addition to RBAC standard entities, the set $UP \subseteq USERS \times PERMS$ identifies permission to user assignments. In an access control system it is represented by entities describing access rights (e.g., access control lists). Given a permission, the function $perm\_users\colon PERMS \to 2^{USERS}$ identifies the set of users possessing it.

3.3 Support, Confidence and Equivalence

We now review some definitions given in [4]. Since RH defines a partial order on the role set, $\langle ROLES, \succeq \rangle$ is thus a poset on which the following definitions are based.

Definition 1. Given a role $r \in ROLES$, the *support* of that role is defined as $support(r) \triangleq |ass\_users(r)|/|USERS|$ and indicates the percentage of users possessing all permissions assigned to r.

Definition 2. Given $r \in ROLES$, the *degree* of that candidate role is defined as $degree(r) \triangleq |ass\_perms(r)|$ and indicates the number of permissions assigned to r.

Definition 3. Given a pair $r_1, r_2 \in ROLES : r_2 \succeq r_1$, the *confidence* between them is $confidence(r_2 \succeq r_1) \triangleq |ass\_users(r_2)|/|ass\_users(r_1)|$, namely the percentage of users possessing permissions of the junior also possessing permissions of the senior.

Lemma 1. *Given a role pair $r_1, r_2 \in ROLES : r_2 \succeq r_1$, the confidence between such a role pair is* $confidence(r_2 \succeq r_1) = support(r_2)/\,support(r_1)$.

Definition 4. Given a role pair $r_1, r_2 \in ROLES$, we call them *equivalent*, and indicate this with $r_1 \equiv r_2$, if and only if $ass\_users(r_1) = ass\_users(r_2)$.

The following properties are additionally demonstrated:

Lemma 2. *The equivalence relation is transitive, meaning that* $\forall r_1, r_2, r_3 \in ROLES$: $r_1 \equiv r_2 \ \wedge \ r_2 \equiv r_3 \ \Rightarrow \ r_1 \equiv r_3$.

Proof. According to Definition 4, $ass\_users(r_1) = ass\_users(r_2)$ and $ass\_users(r_2) = ass\_users(r_3)$, thus $ass\_users(r_1) = ass\_users(r_3)$. □

Lemma 3. *Given* $r_1, r_2 \in ROLES : r_1 \succeq r_2$, *if* $confidence(r_1 \succeq r_2) = 1$ *then* $r_1 \equiv r_2$.

Proof. From Def. 3, $confidence(r_1 \succeq r_2) = 1 \ \Rightarrow \ |ass\_users(r_1)| = |ass\_users(r_2)|$. From Eq. 1, $ass\_users(r_1) \subseteq ass\_users(r_2) \ \Rightarrow \ ass\_users(r_1) = ass\_users(r_2)$. □

4 Roles Based on the Permission-Powerset Lattice

We now introduce the model on which the following analysis is based. Consider the powerset of a set S (the set of all subsets of S) written as 2^S. The set 2^S can easily be ordered via subset inclusion "$\supseteq$". It can be demonstrated that $\langle 2^S, \supseteq, \cup, \cap \rangle$ is a lattice [6]. Setting $S = PERMS$ makes it possible to build an RBAC model based on all derivable roles from a given permission set. As the operator "$\succeq$" (see Section 3.2) is based on the inclusion operator "$\supseteq$" applied to permissions assigned to roles, it is thus natural to map the operators "$\curlyvee$" to "$\cup$" (the join of two roles represented by the union of all assigned permissions) and "$\curlywedge$" to "$\cap$" (the meet of two roles represented by shared permissions). Every permission combination of the lattice $\langle 2^{PERMS}, \succeq, \curlyvee, \curlywedge \rangle$ identifies the following: (1) an element of *ROLES*, (2) its corresponding relationships in *PA* to such permissions, (3) all permission inclusions in *RH* which involve the role and (4) all relationships in *UA* to users possessing such combination. *RH* is defined to represent the transitive reduction of the graph associated to the lattice. Moreover, if a user is assigned to a role r, then *UA* will contain relationships between r, its juniors and users assigned to them, namely $\forall r \in ROLES, \forall j \in \downarrow r : ass\_users(r) \subseteq ass\_users(j)$.

For simplicity sake, from now on the lattice $\langle 2^{PERMS}, \succeq, \curlyvee, \curlywedge \rangle$ is identified only with the set *ROLES*. The following are some basic properties of this lattice:

Lemma 4. *Removing a role r from ROLES, and its corresponding relationships in PA, UA, RH, such that* $ass\_perms(r) \neq \bigcap_{r' \in ROLES} ass\_perms(r')$ *and* $ass\_perms(r) \neq \bigcup_{r' \in ROLES} ass\_perms(r')$, *the resulting set ROLES is still a lattice.*

Proof. The role r such that $ass\_perms(r) = \bigcap_{r' \in ROLES} ass\_perms(r')$ represents a lower bound for any role pairs, similarly $ass\_perms(r) = \bigcup_{r' \in ROLES} ass\_perms(r')$ represents an upper bound, thus lattice properties are preserved. □

Table 1 An example of set *UP*

User	*Perms*	*User*	*Perms*	*User*	*Perms*	*User*	*Perms*	*User*	*Perms*
u_1	$\{1\}$	u_8	$\{1,3\}$	u_{15}	$\{3,6\}$	u_{22}	$\{1,3,5\}$	u_{29}	$\{2,4,6\}$
u_2	$\{2\}$	u_9	$\{1,4\}$	u_{16}	$\{4,5\}$	u_{23}	$\{1,3,6\}$	u_{30}	$\{1,2,3,5\}$
u_3	$\{3\}$	u_{10}	$\{1,5\}$	u_{17}	$\{4,6\}$	u_{24}	$\{1,4,5\}$	u_{31}	$\{1,2,3,6\}$
u_4	$\{4\}$	u_{11}	$\{1,6\}$	u_{18}	$\{1,2,3\}$	u_{25}	$\{1,4,6\}$	u_{32}	$\{1,2,4,5\}$
u_5	$\{5\}$	u_{12}	$\{2,5\}$	u_{19}	$\{1,2,4\}$	u_{26}	$\{2,3,5\}$	u_{33}	$\{1,2,4,6\}$
u_6	$\{6\}$	u_{13}	$\{2,6\}$	u_{20}	$\{1,2,5\}$	u_{27}	$\{2,3,6\}$	u_{34}	$\{2,3,4,5,6\}$
u_7	$\{1,2\}$	u_{14}	$\{3,5\}$	u_{21}	$\{1,2,6\}$	u_{28}	$\{2,4,5\}$	u_{35}	$\{1,2,3,4,5,6\}$

Note 1. Given $r \in ROLES$ then $\forall s \in \uparrow r : support(r) \geq support(s)$. In fact, users possessing permission combination $ass\_perms(r)$ do not necessarily possess other permissions. Analogously, $\forall j \in \downarrow r : support(r) \leq support(j)$. Apriori [1] and RBAM [4] algorithms use this property as a pruning condition to limit the solution space.

Based on the initial hypothesis of Section 1, *roles to which unused permission combinations are assigned do not represent significant candidate roles*. Such roles have support equal to 0 and can be eliminated from *ROLES*, except for the meet and join which are required to preserve lattice properties (see Lemma 4). Removing such roles results in a lattice that satisfies the following property:

Lemma 5. *The immediate seniors of a role $r \in ROLES$ differ from r by a single permission, that is* $\forall r,s \in ROLES : s \gg r \;\Rightarrow\; degree(s) = degree(r) + 1$.

Proof. For Equation 1, any role represented by a subset of $ass\_perms(s)$ has support > 0 and is at least assigned to users $ass\_users(s)$. Thus, *ROLES* contains all roles obtained by removing a single permission from $ass\_perms(s)$, including r. □

5 Equivalent Sublattices

Let *ROLES* be the lattice based on 2^{PERMS} in which roles with support equal to 0 have been eliminated, except for the meet and join. Such set has a very simple property: *every candidate role set is contained within*, since it provides all user-assignable permission combinations. Beyond eliminating roles having support equal to 0, this section shows that *it is also possible to remove roles presenting equivalence with other roles*, as they do not belong to any "reasonable" candidate role set.

Table 1 shows an example of *UP* presenting equivalence relationships. By observing the data, it can be noted that all users simultaneously possessing permissions 3 and 4 also always have permissions 2, 5 and 6. Figure 1 shows the role lattice built on the given set *UP* with junior roles above and senior roles below. Despite this being a directed graph, direction indicators are absent (from top to bottom) to avoid complicating the figure. Thicker lines represent hierarchical relationships with confidence equal to 1, namely equivalence relationships (see Lemma 3).

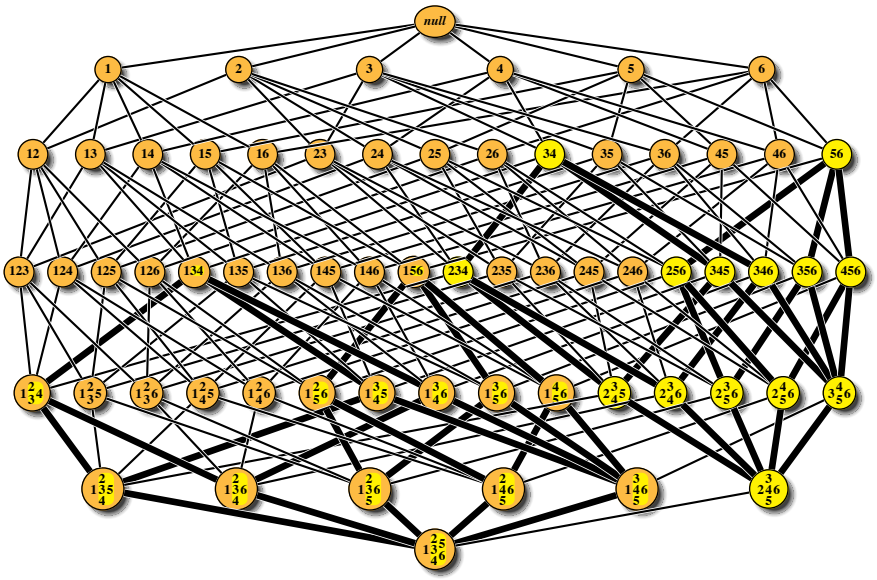

Fig. 1 Hasse diagram of the lattice based on permission powerset derived from Table 1

Next, we want to demonstrate that *when a role has more equivalent seniors, the combination of its assigned permissions still represents an equivalent role*. For example, $\{3,4\} \equiv \{2,3,4\}$, $\{3,4\} \equiv \{3,4,5\}$ and $\{3,4\} \equiv \{3,4,6\}$ implies $\{3,4\} \equiv \{2,3,4,5,6\}$. Moreover, the set of equivalent seniors forms a sublattice. We will now formalize this with a series of theorems demonstrating that: (1) given an interval of roles, if the bounds are equivalent then all roles on the interval are equivalent with each other; (2) by analyzing immediate equivalent seniors, the equivalent role with the maximum degree can be determined; (3) an interval of equivalent roles having the equivalent role with the maximum degree as upper bound is a sublattice of *ROLES*; (4) such sublattice is replicated in *ROLES* with the same "structure".

Theorem 1. *Given a role pair $r_1, r_2 \in ROLES$ such that $r_2 \succeq r_1$ and $r_1 \equiv r_2$, then all roles on the interval $[r_1, r_2]$ are equivalent to each other:*

$$\forall r, r_1, r_2 \in ROLES : r_2 \succeq r \succeq r_1 \wedge r_1 \equiv r_2 \quad \Rightarrow \quad r \equiv r_1 \equiv r_2.$$

Proof. According to Equation 1, $ass\_users(r_2) \subseteq ass\_users(r) \subseteq ass\_users(r_1)$. But $ass\_users(r_1) = ass\_users(r_2)$, so $ass\_users(r_2) = ass\_users(r) = ass\_users(r_1)$. □

Theorem 2. *A role $r \in ROLES$ is equivalent to the role represented by the union of permissions assigned to any set of its equivalent seniors:*

$$\forall r \in ROLES, \forall R \subseteq \uparrow r, \forall r' \in R : r' \equiv r \quad \Rightarrow$$
$$\Rightarrow \quad \exists s \in ROLES : r \equiv s \wedge ass\_perms(s) = \bigcup_{r' \in R} ass\_perms(r').$$

Proof. Users possessing a role are those possessing all the permissions assigned to that role, namely $\forall r' \in ROLES : ass\_users(r') = \bigcap_{p \in ass\_perms(r')} perm\_users(p)$. According to the hypothesis, $\forall r_i \in R : r_i \equiv r$, so all roles in R are assigned with the same users. Then $\bigcap_{r' \in R} \left(\bigcap_{p \in ass\_perms(r')} perm\_users(p) \right) = ass\_users(r)$. Such an equality can also be written as $\bigcap_{p \in \bigcup_{r' \in R} ass\_perms(r')} perm\_users(p) = ass\_users(r)$ but $\bigcup_{r' \in R} ass\_perms(r')$ represent the set of permissions assigned to the role s. □

Definition 5. Given $r \in ROLES$, the *maximum equivalent role* of r, written $\bar{r}$, is the role represented by the union of permissions of its immediate equivalent seniors:

$$ass\_perms(\bar{r}) = \bigcup_{r' \in ROLES \mid r' \gg r \wedge r' \equiv r} ass\_perms(r').$$

The name attributed to the role $\bar{r}$ is justified by the following theorem:

Theorem 3. *Given $r \in ROLES$, $\bar{r}$ is the equivalent role with the highest degree:*

$$\forall r' \in ROLES : r' \equiv r \wedge r' \neq \bar{r} \quad \Rightarrow \quad degree(r') < degree(\bar{r}).$$

Proof. Seeking a contradiction, suppose that $r_{\max} \in ROLES : r_{\max} \neq \bar{r}$ is the highest degree role among all those equivalent to r. Since the same users possess both $\bar{r}$ and $r_{\max}$, then $ass\_perms(\bar{r}) \subseteq ass\_perms(r_{\max})$. If this was not the case, then there would exist another role within $ROLES$ made up of the union of permissions assigned to $\bar{r}$ and $r_{\max}$ having a larger degree than both of these. This other role would also be equivalent to $\bar{r}$ and $r_{\max}$, since it is possessed by the same users. However, this contradicts the fact that $r_{\max}$ is of the highest degree.

Let $\Delta = ass\_perms(r_{\max}) \setminus ass\_perms(\bar{r})$. If $\Delta \neq \emptyset$, then it is possible to identify "intermediate" roles $\rho \in [r, r_{\max}]$ such that $\exists p \in \Delta : ass\_perms(\rho) = ass\_perms(r) \cup \{p\}$. For Lemma 5, $\rho \gg r$, while for Theorem 1, $\rho \equiv r$. Since $\bar{r}$ is obtained by the union of all permissions assigned to all equivalent immediate seniors, it contains all the permissions of Δ. Consequently, it must be that $\Delta = \emptyset$ and so $\bar{r} = r_{\max}$. □

Theorem 4. *Given $r, s \in ROLES : s \succeq r$, the interval $[r,s]$ is a sublattice of $ROLES$.*

Proof. As long as $[r,s]$ is a lattice, it must be true that $\forall r_1, r_2 \in [r,s] : r_1 \curlyvee r_2 \in [r,s] \wedge r_1 \curlywedge r_2 \in [r,s]$. Given $r_1, r_2 \in [r,s]$, let r_{ub} be an upper-bound role such that $ass\_perms(r_{\text{ub}}) = ass\_perms(r_2) \cup ass\_perms(r_2)$. Since $s \succeq r_1, r_2$ then the permissions of s include the union of the permissions of r_1, r_2, so $s \succeq r_{\text{ub}}$. Thus, $r_{\text{ub}} \in [r,s]$. Similarly, it can be demonstrated that $[r,s]$ contains a lower-bound role r_{lb} such that $ass\_perms(r_{\text{lb}}) = ass\_perms(r_2) \cap ass\_perms(r_2)$.

Definition 6. Given a role $r \in ROLES$, we define the *equivalent sublattice* of r, indicated by $\varepsilon(r)$, the interval $[r, \bar{r}]$, that is $\varepsilon(r) \triangleq [r, \bar{r}]$.

Note 2. The set $\varepsilon(r)$ *does not* represent all the equivalent roles of r, rather, only a subset. In fact, we could have $r' \in ROLES$ such that $r \equiv r'$ even though $r \parallel r'$. However, for Theorem 3, from the union of permissions assigned to immediate equivalent seniors of r or r', the same maximum equivalent role is obtained, that is $\bar{r} \equiv \bar{r}'$. In fact, in Figure 1, roles $\{3,4\}$ and $\{5,6\}$ are antichain but, being equivalent to each other, they share the same maximum equivalent role $\{2,3,4,5,6\}$.

Note 3. If a role has equivalent seniors, then no user possesses only its permissions, namely if $\exists r' \in (\uparrow r) \setminus r : r \equiv r'$ then $ass\_users(r) \setminus \bigcup_{\rho \in (\uparrow r) \setminus r} ass\_users(\rho) = \emptyset$. The converse is not true. Particularly, if there is no user possessing a given permission combination, it is unknown whether the role made up of such permissions has immediate equivalent seniors. This is verified in Table 1. Permissions 3 and 4 are always found together with 2, 5 and 6. Thus, no user is assigned to role $\{3,4\}$ unless also assigned to one of its seniors. Yet, the contrary is not true: even though $\{2,3\}$ has no immediate equivalent seniors, it is not assigned with any user.

Theorem 5. *Given a role $r \in ROLES$, let $E = \{r' \in ROLES \mid r' \succcurlyeq r \wedge r' \equiv r\}$ be the set of immediate equivalent seniors of r. Then $|\varepsilon(r)| = 2^{|E|}$.*

Proof. For Lemma 5, $\forall r' \in E : degree(r') = degree(r) + 1$. Thus, permissions assigned to the maximum equivalent role of r include those of r plus a number of other permissions equal to $|E|$, that is $degree(\bar{r}) = degree(r) + |E|$. Further, $\varepsilon(r)$ contains all roles whose permission combinations are between $ass\_perms(r)$ and $ass\_perms(\bar{r})$, all of which have support greater than 0. Hence, the possible permission combinations between $ass\_perms(r)$ and $ass\_perms(\bar{r})$ are $2^{|E|}$.

Theorem 6. *Let there be $r,s \in ROLES$ such that s is an immediate equivalent senior of r. If there is $s' \in ROLES$, an immediate non-equivalent senior or r, then certainly there is a role $s'' \in ROLES$, an immediate equivalent senior of s' and immediate senior of s, represented by the union of permissions of s, s':*

$$\forall r,s,s' \in ROLES \;:\; s \succcurlyeq r \wedge s' \succcurlyeq r \wedge s \equiv r \wedge s' \not\equiv r \quad \Rightarrow \quad \exists s'' \in ROLES \;:\; s'' \succcurlyeq s \wedge s'' \succcurlyeq s' \wedge s' \equiv s'' \wedge ass\_perms(s'') = ass\_perms(s) \cup ass\_perms(s').$$

Proof. The role s'' is a senior of both s, s' since $ass\_perms(s'') \supseteq ass\_perms(s)$ and $ass\_perms(s'') \supseteq ass\_perms(s')$. But $r \equiv s$, so $ass\_users(s'') = ass\_users(s) \cap ass\_users(s') = ass\_users(r) \cap ass\_users(s')$. But $ass\_users(s') \subseteq ass\_users(r)$ because of $s' \succcurlyeq r$, then $ass\_users(s'') = ass\_users(s')$. Finally, for Lemma 5 roles s and s' have an additional permission to that of r. If $s \neq s'$ then $degree(s'') = degree(r) + 2$. Hence, s'' is an immediate senior to both s, s'. □

Note 4. The previous theorem can be observed in Figure 1. The role $\{3,4\}$ has three immediate equivalent senior roles, while $\{1,3,4\}$ represents an immediate non-equivalent senior. For Theorem 6, this means that $\{1,3,4\}$ has *at least* three immediate equivalent seniors, identifiable by adding the permission 1 to equivalent seniors of $\{3,4\}$; according to Theorem 6, further immediate equivalent seniors of $\{1,3,4\}$ are allowed.

Theorem 7. *Let there be $r,s \in ROLES$ such that $s \succcurlyeq r$ and $s \not\equiv r$. Let also $p = ass\_perms(s) \setminus ass\_perms(r)$. Then there is a replica of the sublattice $\varepsilon(r)$ obtained by adding permission p to those of $\varepsilon(r)$.*

Proof. For Theorem 6, role s has among its immediate equivalent seniors at least those obtainable by adding permission p to immediate equivalent seniors of r. Let

then $s' \in ROLES$ be the senior of s represented by the union of such immediate equivalent seniors, meaning $ass\_perms(s') = ass\_perms(\bar{r}) \cup \{p\}$. According to Theorem 2, $s \equiv s'$, while for Theorem 4 the interval $[s, s']$ is a sublattice. Let σ be a role defined from role $\rho \in \varepsilon(r)$ such that $ass\_perms(\sigma) = ass\_perms(\rho) \cup \{p\}$. Then, $s' \succeq \sigma$ since $ass\_perms(\bar{r}) \cup \{p\} \supseteq ass\_perms(\rho) \cup \{p\}$ and $\sigma \succeq s$ because $ass\_perms(\rho) \cup \{p\} \supseteq ass\_perms(r) \cup \{p\}$. Hence, $\sigma \in [s, s']$. □

A direct consequence of the preceding theorem can be seen in Figure 1. The equivalent sublattice $\varepsilon(\{1,3,4\})$ can be obtained from $\varepsilon(\{3,4\})$ by adding the permission 1 to all roles. In the Hasse diagram of *ROLES* it is therefore possible to identify a certain number of equivalent sublattice replicas determined by:

Theorem 8. *Given a role $r \in ROLES$ let S be the set of immediate non-equivalent seniors, $S = \{\rho \in ROLES \mid \rho \succcurlyeq r \ \wedge \ \rho \not\equiv r\}$. Then ROLES has a number of $\varepsilon(r)$ replicas between $|S|$ and $2^{|S|} - 1$.*

Proof. For Theorem 7, for all roles $s \in S$ the sublattice $\varepsilon(r)$ is replicated by adding permission $ass\_perms(s) \setminus ass\_perms(r)$ to every role in $\varepsilon(r)$. So, there are at least $|S|$ sublattice replicas. Starting from S, the set $P = \bigcup_{s \in S} ass\_perms(s) \setminus ass\_perms(r)$ of permissions added to r from non-equivalent seniors of r can be identified. For Lemma 5, the difference of degree between r and $s \in S$ is equal to 1, thus $|P| = |S|$. Every role $s \in S$ has at most $|S| - 1$ immediate non-equivalent seniors, meaning those represented by $ass\_perms(s)$ to which are added one of the permissions of $P \setminus (ass\_perms(s) \setminus ass\_perms(r))$. If, by contradiction, there was a role s', an immediate non-equivalent senior of s, for which $p = ass\_perms(s') \setminus ass\_perms(s) \ \wedge \ p \notin P$, then a role r' such that $ass\_perms(r') = ass\_perms(r) \cup \{p\}$ would have a support greater than 0 and would belong to S. This means that, still for Theorem 7, the role s can produce, at most, another $|S| - 1$ replicas. Reiterating the same reasoning for all seniors of r, it can be deduced that at most $2^{|S|} - 1$ replicas can be constructed by roles of $\varepsilon(r)$ to which are added permission combinations of $2^P \setminus \{\emptyset\}$. □

6 Discussion and Applications

The previous section analyzed some properties of a role lattice based on the powerset of permissions excluding combinations of support equal to 0. It was shown that a certain number of equivalent sublattice replicas could exist within such lattice. Based on the premises of Section 1, *all these replicas can be eliminated from the set of candidate roles except for maximum equivalent roles*. In fact, a maximum equivalent role can be considered a "representative" of all sublattices to which it belongs. Removing equivalent sublattices prunes the candidate role set solution space. Given a role $r \in ROLES$, let $E = \{r' \in ROLES \mid r' \succcurlyeq r \ \wedge \ r' \equiv r\}$ be the set of immediate equivalent seniors and $S = \{r' \in ROLES \mid r' \succcurlyeq r \ \wedge \ r' \not\equiv r\}$ be the set of immediate *non*-equivalent seniors. For Theorem 5, the equivalent sublattice generated by r contains $|\varepsilon(r)| = 2^{|E|}$ roles, all of which can be eliminated from *ROLES* except for $\bar{r}$.

Algorithm 1 Procedure Remove-Equivalent-Sublattices

Require: R_k, H_k, PA, UA, k
Ensure: R_k, H_k, PA, UA, M_i
1: $W \leftarrow \emptyset$ ▷ Set of equivalent roles to be deleted
2: $M_i \leftarrow \emptyset$ ▷ Set of maximum equivalent roles
3: **for all** $\rho \in \{h.junior \mid h \in H_k : h.confidence = 1\}$ **do**
4: ▷ Identify equivalences in R_k to be deleted and maximum equivalent role permissions
5: $E \leftarrow \{h.senior \mid h \in H_k : h.junior = \rho \wedge h.confidence = 1\}$ ▷ Equivalent seniors
6: $S \leftarrow \{h.senior \mid h \in H_k : h.junior = \rho \wedge h.confidence < 1\}$ ▷ Non-equivalent seniors
7: $P \leftarrow (\bigcup_{r \in E} ass\_perms(r)) \setminus ass\_perms(\rho)$ ▷ Perms diff between maximum equiv role
8: $W \leftarrow W \cup E$ ▷ Mark equivalent immediate seniors for deletion
9: ▷ Transform ρ into its maximum equivalent role. Enrich roles in S with permissions P.
10: **for all** $\sigma \in S \cup \{\rho\}$ **do**
11: $\sigma.degree \leftarrow \sigma.degree + |P|, \quad PA \leftarrow PA \cup (P \times \{\sigma\}), \quad M_i \leftarrow M_i \cup \{\sigma\}$
12: **end for**
13: **end for**
14: ▷ Delete equivalent roles in R_k
15: $R_k \leftarrow R_k \setminus W, \quad PA \leftarrow \{\langle p, r\rangle \in PA \mid r \notin W\}, \quad UA \leftarrow \{\langle u, r\rangle \in UA \mid r \notin W\}$
16: $H_k \leftarrow \{h \in H_k \mid h.senior \notin W\}$

Based on the theorems of the preceding section, $\varepsilon(r)$ and $\bar{r}$ can be derived from r and E. Prospective algorithms calculating roles based on the permission-powerset lattice could benefit from eliminating equivalent sublattices if $2^{|E|} > |E| + 1$, namely when the cost of calculating $\varepsilon(r)$ is greater than the cost of calculating only the roles necessary for identifying $\bar{r}$. For simplicity, operating costs necessary for constructing role $\bar{r}$ from r and E are deemed negligible. The inequality $2^{|E|} > |E| + 1$ is always true when $|E| > 1$, namely when role r has more than one equivalent junior. For Theorem 8, every equivalent sublattice has at least $|S|$ number of replicas derivable from r, E, S. It is thus advantageous to remove these when $(|S| + 1)2^{|E|} > |E| + |S| + 1$, that is true when $|E| > 1$, where $(|S| + 1)2^{|E|}$ represent the amount pruned.

6.1 Equivalent Sublattice Deletion in Apriori

This section introduces the RB-Apriori (*Role-Based Apriori*) algorithm to identify roles based on permission-powerset lattices with no equivalent sublattices. Using the Apriori [1] algorithm makes it possible to generate a partial lattice by pruning permission combinations whose support is lower than a pre-established threshold $s_{\min}$ [4]. RB-Apriori extends Apriori removing equivalent sublattices except for the maximum equivalent roles. The following are the main steps of Apriori summarized. The set $R_k \subseteq ROLES$ denotes all roles calculated at step k of the algorithm, while $H_k \subseteq RH$ gathers the immediate hierarchical relations among roles in R_i and R_{i-1}.

Step 1 An initial analysis of UP provides the set R_1 containing candidate roles of degree 1 with a support greater than the minimum.

Step k When $k \geq 2$, the set R_k is generated merging all possible role pairs in R_{k-1} (*join step*). In order not to generate roles with the same permission set, only role pairs differing in the greater permission are considered. Combinations not meeting minimum support constraints are rejected (*prune step*). Hierarchical associations (H_k) are also identified, relating roles in R_k whose assigned permissions are a superset of permissions of roles in R_{k-1}.

Stop The algorithm completes when $R_k = \emptyset$, returning *ROLES* as the union of all calculated R_i and *RH* as the union of all calculated H_i.

RB-Apriori is obtained from Apriori by calling the Remove-Equivalent-Sublattices procedure at the end of every step k. The procedure is described in Algorithm 1. Given $r \in ROLES$, $r.degree$ indicates the number of permissions assigned to it; given $h \in RH$, $h.junior$ and $h.senior$ indicate the pair of roles hierarchically related, while $h.confidence$ is the confidence value between them. Step 3 of Algorithm 1 identifies all roles calculated in step $k-1$ presenting immediate equivalent seniors in R_k. For each of these roles, the steps immediately following determine sets E, S and the permission set P to be added to the role in order to obtain the maximum equivalent role. Steps 10–12 make up the maximum equivalent role by adding permissions P to the current role. The immediate non-equivalent seniors are also enriched with the same permissions; if not, eliminating roles E (Steps 8, 15–16) could prevent identification of the combination of permissions assigned to those roles during step $k+1$. Based on the Note 4, enriching permissions assigned to immediate non-equivalent seniors with P it is not definite that the respective maximum equivalent roles will be generated. This means that RB-Apriori *prunes only one sublattice at a time*, without also simultaneously eliminating any replicas.

As described in Note 2, there could exist $r_1, r_2 \in ROLES : r_1 \equiv r_2 \wedge r_1 \parallel r_2$. In Figure 1, roles $\{3,4\}$ and $\{5,6\}$ are equivalent and share the same maximum equivalent role $\{2,3,4,5,6\}$. According to Algorithm 1, the role $\{2,3,4,5,6\}$ is built twice. This means that after the last step ("Stop") of RB-Apriori it is necessary to check for duplicate roles. Particularly, given the set $M = \bigcup M_i$ of identified maximum equivalent roles, for every $m \in M$ each $r \in ROLES \setminus \{m\} : ass\_perms(r) \subseteq ass\_perms(m) \wedge support(r) = support(m)$ needs to be discarded.

6.2 Testing on Real Data

To assess the efficiency of the RB-Apriori algorithm described in the previous section, many tests have been conducted using real data. In order to highlight the properties of the algorithm, consider the results obtained from analyzing data of an application with a heterogeneous distribution of user permissions. In the analyzed data set, 954 users were possessing 1,108 different permissions. By applying the Apriori algorithm with $s_{\min} = 10\%$, a total of 299 roles were generated in about 119 seconds through the adopted Apriori implementation. These 299 roles were assigned with only 16 of the available 1,108 permissions resulting in 890 users possessing these permissions. Using the same minimum support, with RB-Apriori we obtained only

109 roles in 87 seconds, thus reducing the number of roles by 64% and the computation time by 27%. The difference in improvement between role number and computation time was due to time "wasted" in identifying equivalent sublattices. Actually, the algorithm identified 167 roles; although 58 of the 167 were subsequently eliminated as equivalents, time was saved avoiding computation of entire equivalent sublattices. Changing the minimum support to $s_{\min} = 5\%$, 8,979 roles were produced with Apriori in about 3,324 seconds, involving 31 permissions and 897 users. With RB-Apriori we obtained only 235 roles in 349 seconds, thus reducing the number of roles by 97% and computation time by 90%.

6.3 Comparison to the RBAM Algorithm

The RBAM [4] algorithm leverages the RBAC administration cost estimate to find the lowest cost candidate role-sets, implementing an extended version of Apriori to identify the optimal role set. Pruning operations are based on the variable minimum support concept. According to [4], a role $r \in ROLES$ can be removed when the percentage of users assigned to r but none of its seniors is below a threshold related to the administration cost of r. When r has equivalent seniors, this percentage is equal to 0 because of Note 3. Thus, RBAM always removes its equivalent sublattice. Since RBAM is an extended version of Apriori, it is easy to improve performances of the RBAM algorithm, basing it on RB-Apriori instead of Apriori. While producing the same candidate role sets, computation of the entire equivalent sublattices is avoided, thus improving the efficiency and obtaining performance comparable to RB-Apriori.

7 Conclusions and Future Work

This paper introduces a new formal framework based on a rigorous pattern analysis in access permissions data. In particular, it is possible to derive a *lattice* of candidate roles from the permission powerset. We have proved some interesting properties about the above-defined lattice useful for optimizing role mining algorithms. By leveraging our results, data redundancies associated with co-occurrence of permissions among users can be easily identified and eliminated, hence increasing output quality and reducing process time of data mining algorithms.

To prove the effectiveness of our proposal, we have applied our results to two role mining algorithms: Apriori and RBAM. Applying these modified algorithms to a realistic data set, we drastically reduced the running time, while the output quality was either unaffected or even improved. Thus, we confirmed our analytical findings.

As for future work, we are currently pursuing two activities: the first is to apply our findings to other role mining algorithms; the second is investigating equivalence relationships between a single role and a set of roles.

Acknowledgement

Roberto Di Pietro was partly supported by the Spanish Ministry of Science and Education through projects TSI2007-65406-C03-01 E-AEGIS and CONSOLIDER CSD2007-00004 ARES, and by the Government of Catalonia under grant 2005 SGR 00446. The authors would also like to thank Richard A. Parisi Jr. for his helpful comments and valuable review.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: J.B. Bocca, M. Jarke, C. Zaniolo (eds.) Proceedings of the 20th International Conference on Very Large Data Bases, VLDB, pp. 487–499. Morgan Kaufmann (1994)
2. Aho, A.V., Garey, M.R., Ullman, J.D.: The transitive reduction of a directed graph. SIAM Journal on Computing **1**(2), 131–137 (1972)
3. ANSI/INCITS 359-2004, Information Technology – Role Based Access Control (2004)
4. Colantonio, A., Di Pietro, R., Ocello, A.: A cost-driven approach to role engineering. In: Proceedings of the 23rd ACM Symposium on Applied Computing, SAC '08, pp. 2129–2136. Fortaleza, Ceará, Brazil (2008)
5. Coyne, E.J.: Role-engineering. In: Proceedings of the 1st ACM Workshop on Role-Based Access Control, RBAC '95 (1995)
6. Davey, B.A., Priestley, H.A.: Introduction to Lattices and Order, 2 edn. Cambridge University Press (2002)
7. Epstein, P., Sandhu, R.: Engineering of role/permission assignments. In: Proceedings of the 17th Annual Computer Security Applications Conference, ACSAC, pp. 127–136. IEEE Computer Society (2001)
8. Kern, A., Kuhlmann, M., Schaad, A., Moffett, J.: Observations on the role life-cycle in the context of enterprise security management. In: Proceedings of the 7th ACM Symposium on Access Control Models and Technologies, SACMAT '02 (2002)
9. Kuhlmann, M., Shohat, D., Schimpf, G.: Role mining - revealing business roles for security administration using data mining technology. In: Proceedings of the 8th ACM Symposium on Access Control Models and Technologies, SACMAT '03, pp. 179–186 (2003)
10. Neumann, G., Strembeck, M.: A scenario-driven role engineering process for functional RBAC roles. In: Proceedings of the 7th ACM Symposium on Access Control Models and Technologies, SACMAT '02 (2002)
11. Röckle, H.: Role-finding/role-engineering. In: Proceedings of the 5th ACM Workshop on Role-Based Access Control, RBAC 2000, p. 68 (2000)
12. Röckle, H., Schimpf, G., Weidinger, R.: Process-oriented approach for role-finding to implement role-based security administration in a large industrial organization. In: Proceedings of the 5th ACM Workshop on Role-Based Access Control, RBAC 2000, vol. 3 (2000)
13. Schlegelmilch, J., Steffens, U.: Role mining with ORCA. In: Proceedings of the 10th ACM Symposium on Access Control Models and Technologies, SACMAT '05, pp. 168–176 (2005)
14. Vaidya, J., Atluri, V., Guo, Q.: The role mining problem: finding a minimal descriptive set of roles. In: Proceedings of the 12th ACM Symposium on Access Control Models and Technologies, SACMAT '07, pp. 175–184 (2007)
15. Vaidya, J., Atluri, V., Warner, J.: RoleMiner: mining roles using subset enumeration. In: Proceedings of the 13th ACM Conference on Computer and Communications Security (2006)
16. Zhang, D., Ramamohanarao, K., Ebringer, T.: Role engineering using graph optimisation. In: Proceedings of the 12th ACM Symposium on Access Control Models and Technologies, SACMAT '07, pp. 139–144 (2007)