Sorting Out Role Based Access Control

Wouter Kuijper
Nedap N.V.
Parallelweg 2, Groenlo
the Netherlands, 7141 DC
wouter.kuijper@nedap.com

Victor Ermolaev
Nedap N.V.
Parallelweg 2, Groenlo
the Netherlands, 7141 DC
victor.ermolaev@nedap.com

ABSTRACT

Role-based access control (RBAC) is a popular framework for modelling access control rules. In this paper we identify a fragment of RBAC called *bi-sorted role based access control* (RBÄC). We start from the observation that "classic" RBAC blends together subject management aspects and permission management aspects into a single object of indirection: a *role*. We posit there is merit in distinguishing these administrative perspectives and consequently introducing two distinct objects of indirection: the *proper role* (which applies solely to subjects) and the *demarcation* (which applies solely to permissions). We then identify a third administrative perspective called *access management* where the two are linked up. In this way we enhance organisational scalability by decoupling the tasks of maintaining abstractions over the set of subjects (assignment of subjects into proper roles), maintaining abstractions over the set of permissions (assignment of permissions into demarcations), and maintaining abstract access control policy (granting proper roles access to demarcations). Moreover, the latter conceptual refinement naturally leads us to the introduction of *negative roles* (and, dually, *negative demarcations*). The relevance of the four-sorted extension called *polarized, bi-sorted role based access control* ($R^{\pm}$BÄC), in a semantic sense, is further supported by the existence of Galois connections between sets of subjects and permissions and between positive and negative roles.

Categories and Subject Descriptors

H.1 [**Models and principles**]: General—*RBAC*; H.4 [**Information Systems Applications**]: Miscellaneous—*physical access control*

General Terms

Mathematical foundations of RBAC, Galois connection

Keywords

RBAC; Organizational Structure; Physical Access Control; Scalability; Positive Specification; Negative Specification; Galois Connections; Domain Specific Languages

SACMAT'14, June 25–27, 2014, London, Ontario, Canada.

ACM 978-1-4503-2939-2/14/06 ...$15.00.
DOI STRING GOES HERE .

1. INTRODUCTION

In the present paper we introduce a fragment of *role based access control* (RBAC), this fragment is called *bi-sorted role based access control* (RBÄC). RBÄC suggests to treat permissions via *demarcations* separately from subjects via *proper roles*. Being a fragment implies that RBÄC might be directly used with existing RBAC implementations. Nonetheless RBÄC has an added value which lies mainly in the conceptual boundaries that it introduces. As a first step towards validating this we have built a prototype implementation of our access control scheme on top of the graph database Neo4j (cf. Appendix A).

In the remainder of the paper we explain how the added value of RBÄC derives from the fact that permission and subject management are decoupled and a new higher level administrative level for access management is introduced. For practitioners working in physical access control such a decoupling implies that:

- The human resources team can be purely concerned with personnel management placing designation on individual subjects that make semantic sense without becoming overly concerned with the actual physical spaces that these roles grant/withhold.
- The site management team controlling the physical space of the company would designate certain areas as public, general-use, sensitive, etc. These *demarcations* stand in a hierarchy (in much the same way as do roles on the subject side), e.g. access to sensitive areas will probably imply access to the public ones. They do this without getting overly concerned with the actual employees that these demarcations invite/ban.
- The team of security officers then will set up the actual access control rules on an abstraction level suitable to their responsibilities without getting bogged down by either subject or permission management issues.

We will show how RBÄC enables many–to–many administrative mutations and ultimately leads to more organisational scalability. Besides that we speculatively state that in practice such an approach might be beneficial in the following senses:

- The elimination of communication errors between different teams trying to blend all aspects of AC into roles.

- Increased level of employees' privacy: only HR department directly works with employees' data.

- Less exposed infrastructure: only demarcations need be exposed; the actual building plans, they are mapped on, are only accessible to the relevant site managers.

In addition, we introduce natural and useful extension of RBÄC called R$^{\pm}$BÄC which makes use of *negative* counterparts of RBÄC's proper roles and demarcations. Introduction of negative counter parts is further supported by the existence of a Galois connection between the negative and positive part of the specification, Definition 4. Our design of R$^{\pm}$BÄC governed by the found regularities excludes combinations of negative and positive permissions in a *single* object of indirection (this has been an issue in the widely-accepted NIST RBAC [19]).

For practitioners this typically means, that, when an exception arises, instead of exercising in transformations of logical expressions and consequent rewriting of existing AC rules, security officers can *generalise* the exception, and cleanly and compositionally adapt the previous AC specification touching far fewer rules (for very practical examples see [1]). For that matter we shall give an extensive example on how and why negative objects of indirection could work in practice. In particular, professionals working in the subject management perspective can rest assured the following will always hold:

- assigning a positive role will not, inadvertently, revoke permissions from the subject;

- assigning a negative role will not, inadvertently, reassign permissions to the subject.

On the other hand, professionals working in the permission management perspective can rest assured that:

- assigning a positive demarcation will not, inadvertently, ban subjects from the permission;

- assigning a negative demarcation will not, inadvertently, invite subjects to the permission.

At the same time, the third administrative perspective (access management) allows security officers degrees of freedom on an abstraction level suitable for their responsibilities. The only small price to pay by the security officers for having the framework be such well-behaved to the people they delegate subject/permission management tasks to is that they may never use a positive object of indirection in a negative context or use a negative object of indirection in a positive context. Which, in fact, forces them to make clear design trade-offs as to what aspects of the security policy are handled positively and which are handled negatively.

In general, we take an approach that somewhat breaks with the trend of extending RBAC. Even though we will present the section on language first and the section on Galois connections second, in our approach we actually went back to first principles first and developed the language(s) second. In particular once the Galois connections were clear it became rather straightforward to uncover the, so far unexploited, symmetries in existing RBAC which finally led to RBÄC and then to $R^{\pm}$BÄC.

2. ACCESS CONTROL

Access control (AC) is all about *who* gets to do *what*. As such it involves at least two domains. We distinguish the set of *subjects* S (the *who*), the set of *permissions* P (the *what*), and the *access relation* $SP \subseteq S \times P$ (*who* gets to do *what*). In Figure 1a this situation is depicted using an entity relationship diagram[1]

This very limited conceptual framework suffices in practice only as long as S and P are relatively small sets. If the cardinalities of S and P are too large there are too many pairs in $S \times P$ to consider. We call this the *subject–permission explosion problem*.

In order to deal with the subject–permission explosion we can turn to the framework of *role–based access control* (RBAC) [18]. In this framework an extra layer of indirection in the form of a set of *roles* R is introduced. Now permissions are not assigned directly to subjects anymore, they are instead assigned to roles which are, in turn, assigned to subjects. This situation is shown in Figure 1b, the two white diamonds represent the *subject assignment* and *permission assignment* relations, the grayed out diamond still represents the *access relation* which now *follows* from the subject assignment and the permission assignment relations, i.e.: it is a *computed* relation. As can be seen from Figure 1b: we do not (yet) have the desirable square domain structure where we would be able to recover the access relation *on the abstract level*.

Nevertheless, assuming that the set of roles remains small, classic RBAC is an adequate solution to the subject–permission explosion. There is, however, an important extra dimension to consider when evaluating the merit of an access control scheme. In practice it is rarely the case that an access control policy is written once from scratch and remains static from that moment onwards, citing [18]:

> The ability to modify policy to meet the changing needs of an organization is an important benefit of RBAC.

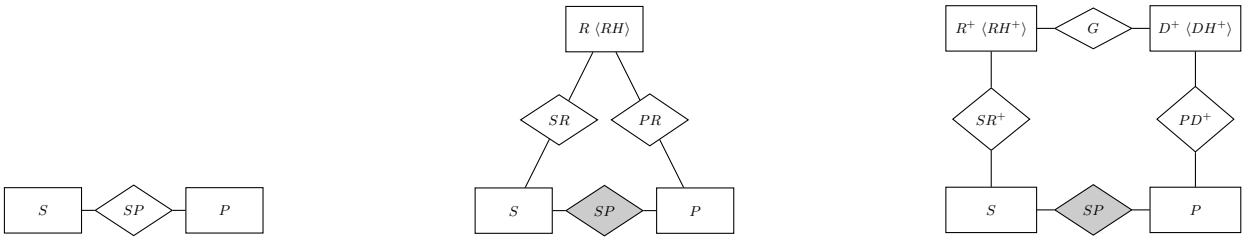
Yet, RBAC is rather limited where it concerns administrative degrees of freedom. We can distinguish the following four basic RBAC mutations:

1. Enroll a subject $s \in S$ to role $r \in R$, i.e.: add (s, r) to SR.
2. Disenroll a subject $s \in S$ from role $r \in R$, i.e.: remove (s, r) from SR.
3. Assign a permission $p \in P$ to role $r \in R$, i.e.: add (p, r) to PR.
4. Unassign a permission $p \in P$ from role $r \in R$, i.e.: remove (p, r) from PR.

In the first case the effect on the access relation SP is that a *single* subject gains (potentially) *many* permissions. In the second case the effect on the access relation SP is that a

[1]Entity relationship diagrams are often used to describe relational databases, here we use them on a more conceptual level. Please note that this is, in fact, the original intent in [6]

[2]Strictly speaking $\langle RH \rangle$, $\langle RH^+ \rangle$ and $\langle DH^+ \rangle$ are reflexive relations that can be drawn inside their own diamonds, however, for presentation purposes, we write them directly after the entity names.



(a) AC (b) RBAC (c) RBÄC

Figure 1: Entity/relationship diagram depicting the entities (rectangles) and relations (diamonds) involved in the various access control schemes discussed in the paper.[2]

single subject loses (potentially) *many* permissions. In the third case the effect on the access relation SP is that (potentially) *many* subjects gain a *single* permission. In the fourth case the effect on the access relation SP is that (potentially) *many* subjects lose a *single* permission. In short, the effect of an atomic RBAC mutation on SP is always *either* one–to–many *or* many–to–one, never *many–to–many*. We call this the *administrative micro-stepping problem*. We see that, in order to obtain administrative scalability we need to break away from individual subjects and individual permissions, hence, at the very least, mutations with a many–to–many effect on the access relation must be enabled.

It would be inaccurate to claim that RBAC does *not* allow many–to–many mutations in the aforementioned sense. It does so, however, in a counterintuitive way, namely through the *inheritance* relation: RH. Combined with items 1 up to 4 above the following completes the *exhaustive* list of possible RBAC mutations:

5. Make a role r a senior role with respect to role r', i.e.: add (r, r') to RH.
6. *Unmake* a role r as a senior role of r', i.e.: *remove* (r, r') from RH.

For case 5 the effect on the access relation is that (potentially) *many* subjects gain (potentially) *many* permissions; for case 6 the effect on the access relation is that (potentially) *many* subjects lose (potentially) *many* permissions.

The inheritance relation RH is, by design, something that has a *static* flavor: for a properly engineered set of roles it is desirable that the inheritance relation is fixed at the time the set of roles is fixed and does not change *dynamically* (it should typically only change in response to new roles being added or old ones being removed). As an example consider the role $\mathsf{manager} \in R$ and $\mathsf{employee} \in R$ such that $(\mathsf{manager}, \mathsf{employee}) \in RH$, i.e.: manager is senior role with respect to employee. The latter entails that the manager role inherits all the permissions of the employee role, and, vice versa, the employee role inherits all the subjects of the manager role. Over time it may well be the case that the manager and/or employee roles gain and/or lose new permissions and/or subjects. However we would expect that the latter is realized by introducing/removing pairs from PR and/or SR and *not* by introducing/removing pairs from RH. In particular we would not expect the inheritance relation between managers and employees to be removed as the latter could have many more unpredictable effects both upward as well as downward in the entire role hierarchy.

So we may conclude that, although, in principle, RBAC supports many–to–many administrative mutations, in practice, it does not provide the necessary safeguards to do so effectively[3]. In the rest of this paper we present a solution to this problem in the form of a safe, two–sorted fragment of RBAC (RBÄC), which, at the same time, can be seen as a conceptual enrichment. In particular we partition the set of roles R into *proper roles* R^+ and *demarcations* D^+ and stipulate that proper roles can only have subjects assigned to them and demarcations can only have permissions assigned to them. This then leads to a corresponding partition on RH into *proper role inheritance*, *demarcation inheritance*, and *grant–rule–pairs*, the latter being suitable for *dynamic* many–to–many administrative mutations.

As it turns out, the two–sorted fragment of RBAC naturally leads us to a proper extension with *negative roles*. So far, in the literature, negative roles have definitely been explored yet usually dismissed on grounds of being too *confusing*, [19, subsection 7.3], the later standard [10] avoids discussing negative permissions or roles. We show how negative roles are introduced in a clean way into RBÄC whilst avoiding much of the confusion that would occur in the case of adding them to classic RBAC directly.

We support our observations with a thorough mathematical analysis. In particular we identify two Galois connections between the domains of subjects and permissions and between the domains of positive (grant) sets and negative (withhold) sets. We interpret these results as additional confirmation that there is structure on the underlying semantic domain to justify the additional conceptual distinctions that we make (roles vs. demarcations, positive vs. negative roles).

3. LANGUAGE

3.1 Role Based Access Control

The formalism we discuss in this section was previously introduced in [18] as RBAC_1. The only difference is that we will not concern ourselves here with the distinction between *having* a role and *activating* it [17] (i.e.: we simply assume a subject will always activate all of its roles).

Definition 1 (RBAC)**.** The following two sets form the *principal semantic domains* underlying RBAC:

- Let S be a set of *subjects* (sometimes notions of *users*, [17, 18], or *principals*, [2], are used).

[3]In fact, intuition suggests that not having a rather stable role hierarchy will amplify the problem of "maintaining a good understanding of the currently implemented policy", a common problem among practitioners [4].

- Let P be a set of *permissions* (sometimes introduced as *actions on objects*).

The following primitives form the *syntax* of RBAC meaning that these are entities which the *user* of RBAC (policy designer/security officer) *directly manipulates*:

- Let R be a set of *roles*.
- Let $SR \subseteq S \times R$ be a subject–role assignment relation.
- Let $PR \subseteq P \times R$ be a permission–role assignment relation.
- Let $RH \subseteq R \times R$ be a role–hierarchy relation, RH is required to be *acyclic*.

The following definitions lay down the *semantics* of RBAC, meaning they map the syntactical constructs into the principal semantic domains.

- We define $\geq$ as the transitive reflexive closure of RH. For two roles $r, r' \in R$ such that $r > r'$ we say r is a *senior* role of r', or, vice versa, r' is a *junior* role of r.
- We define the *access relation* $SP \subseteq S \times P$ such that $(s, p) \in SP$ iff there exists roles r, r' in R such that the following conditions are fulfilled:
  1. $(s, r) \in SR$, i.e.: subject s is a member of role r.
  2. $r \geq r'$, i.e.: $r = r'$ or r is senior role of r'.
  3. $(p, r') \in PR$, i.e. permission p is part of role r'.$\diamond$

The numbered conditions of the last definition can be visualized superimposed on the triangle in Figure 1b: one must traverse the north–west and north–east sides in order to prove a tuple (s, p) belongs to SP (the south side).

Example 1 (RBAC)**.** We consider a simple example where:

$$R = \{\mathsf{manager}, \mathsf{employee}\},\ RH = \{(\mathsf{manager}, \mathsf{employee})\},$$
$$SR = \{(s_1, \mathsf{manager}), (s_2, \mathsf{employee})\},$$
$$PR = \{(p_1, \mathsf{manager}), (p_2, \mathsf{employee}), (p_3, \mathsf{employee})\}$$

In Figure 2a we show this policy graphically. $\triangle$

In the diagramming style of Figure 2a the dotted lines represent the subject–role and permission–role assignment relations and the solid lines represent the role–hierarchy relation. By convention the senior roles are drawn towards the top of the diagram and the more junior roles are drawn towards the bottom of the diagram. Because subjects are inherited top–down they are drawn one level up from the roles they are assigned, and because permissions are inherited bottom–up they are drawn one level down from the roles they are assigned to. It now becomes possible to follow the paths through the diagram starting from some subject and always going down and/or to the right in order to derive which subject–permission pairs are in the access relation. For this example those are:

1. (s_1, p_1) using path: $s_1, \mathsf{manager}, p_1$
2. (s_1, p_2) using path: $s_1, \mathsf{manager}, \mathsf{employee}, p_2$
3. (s_1, p_3) using path: $s_1, \mathsf{manager}, \mathsf{employee}, p_3$
4. (s_2, p_2) using path: $s_2, \mathsf{employee}, p_2$
5. (s_2, p_3) using path: $s_2, \mathsf{employee}, p_3$

As can be seen the final semantics of the policy become:

$$SP = \{(s_1, p_1), (s_1, p_2), (s_1, p_3), (s_2, p_2), (s_2, p_3)\}$$

The example also demonstrates the limited administrative degrees of freedom for RBAC. Consider, for example, the case where we want to remove permission (s_2, p_2) this means we would need to perform *two* actions. First we must remove $(p_2, \mathsf{employee})$ from PR, next we must *add* $(p_2, \mathsf{manager})$ to PR (otherwise also (s_1, p_2) would be cancelled from SP). This is shown in Figure 2b.

3.2 Two–sorted Role Based Access Control

Definition 2 (RBÄC)**.** We let the principal semantic domains S and P be as in Definition 1. The following forms the *syntax* of RBÄC meaning that these are entities which the *users* of RBAC (policy designer/security officer, HR–dept., IT–dept./Infrastructure–dept.) *directly manipulate*:

- Let R^+ be a set of *proper roles* and let D^+ be a set of *demarcations*, both sets are required to be disjoint.
- Let $SR^+ \subseteq S \times R^+$ be a subject–role assignment relation and let $PD^+ \subseteq P \times D^+$ be a permission–demarcation assignment relation.
- Let $RH^+ \subseteq R^+ \times R^+$ be a proper–role–hierarchy relation, and let $DH^+ \subseteq D^+ \times D^+$ be a demarcation–hierarchy relation, both DH^+ and RH^+ are required to be *acyclic*.
- Let $G \subseteq R^+ \times D^+$ be a *grant* relation.

The following definitions lay down the *semantics* of RBÄC meaning they map the syntactical constructs into the principal semantic domains.

- We define $\geq_{\mathsf{r}}^+$ as the transitive reflexive closure of RH^+ and $\geq_{\mathsf{d}}^+$ as the transitive reflexive closure of DH^+.
- We define the *access relation* $SP \subseteq S \times P$ such that $(s, p) \in SP$ iff there exists roles $r, r' \in R^+$ and demarcations $d, d' \in D^+$ such that the following conditions hold:
  1. $(s, r) \in SR^+$, i.e.: subject s is a member of proper role r.
  2. $r \geq_{\mathsf{r}}^+ r'$, i.e.: $r = r'$ or r is senior role of r'.
  3. $(r', d') \in G$, i.e.: proper role r' is granted access to demarcation d'.
  4. $d' \geq_{\mathsf{d}}^+ d$, i.e. $d = d'$ or d is a sub-demarcation of d'.
  5. $(p, d) \in PD^+$, i.e. permission p is part of demarcation d. $\diamond$

The enumerated conditions of the last definition can be visualized superimposed on the square in Figure 1c: one must traverse the west, north and east sides in order to prove a tuple (s, p) belongs to SP (the south side).

Example 2 (RBÄC)**.** We consider a simple example where:

$$R^+ = \{\mathsf{manager}, \mathsf{employee}\},\quad RH^+ = \{(\mathsf{manager}, \mathsf{employee})\},$$
$$D^+ = \{\mathsf{red}, \mathsf{amber}, \mathsf{green}\},$$
$$DH^+ = \{(\mathsf{red}, \mathsf{amber}), (\mathsf{amber}, \mathsf{green})\},$$
$$SR^+ = \{(s_1, \mathsf{manager}), (s_2, \mathsf{employee})\},$$
$$PD^+ = \{(p_1, \mathsf{red}), (p_2, \mathsf{amber}), (p_3, \mathsf{green})\},$$
$$G = \{(\mathsf{manager}, \mathsf{red}), (\mathsf{employee}, \mathsf{green}), (\mathsf{employee}, \mathsf{amber})\}$$

(a) Before (b) After

Figure 2: The mutation of Figure 3 for RBAC: administrative micro-stepping.

(a) Before (b) After

Figure 3: Lowering the access–level of the employee proper role in the RBÄC case.

This situation is shown in Figure 3a. △

The first thing to note from the diagram in Figure 3a is the redundancy in deriving SP pairs, e.g. for a pair (s_1, p_3) for example, there are three ways to prove this pair belongs in the access relation SP. The first is through the path s_1, manager, red, amber, green, p_3, the second is through the path s_1, manager, employee, amber, green, p_3 and the third is through the path s_1, manager, employee, green, p_3. The latter type of redundancy would be easy to remove by tool support: we could simply warn the user that the pair (employee, green) is subsumed by the pair (employee, amber) so removing the former would be cleaner. But what about the remaining redundancy: the square manager, red, amber, employee? We posit this type of redundancy is not harmful, in fact it is crucial to administrative scalability. The formal reason is that both horizontal edges constitute fixed points in the Galois connection that exists between subject and permission sets and neither can be removed without changing SP (cf. Example 4). A more conceptual reason, perhaps, is that all four pairs express a distinct *intent* and, from that perspective, neither pair subsumes any of the others. Let us evaluate the four pairs to see what their intents are:

1. (manager, red) $\in G$: managers need red security clearance to do their job.
2. (red, amber) $\in DH^+$: red is a strictly higher security clearance than amber.
3. (employee, amber) $\in G$: employees need amber security clearance.
4. (manager, employee) $\in RH^+$: managers can do anything that employees can.

What would happen if, in the example, we find out that item 3 is inaccurate: actually all employees can perform all their responsibilities having only green security clearance. In that case the principle of least privilege dictates to lower the security clearance of employees to green. Now an administrative mutation has to be addressed and we see how the aforementioned redundancy actually plays out to our advantage: (employee, amber) can be simply removed from G *and all the other intents, 1, 2 and 4, remain valid.* The result is shown in Figure 3b. Note that the redundancy that was in the diagram ensures that the impact of this *local* change remains limited to the proper role employee: the proper role manager is not affected.

Contrast this to the classic situation in Figure 2. In that case the mutation required examination of individual permissions, i.e.: the change occurred on the micro–level. Moreover the change violated locality because it impacted the proper role manager.

3.3 Positive and Negative Roles and Demarcations

Definition 3 ($\mathrm{R}^{\pm}$BÄC)**.** We let the principal semantic domains S and P be as in Definition 1. The *syntax* of $\mathrm{R}^{\pm}$BÄC is defined analogously to Definition 2 except for the fact that we introduce *positive* and *negative* versions for all the relevant types.

- Let R^+ be a set of *(positive) proper roles*, let R^- be a set of *negative proper roles* (also called *castes*), let D^+ be a set of *(positive) demarcations* and let D^- be a set of *negative demarcations* (also called *delimitations*).
- Let $SR^+ \subseteq S \times R^+$ and $SR^- \subseteq S \times R^-$ be the positive and negative subject–role assignment relations and let $PD^+ \subseteq P \times D^+$ and $PD^- \subseteq P \times D^-$ be the positive and negative permission–demarcation assignment relations.
- Let $RH^+ \subseteq R^+ \times R^+$ and $RH^- \subseteq R^- \times R^-$ be the positive and negative proper–role–hierarchy relations, and let $DH^+ \subseteq D^+ \times D^+$ and $DH^- \subseteq D^- \times D^-$ be the positive and negative demarcation–hierarchy relations, all four hierarchies are required to be *acyclic*.
- Let $T = \{\langle G_1, W_1\rangle, \ldots \langle G_n, W_n\rangle\}$ be an indexed set of *specification tuples* for each $i \leq n$ consisting of a

grant relation $G_i \subseteq R^+ \times D^+$ and a withhold relation $W_i \subseteq R^- \times D^-$.

The following definitions lay down the *semantics* of R$^{\pm}$BÄC meaning they map the syntactical constructs into the principal semantic domains.

- We define $\geq_{\mathsf{r}}^+$, $\geq_{\mathsf{r}}^-$, $\geq_{\mathsf{d}}^+$ and $\geq_{\mathsf{d}}^-$ as the transitive reflexive closure of RH^+, RH^-, DH^+, and DH^- respectively.
- We define the *access relation* $SP \subseteq S \times P$ such that $(s,p) \in SP$ iff there exists a specification tuple $\langle G, W\rangle \in T$ such that there exist positive proper roles $r, r' \in R^+$ and positive demarcations $d, d' \in D^+$ for which conditions 1. up to 5. of Definition 2 hold, *and*, in addition, there do *not* exist negative proper roles $r, r' \in R^-$ and negative demarcations $d, d' \in D^-$ for which the following conditions hold:
  6. $(s, r) \in SR^-$, i.e.: subject s is a member of negative proper role r.
  7. $r \geq_{\mathsf{r}}^- r'$, i.e.: $r = r'$ or r is a negative senior role of r'.
  8. $(r', d') \in W$, i.e.: negative proper role r' is withheld access to d'.
  9. $d' \geq_{\mathsf{d}}^- d$, i.e. $d = d'$ or d is a negative subdemarcation of d'.
  10. $(p, d) \in PD^-$, i.e. permission p is part of negative demarcation d. ◇

Conditions 6 to 10 are completely analogous to conditions 1 to 5 in Definition 2. It is important to note that the negative aspects of the language are kept completely separate from the positive aspects on all levels. In this definition, withhold pairs always override grant pairs. It is possible to reverse this w.l.o.g. throughout the paper, however in the definitions that follow we implicitly assume that withhold pairs override grant pairs.

Example 3 (R$^{\pm}$BÄC)**.** As an example of negative rules we consider the policy shown in Figure 3a, the extensions of all the relevant (positive) domains are given in Example 2.

Now let us assume we are in the position of a security officer who was issued a request to change a security policy: subject s_2 (John) should no longer have permission p_2 (Root–Access).

As a general principle we would like to keep the security policy general, i.e. with no explicit references to particular subjects or permissions. Hence, we shall try to elicit *why* it is the case that John should no longer have root permission. Our goal is to generalize this and embed this knowledge into our organization's security policy, applying the same rule to all the other subjects and permissions under our control. Assume that instead of refactoring the existing (positively specified) policy into a new set of positive rules, we prefer to express the change with a withhold clause and avoid touching the present positive specification.

To do so, we consult with the HR dept. about *what the defining property of John is, that makes him unsuitable for wielding root permission*, and with the IT dept. about *what the property of the root permission is, which makes it off–limits to someone like John.* Let us assume the HR dept. identifies the problem as John lacking the *certified* system

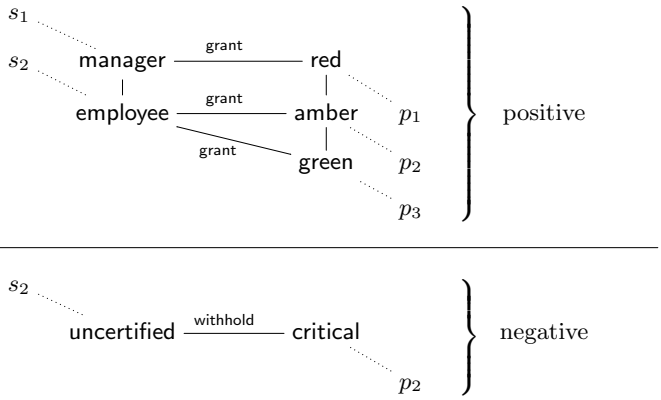


Figure 4: Example with a combination of positive and negative specification styles.

administrator training diploma, and the IT dept. identifies the problem as root permission being a security *critical* permission.

After having figured out the identifying features, a new negative role $R^- = \{\mathsf{uncertified}\}$ and a new negative demarcation $D^- = \{\mathsf{critical}\}$ are introduced. The HR and IT departments are henceforth instructed to maintain these designations accurately for *all* relevant subjects and *all* relevant permissions in the organization. For the moment this results in $(s_2, \mathsf{uncertified}) \in SR^-$ and $(p_2, \mathsf{critical}) \in PD^-$. The complete policy is shown in Figure 4. △

There are two reasons why in R$^{\pm}$BÄC the introduction of negative roles is less confusing than is the case in RBAC. The first reason lies in the fact that R$^{\pm}$BÄC does not mix positive and negative concepts. The framework is fully *monotone* with respect to positive proper roles and demarcations, and it is fully *antitone* with respect to negative proper roles and demarcations, i.e.: assigning a positive proper role is guaranteed to only *add* permissions, while assigning a negative proper role is guaranteed to only *remove* permissions from the subject. Similarly, assigning a positive demarcation is guaranteed to only *invite* subjects to use the permission, while assigning a negative demarcation is guaranteed only to *ban* subjects from using the permission. These two separate perspectives are illustrated in Figure 5.

The second reason lies in the fact that R$^{\pm}$BÄC does not mix subject and permission concepts. As is shown in the example: when a negative (withhold) rule is used there is always a property of the subject (i.e.: uncertified) and a property of the permission (i.e.: critical) involved. In RBAC these two aspects have to be mapped onto one and the same object of indirection which makes it hard to elicit meaningful negative roles.

4. RBÄC AND R$^{\pm}$BÄC IN RELATION TO RBAC

To properly understand the relationship between RBAC and RBÄC we need to distinguish at least three levels: the formal, the conceptual and the expressive. On the formal level RBÄC is a fragment of RBAC because we do not allow roles that simultaneously connect subjects and permissions. On the conceptual level RBÄC is an enrichment of RBAC because we make conceptual distinctions that RBAC simply does not make. Finally, in terms of expressiveness both for-

Subject: John	
Positive Roles	Negative Roles
employee	uncertified
...	...

(a) Subject Management Perspective

Permission: Root–Access	
Demarcations	Delimitations
amber	critical
...	...

(b) Permission Management Perspective

Figure 5: Example for the two administrative perspectives that act at micro level.

malisms are equivalent because there are one–to–one translations in both directions.

Theorem 1. *There exists a linear translation from RBAC to RBÄC and, vice versa, there exists a linear translation from RBÄC back to RBAC.*

Proof. We give a constructive proof. First we consider the case of translating RBAC to RBÄC. For this we proceed by mapping each role to a proper role and demarcation.

$$\begin{aligned} R^+ &= \{\mathsf{proper}[r] \mid r \in R\} \\ D^+ &= \{\mathsf{demarc}[r] \mid r \in R\} \\ SR^+ &= \{(s, \mathsf{proper}[r]) \mid (s,r) \in SR\} \\ PD^+ &= \{(p, \mathsf{demarc}[r]) \mid (p,r) \in PR\} \\ RH^+ &= \{(\mathsf{proper}[r], \mathsf{proper}[r']) \mid (r,r') \in RH\} \\ DH^+ &= \{(\mathsf{demarc}[r], \mathsf{demarc}[r']) \mid (r,r') \in RH\} \\ G &= \{(\mathsf{proper}[r], \mathsf{demarc}[r]) \mid r \in R\} \end{aligned}$$

it is clear that the size of the new policy is linear in the size of the original policy and defines the same access relation.

Next we consider the case of translating RBÄC to RBAC. For this we proceed by identifying concepts: proper roles are roles that are not (directly) assigned any permissions and demarcations are roles that are not (directly) assigned any subjects:

$$\begin{aligned} R &= R^+ \cup D^+ & RH &= RH^+ \cup DH^+ \cup G \\ SR &= SR^+ & PR &= PD^+ \end{aligned}$$

clearly the size of the new policy is linear in the size of the original policy and defines the same access relation. □

As can be seen the first case in the proof completely fixes G, i.e.: the only reasonable choice is to set the map as it is was defined, which implies there are no degrees of freedom left. This demonstrates clearly that in RBAC the access management administrative perspective (north side of Figure 1c) is not present. Which, in turn, constitutes formal confirmation of the observations that we detailed in the introduction: in RBAC it is not possible to perform many–to–many administrative mutation on an abstract level unless we do this through the role–hierarchy relation (but this is not desirable for obvious reasons).

The case of $\mathrm{R}^{\pm}$BÄC is different: there is no straightforward translation from $\mathrm{R}^{\pm}$BÄC to RBAC because the former allows a (restricted form) of negation and the latter does not. Clearly, in the other direction, it is trivial to translate RBAC to $\mathrm{R}^{\pm}$BÄC using the first part of Theorem 1.

5. ACCESS CONTROL SEMANTICS

This section discusses the tacit connections between subjects and their permissions and between the positive (grant) and negative (withhold) style of specification.

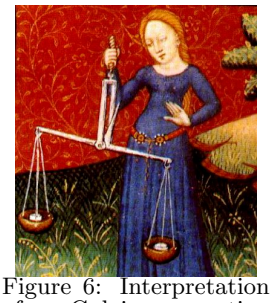


Figure 6: Interpretation of a Galois connection as an inherent trade–off, [22].

In the remainder of this section we show the existence of two Galois connections. The first between subject sets and permission sets and the second between grant sets and withhold sets (which can be seen as the positive and negative semantic objects corresponding to the positive and negative parts of an $\mathrm{R}^{\pm}$BÄC specification). In this section we will follow a technique whereby we assume some target access relation that may be implicit in the organization but at least still needs to be elicited and formalized. We then analyse w.r.t. this target access relation what are the tacit structures on S and P that can potentially be exploited for writing concise access control policies. First and foremost this is a syntax-agnostic analysis that acts entirely on the semantic domains.

Definition 4 (Target Access Relation)**.** We let S be some fixed background set of subjects, we let P be some fixed background set of permissions, and we let $SP_{\circledcirc} \subseteq S \times P$ be some fixed, target access relation. ◇

5.1 Galois Connection: Subjects–Permissions

Definition 5 (Powersets)**.** Let $\mathcal{S} = 2^S$ be the set of sets of subjects and $\mathcal{P} = 2^P$ be the set of sets of permissions. These sets are partially ordered w.r.t. set-inclusion ordering. ◇

Using the full powersets over the principal semantic domains we look at what are possible grant pairs for the *given target* access relation: we define for any given set of subjects the greatest common set of granted permissions, and, vice versa, for any given set of permissions the greatest common set of invited subjects.

Definition 6 (Subject/Permission Adjoints)**.** We define the *greatest common granted permission adjoint* $\mathcal{SP} : \mathcal{S} \to \mathcal{P}$ and the *greatest common invited subject adjoint* $\mathcal{PS} : \mathcal{P} \to \mathcal{S}$ as follows:

$$\begin{aligned} \mathcal{SP}(\hat{S}) &= \bigcup\{\hat{P} \in \mathcal{P} \mid \hat{S} \times \hat{P} \subseteq SP_{\circledcirc}\}, \\ \mathcal{PS}(\hat{P}) &= \bigcup\{\hat{S} \in \mathcal{S} \mid \hat{S} \times \hat{P} \subseteq SP_{\circledcirc}\} \end{aligned}$$

A *grant pair* $(S', P') \in \mathcal{S} \times \mathcal{P}$ is a *fixed–point* pair iff it holds that $S' = \mathcal{PS}(P')$ and $P' = \mathcal{SP}(S')$. ◇

Theorem 2 (Subject/Permission Connection)**.** $\langle \mathcal{SP}, \mathcal{PS} \rangle$ *is an anti-tone Galois connection between* $\mathcal{S}$ *and* $\mathcal{P}$*.*

Proof. Connection $\langle \mathcal{SP}, \mathcal{PS} \rangle$ is a particular case of *relation-generated* Galois connections class. A general proof is at hand from [20, Theorem 2.5.1], [15, Theorem 13]. □

Intuitively this means that there exists an inherent trade–off between abstracting over the set of subjects and abstracting over the set of permissions: if a grant pair is very general

with respect to the subject set, it must be very specific with respect to the permission set and, vice versa, if a grant pair is very general with respect to the permission set it must be very specific with respect to the subject set. The necessity to balance the left– and the right–hand–side in order to arrive at beautiful (concise, readable, understandable, maintainable) specifications is illustrated in Figure 6.

Example 4 (Subject/Permission Fixed–Point Pairs)**.** Let us consider once more the RBÄC specification of Example 2. By applying the upper and lower adjoint we can compute the fixed point pairs over SP. For the example the following are fixed–point pairs: $(\{s_1, s_2\}, \{p_2, p_3\})$, and $(\{s_1\}, \{p_1, p_2, p_3\})$. These grant pairs precisely correspond to the proper–role/demarcation pairs $(\mathsf{employee}, \mathsf{amber})$ and $(\mathsf{manager}, \mathsf{red})$. △

5.2 Galois Connection: Positive–Negative

For reasons of exposition in this section we simplify the situation with respect to grant and withhold pairs. In the previous section we treated grant pairs as consisting of a *set* of subjects and a *set* of permissions, in this way there are many possible combinations that lead to the same access relation, in this section we will assume w.l.o.g. that the subject set and the permission set are always *singleton*. Note that it is always possible to normalize any set of grant/withhold pairs to such a set of singleton grant/withhold pairs.

Definition 7 (Grant/Withhold Set)**.** A *grant set* $\mathsf{G} \subseteq \{\mathsf{grant}\} \times S \times P$ consists of a marked set of subject/permission pairs written as terms: $\mathsf{grant}[s, p] \in \mathsf{G}$, with $\mathcal{G}$ we denote the set of all such grant sets. We define a partial order $\sqsupseteq$ on $\mathcal{G}$ such that $\mathsf{G} \sqsupseteq \mathsf{G}'$ iff for all $\mathsf{grant}[s, p] \in \mathsf{G}'$ there exists $\mathsf{grant}[s, p] \in \mathsf{G}$ we say G is *more permissive* than G', for two grant sets G, G' we define their join $\mathsf{G} \sqcup \mathsf{G}'$ as the (set theoretical) union. A *withhold set* $\mathsf{W} \subseteq \{\mathsf{withhold}\} \times S \times P$ consists of a marked set of subject/permission pairs written as terms: $\mathsf{withhold}[s, p] \in \mathsf{W}$, with $\mathcal{W}$ we denote the set of all such withhold sets. We define a partial order $\sqsupseteq$ on $\mathcal{W}$ such that $\mathsf{W} \sqsupseteq \mathsf{W}'$ iff for all $\mathsf{withhold}[s, p] \in \mathsf{W}$ there exists $\mathsf{withhold}[s, p] \in \mathsf{W}'$ we say W is *more permissive* than W', for two withhold sets W, W' we define their join $\mathsf{W} \sqcup \mathsf{W}'$ as the (set theoretical) intersection. Note that we overloaded $\sqsupseteq$ and $\sqcup$ for grant and withhold sets, it will be clear from the context which lattice is intended. For a given grant set G and a given withhold set W we define the corresponding access relation $SP(\mathsf{G}, \mathsf{W}) = \{(s, r) \in S \times R \mid \mathsf{grant}(s, r) \in \mathsf{G}, \mathsf{withhold}(s, r) \notin \mathsf{W}\}$. ◇

Definition 8 (Grant/Withhold Adjoints)**.** We define the *negative correcting adjoint* $\mathcal{GW} : \mathcal{G} \to \mathcal{W}$ and the *positive correcting adjoint* $\mathcal{WG} : \mathcal{W} \to \mathcal{G}$ as follows:

$$\mathcal{GW}(\mathsf{G}) = \bigsqcup\{\mathsf{W} \in \mathcal{W} \mid SP(\mathsf{G}, \mathsf{W}) \subseteq SP_{\circledcirc}\},$$

$$\mathcal{WG}(\mathsf{W}) = \bigsqcup\{\mathsf{G} \in \mathcal{G} \mid SP(\mathsf{G}, \mathsf{W}) \subseteq SP_{\circledcirc}\}$$

A *grant/withhold specification pair* $(\mathsf{G}, \mathsf{W}) \in \mathcal{G} \times \mathcal{W}$ is a *fixed–point* pair iff it holds that $\mathsf{W} = \mathcal{GW}(\mathsf{G})$ and $\mathsf{G} = \mathcal{WG}(\mathsf{W})$. ◇

Theorem 3 (Grant/Withhold Connection)**.** *$\langle \mathcal{GW}, \mathcal{WG} \rangle$ is an antitone Galois connection between $\mathcal{G}$ and $\mathcal{W}$.*

Proof. Identical to the proof of Theorem 2. □

Intuitively, if we are more permissive in the positive part of a specification tuple this means that we must be less permissive in the negative part of the specification tuple and, vice versa, if we are more permissive in the negative part of the specification tuple we must be less permissive in the positive part of the specification.

6. RELATED WORK

The research on RBAC was initiated in the 90s [18]. The standard was introduced in [7, 10, 19]. Since then the conceptual core of RBAC has remained stable. Only few authors reported on work that goes beyond the conceptual framework as given in the standard.

Oh and Park [14] were, to our knowledge, the first who realized that permissions should be grouped independently of roles. They called such grouped permissions *tasks*. Tasks were then assigned to roles making their model well-tailored for enterprise environment where work-flow has to be controlled. Task inheritance is not directly addressed. The findings of [14] underline how the need for a separate concept for grouping on the permission side arises naturally in practice.

The more recent critique of the ANSI standard of RBAC [12] contained several suggestions for improving the standard. We draw reader's attention to Suggestion 5 "The semantics of role inheritance should be clearly specified and discussed". Li et al. [12] rightly observed that role hierarchies can be viewed differently as RBAC combined different entities (subjects and permissions) into a single concept of a role. This observation allowed them to describe several peculiarities when role hierarchy was used solely for either user inheritance or permission inheritance. The authors of the ANSI standard for RBAC replied [8] on the aforementioned critique as follows: "... (the) proposal to interpret hierarchies differently in different circumstances is not conducive to conceptual simplicity in the model and is likely to lead to considerable confusion among practitioners". We agree on this point, but, moreover, we identify the reason for such a confusion as having *several* different hierarchies for *one* set of roles. Our first model (RBÄC) splits the concept of a role into a proper role and demarcation and introduces separate hierarchies for both of them therefore preventing the very cause of any possible confusion for interpretation of hierarchies.

The work by Kern et al. [11] on enterprise role-based access control clearly demonstrates the practicality of maintaining two distinct role hierarchies. The terminology they use is *enterprise* role on the enterprise level and *functional* role on the level of target systems. Our proposal is different from enterprise role-based access control in that we advocate a conceptual split right down the middle through the core of the access control model. Indeed in the latter work the triangular domain structure of RBAC remains untouched. Our goals are also slightly different in that we try to address administrative scalability (by preventing administrative microstepping) rather than trying to tie together disjoint access control models (i.e.: Oracle, UNIX, RACF) as per Kern's work. However, the fact that the same mechanism arises naturally in that context we consider as additional confirmation of the relevance of our work.

The work by Nyanchama and Osborn [13] on role graph models underlines once more the relevance of the dichotomy between roles and demarcations that we point out. At the

same time roles are still presented as the central object of indirection resulting in a slightly skewed but essentially still triangular domain structure. Another important difference is that Role Graphs are emergent, i.e.: their structure is *defined* in terms of permission (privilege) inclusion. More precisely the is-junior (is-senior) relation is *partly* computed based on privilege inclusion and *partly* given in the form of hints.

Due to the opinion, common in the community, that "... negative permission can be very confusing, especially in presence of general hierarchies" [19, subsection 7.3] and that it is better to "base decisions on permissions rather than exclusion" [16], very few researchers addressed the use of negative permissions. Alfaro et al. [1] "... point out the necessity of full expressiveness for combining both negative and positive conditions". However authors solely focus on the use of negative and positive rules in firewall languages, provide no formal analysis why mixing positive and negative styles is beneficial for AC management and, thus, argue about pros of such a mixture on a meta-level purely.

Tripunitara and Li [21] introduce a theory for comparing the expressive power of access control models. Their approach is to formalize an access control scheme as a state transition system and to define an expressivity ordering among schemes based on simulation relations. Of course an important point is then how strong or weak we require these simulation relations to be. They demonstrate this clearly by showing pathological cases such as the simulation of RBAC into DAC.

One point that is not addressed in the aforementioned work is *simplicity* (or alternatively: a cap on complexity). Tripunitara and Li [21]'s approach is used in [9] to prove that all existing access control schemes can be simulated by their *tag based access control* scheme but for that they use very liberal formalisation of their labeled transition systems, basically allowing the user to write arbitrary logic sentences and even to choose the actual logic (proof system). In contrast: under the most natural formalisation of an labeled transition system for the RBÄC scheme it may well turn out that RBÄC is *less* expressive (in the strict Tripunitara and Li [21] sense) than RBAC, while under a more liberal formalisation (based on Theorem 1) they would turn out to be equivalent.

In general the key to comparing access control languages cannot lie, solely, in their expressivity. If the latter were true then this problem would already have been solved by using some powerful variant of first or even second order logic. Yet there are many reports in the literature that indicate the problem instead arises on a conceptual level as users are struggling to specify clearly and concisely, to cope with changing policies over time and to keep an overview on the policy that is actually in effect at any given time [3–5].

7. CONCLUSION

As has been shown in the previous section, the problem with role hierarchies was identified by a number of researchers, but no formal analysis was carried out to investigate it. In this respect we can compare the common RBAC approach with macro physics which manipulates with pressure, temperature, etc. and our approach with statistical physics which starts with simple micro interactions (i.e. $\mathcal{SP}$ and $\mathcal{PS}$ adjoints) and, besides reproducing all macroscopic quantities, improves understanding of the analyzed system providing new insight and a firmer base for future developments. This new knowledge allowed us to introduce negative permissions in a non-confusing manner. In the rest of the section we sum up our findings.

One of the challenges in AC is designing a model which allows to manage a large number of permissions and subjects. While the set of subjects and permissions may considered as a given, a first logical step to take is to apply a "divide et impera" strategy: introduction of RBÄC is a natural progress in this direction. The discussed model RBÄC makes a clear distinction between subject and permission and in doing so introduces a number novel of concepts in access management.

It is often a challenge to design roles and demarcations which are no subject to exclusions. There are two main reasons for that: 1)when all factors influencing the role design are known, it renders very possible to have an unnecessarily granular role structure, and 2)when some factors are not known or missed, a re-factoring of the actual role structure may be very difficult in terms of refining roles and demarcations, re-assigning subjects and permissions and, finally altering access control rules. Introduction of negative counterparts of proper roles and demarcation in RBÄC resulted in $R^{\pm}$BÄC which addresses the mentioned issues. Therefore there are two ways a negative configuration can be used: 1)avoid micro-granular roles and have human-understandable roles, or/and 2) manage exclusions in a hustle-free manner.

We agree with [19] on that, although RBAC generally allows negative permissions, combination of positive and negative permissions can lead to an unpredictable AC decisions, especially if hierarchies are used. We feel that an anecdotal example comparing negative permissions to generally harmless sleeping pills which may trigger death if not used correctly is appropriate here. To avoid possible unpleasant side effects we give a prescription of how to use them:

- *use two different levels of indirections for subjects and permissions;*
- *maintain polarities: proper roles can only add permissions, positive demarcations can only invite subjects, proper negative roles can only withdraw permissions, negative demarcation can only ban subjects.*

Put formally, we claim that the monotonicity of the access relation defined through positive roles and positive demarcations, and the antitonicity of the access relation in the case of negative roles and negative demarcations are desirable properties that are relied upon by (and hence become particularly important from) the decoupled subject management and permission management perspectives.

7.1 Economy of Roles/Demarcation Split

Although the economy of the approach remains to be validated there are a number of concerns that, in absence of quantitative data, can be preliminarily addressed based on the theoretical analysis developed in the present paper.

One possible concern with the approach advocated here can be paraphrased as saying "more layers implies more things to engineer and sustain". We would like to point out here that this is primarily a question about economy of scale. As mentioned in the introduction, for smaller problem instances, or problem instances that are otherwise well-behaved, a different, simpler type of access control scheme might well be sufficient and therefore better suitable purely

on the merit of being *simpler*. In general it is true that the best solution to any problem is the *simplest* solution that qualifies as an adequate solution to the *whole* problem (in all of its facets).

It is therefore true that also bi-sorted role based access control must target a particular sweet spot of domain complexity. Our analysis suggest that the two main prerequisites on the organisation willing to adopt such a scheme will be:

- The organisation is of a certain minimum size, such that the three administrative perspectives (*subject*, *permission* and *access* management) are clearly discernible and are delegated to distinct teams or at least to distinct individuals.

- The organisation experiences a rate of administrative changes (*policy flux*) which merits the extra overhead in implementing the proposed decoupling and which overweights the cost of maintaining a more traditional static policy.

7.2 Economy of Positive/Negative Split

In a similar vein as the split between roles and demarcations we can discuss concerns about the split between positive and negative objects of indirection.

One such concern can be paraphrased as saying "in the presence of negation, policies can quickly become inconsistent." Inconsistency is a notion that is important for *logical specification* where the set of models characterized by a set of formulae can easily become empty if two or more of these formulae lead to a logical contradiction. Note that in our case the situation is different: there is always exactly one well-defined model (the access relation). That model is completely and unambiguously fixed by the policy specification. In fact it only makes sense to withhold a permission after it has first been granted. Inconsistency, in the strict logical sense, does not arise.

As soon as one starts considering constraints or invariants (written in some logical language) that are to hold *over* a given policy specification, consistency becomes important again and is a good candidate property to check for during static analysis of the constraints and invariants.

7.3 Economy of Graphical Languages

We would like to point out that although we use *graphical* language throughout the paper to explain and illustrate the *symbolic* expressions, in no way this means we advocate the exclusive use of graphical language in this context.

In general, simple examples denoted using graphical languages have the advantage to be more immediately intuitive. Perhaps because they appeal more directly to spatial pattern recognition abilities of the user. However this apparent advantage is offset by the fact that complexity of real–life problem instances rather quickly surpasses these same spatial pattern recognition abilities. In other words: graphs quickly get too cluttered, thereby limiting the utility of graphical languages.

At the same time, as shown in Appendix A, graphical languages can be rather intuitive for presenting query results where only a small portion of the graph is shown, i.e.: only the portion that is relevant to the query parameters.

7.4 Future Work

There are several avenues to consider for future work. The first, practical avenue, is to build an actual AC system employing all the introduced primitives and do a case study. We have already taken the first steps in this direction by building a prototype on top of the Neo4j graph database (cf. Appendix A).

On the theoretical side it is interesting to consider the introduction of *constraints*. Already explored for RBAC we know that constraints are a useful addition there. The situation for RBÄC (and $R^{\pm}$BÄC) we expect to be even more interesting with respect to constraints because of the two (four) sorted nature of the languages. As an example we can mention, besides the usual static and dynamic separation of duties, the introduction of bi–sorted *implication* constraints. With such a constraint one might for example say that some proper role intern implies a *negative* proper role uncertified.

Constraints are not to be confused with inheritance. A user of the AC language may, in principle set an inheritance relation between any two roles and/or demarcations and, in doing so, alter the resulting access function. There is no notion of an inheritance relation either *holding* or *not holding* for a given specification (it is *part of* the specification).

As mentioned before in the context of *consistency*: a logical constraint serves the purpose of restricting the possible role/demarcation assignments and it can either hold or not for a given set of assignments. As such, a second, more pragmatic, question to consider here would be how such constraints are best enforced or how a user can best be guided in satisfying them. This ties in with work on model-checking entire policy specifications.

8. ACKNOWLEDGEMENTS

We would like to thank Albert Dercksen for valuable input. Additionally we would like to thank Prof. Ninghui Li for constructive feedback and useful pointers to literature. Authors are thankful to anonymous reviewers for indicating improvement points.

References

[1] J. Alfaro, F. Cuppens, and N. Cuppens-Boulahia. Management of exceptions on access control policies. In *New Approaches for Security, Privacy and Trust in Complex Environments*, pages 97–108. Springer, 2007.

[2] M. Barletta, S. Ranise, and L. Viganò. Automated Analysis of Scenario-based Specifications of Distributed Access Control Policies with Non-Mechanizable Activities (Extended Version). *arXiv preprint arXiv:1206.3180*, 2012.

[3] L. Bauer, L. Cranor, R. Reeder, M. Reiter, and K. Vaniea. A user study of policy creation in a flexible access-control system. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, pages 543–552. ACM, 2008.

[4] L. Bauer, L. Cranor, R. Reeder, M. Reiter, and K. Vaniea. Real life challenges in access-control management. In *Proceedings of the 27th international conference on Human factors in computing systems*, pages 899–908. ACM, 2009.

[5] D. Botta, R. Werlinger, A. Gagné, K. Beznosov, L. Iverson, S. Fels, and B. Fisher. Towards understanding IT security professionals and their tools. In *Proceedings of the 3rd symposium on Usable privacy and security*, pages 100–111. ACM, 2007.

[6] P. P.-S. Chen. The entity-relationship model — toward a unified view of data. *ACM Transactions on Database Systems*, 1(1):9–36, Mar. 1976. ISSN 0362-5915. doi: 10.1145/320434.320440. URL http://doi.acm.org/10.1145/320434.320440.

[7] D. Ferraiolo, R. Sandhu, S. Gavrila, D. Kuhn, and R. Chandramouli. Proposed NIST standard for role-based access control. *ACM Transactions on Information and System Security (TISSEC)*, 4(3):224–274, 2001.

[8] D. Ferraiolo, R. Kuhn, and R. Sandhu. RBAC standard rationale: Comments on "Comments on a Critique of the ANSI Standard on Role-Based Access Control". *Security & Privacy, IEEE*, 5(6):51–53, 2007.

[9] T. L. Hinrichs, W. C. Garrison III, A. J. Lee, S. Saunders, and J. C. Mitchell. TBA : A Hybrid of Logic and Extensional Access Control Systems. In G. Barthe, A. Datta, and S. Etalle, editors, *Formal Aspects of Security and Trust*, volume 7140 of *Lecture Notes in Computer Science*, pages 198–213. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-29419-8. doi: 10.1007/978-3-642-29420-4_13. URL http://dx.doi.org/10.1007/978-3-642-29420-4_13.

[10] A. INCITS. INCITS 359-2004. Role Based Access Control. *Role based access control*, 2004.

[11] A. Kern, A. Schaad, and J. Moffett. An administration concept for the enterprise role-based access control model. In *Proceedings of the eighth ACM symposium on Access control models and technologies*, pages 3–11. ACM, 2003. URL http://www.moffett.me.uk/jdm/pubs/AdminRBAC_SACMAT2003.pdf.

[12] N. Li, J. Byun, and E. Bertino. A critique of the ANSI standard on role-based access control. *Security & Privacy, IEEE*, 5(6):41–49, 2007.

[13] M. Nyanchama and S. Osborn. The role graph model and conflict of interest. *ACM Transactions on Information and System Security (TISSEC)*, 2(1):3–33, 1999. URL http://www.matunda.org/wp-content/uploads/2006/12/p3-nyanchama1.pdf.

[14] S. Oh and S. Park. Task–role-based access control model. *Information Systems*, 28(6):533–562, 2003.

[15] O. Ore. Galois connexions. *Transactions of the American Mathematical Society*, pages 493–513, 1944.

[16] J. H. Saltzer and M. D. Schroeder. The protection of information in computer systems. *Proceedings of the IEEE*, 63 (9):1278–1308, 1975.

[17] R. Sandhu. Role activation hierarchies. In *Proceedings of the third ACM workshop on Role-based access control*, pages 33–40. ACM, 1998.

[18] R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996. ISSN 0018-9162. doi: 10.1109/2.485845.

[19] R. Sandhu, D. Ferraiolo, and R. Kuhn. The NIST model for role-based access control: towards a unified standard. In *Proceedings of the fifth ACM workshop on Role-based access control*, pages 47–63, 2000.

[20] P. Smith. The Galois connection between syntax and semantics. *University of Cambridge*, 2010. URL http://logicmatters.net/resources/pdfs/Galois.pdf.

[21] M. V. Tripunitara and N. Li. Comparing the expressive power of access control models. In *Proceedings of the 11th ACM conference on Computer and communications security*, CCS '04, pages 62–71, New York, NY, USA, 2004. ACM. ISBN 1-58113-961-6. doi: 10.1145/1030083.1030093. URL http://doi.acm.org/10.1145/1030083.1030093.

[22] WikiMedia Commons. Libra. http://en.wikipedia.org/wiki/File:Libra2.jpg, January 2011. This work is in the public domain.

APPENDIX

A. PROTOTYPE IMPLEMENTATION

Neo4j is an open–source graph database that has been gaining popularity in the community. In this appendix we outline a prototype implementation of a bi-sorted role based access control engine built on top of the Neo4j graph database. The goal is to specify the entire policy as a graph and evaluate the access function purely using the graph database query language. In this way we are effectively leveraging the database engine to take care of the heavy lifting (query planning, indexing, caching, etc.)

In addition we will show how this implementation allows for clear feedback from the access control engine to the policy designer by asking the right type of queries and presenting the result back as a graph. The latter is especially useful when combining the positive and negative style of specification.

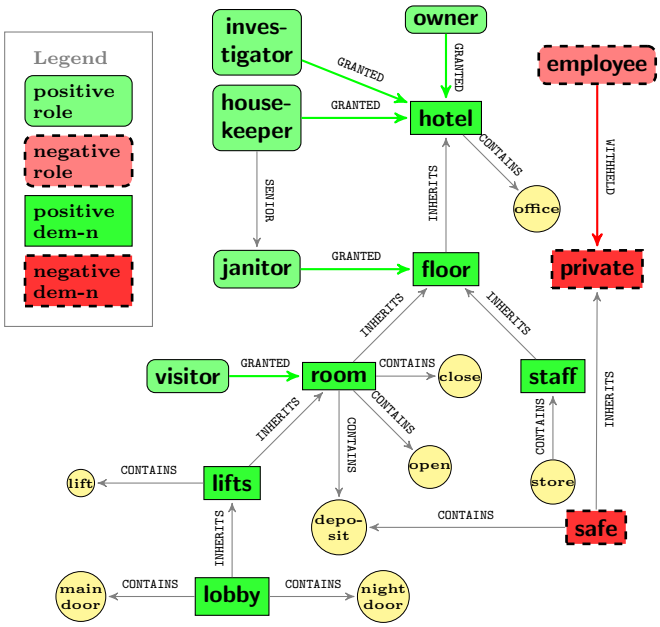
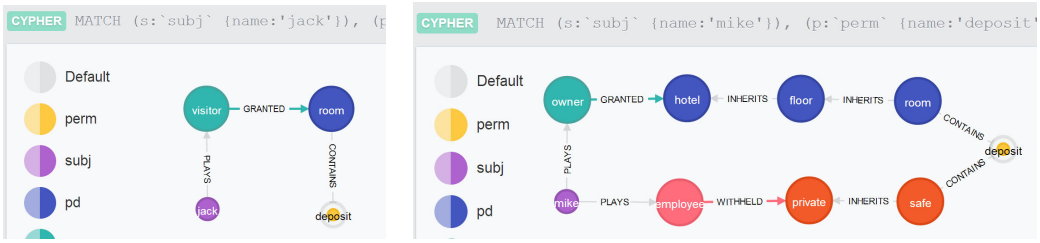


Figure 8: Roles and demarcations

We consider an anecdotal case of a small hotel. Assume that owner does not know much of Role-Based Access Control, but he understands challenges and regulations involved in such business:

- he, as owner, has to have access everywhere in the hotel;
- visitors have to have access to their rooms and all appliances there — let there be a safe, where one can deposit things in;
- there must be a janitor at each floor responsible for cleaning rooms at that floor, janitors are coordinated by a housekeeping office;



(a) `grant(jack, deposit)` (b) `grant(mike, deposit)` and `deny(mike, deposit)` combined

Figure 7: Two examples of graphical explanations of a computed AC relation

- according to city regulations, policemen, after having shown a search warrant, have to have access everywhere.

In this case the best candidate for demarcations hierarchy is the actual physical division of the hotel into rooms and floors. Having considered all the factors he comes up with a green part (left part) of the sketch given in Figure 8. We see that this part is, in fact, very similar to the regular RBAC: members of senior roles are also members of their respective juniors and permissions of junior roles are accessible for members of senior ones. One can apply the result of Theorem 1 to reconstruct the corresponding RBAC model. Nevertheless we can immediately see the benefits of the RBÄC scheme: roles of owner, housekeeper and investigator have the *same* permission set, though these roles are conceptually distinct. Quickly the positive design shows its flaw: employees can deposit things in safes, thus open and close them. Now instead of redefining the positive hierarchy, he introduces a single negative role "employee" and a single negative demarcation "safe" which include exactly one permission. Withdrawal of access rights to the safe demarcation from employees fixes the flaw. Now Figure 8 depicts the complete AC design (note the similarities to Figure 4).

We have implemented the hotel AC design with some floors and rooms in graph database Neo4j. There are two types of entities in Neo4j: nodes and relationships. The former are usually written in lower case letters surrounded with round brackets, e.g. `(node)`, relationships are usually written in capitals surrounded with square brackets, e.g. `[:RELATIONSHIP]`. Two nodes can be connected with such a relationship in a directional manner. Cypher query language of Neo4j allows to condition a length of a relation path between two nodes, e.g. `[:RELATIONSHIP*1..20]`. These features will allow us to compute AC decisions in a rather simple way.

In our example we assigned `mike` to `owner` and, thus, `employee` roles and `jack` to visitor role in some room. To find out if a subject can carry out some permission, simple Cypher queries, Listings 1 and 2, must be executed. Placeholders `subject` and `permission` need be replaced with actual subject and permission names. Logically, access is granted if grant *is not* empty and deny *is* empty, denied — in all other cases. It is possible to combine the last two queries, but we preferred to state them separately for the sake of clarity. The grant query requires the db engine to first match a certain subject and permission we are interested in. Second, a match for a role directly assigned to the given subject is looked for. Next we try to trace (via `[:SENIOR]` relation to some junior roles) a role which is directly granted access to a demarcation which, in turn, inherits (via `[:INHERITS]` relation to some junior demarcations) the given permission. These results combined form a (non-unique) path `grant` from the subject to permission.

```
MATCH (s:subj {name:'subject'}),
      (p:perm {name:'permission'}),
grant=
    (s)-[:PLAYS]->(PosRole:pr)-[:SENIOR*0..]->
    (somePosRole:pr)-[:GRANTED]->
    (somePosDemn:pd)<-[:INHERITS*0..]-
    (PosDemn:pd)-[:CONTAINS]->(p)
RETURN grant
```

Listing 1: Grant query

We relax the condition on the number of relations in-between the nodes and allow it to be absent. Similar procedure holds when a (non-unique) path `deny` is looked for.

```
MATCH (s:subj {name:'subject'}),
      (p:perm {name:'permission'}),
deny=
   (s)-[:PLAYS]->(NegRole:nr)-[:SENIOR*0..]->
   (someNegRole:nr)-[:WITHHELD]->
   (someNegDemn:nd)<-[:INHERITS*0..]-
   (NegDemn:nd)-[:CONTAINS]->(p)
RETURN deny
```

Listing 2: Deny query

Assume `jack` would like to `deposit` some personal belongings in a safe, then `grant(jack, deposit)` yields a path given in Figure 7a and `deny(jack, deposit)` expectedly produces an empty path. It is more interesting to see if `mike` can exercise the permission, for that we execute `grant(mike, deposit)` and `deny(mike, deposit)`, the result of combination of the two outputs is given in Figure 7b. We see that the `mike`'s granted `deposit` right is cancelled with his membership in the negative `employee` role.

In a practical setting it is very important to provide high quality feedback (*explanations*) to a policy designer. Arguably this is an even more fundamental requirement than expressivity of the AC language. Our framework has excellent properties in this respect. As an example take the case outlined above. Imagine a scenario where Mike wants to deposit something in an hotel room safe. Being precluded to do so Mike calls the security officer responsible complaining that he cannot access the safe. The security officer asks the AC system to explain to him the access request (`mike`, `deposit`), the result is shown graphically in Figure 7b. Following the blue path it is clear that Mike in principle has access to the safe by virtue of holding the *owner* role. However,

following the red path it becomes clear that this privilege is withdrawn because as an employee he has no access to private belongings of hotel guests.