Access Policy Design Supported by FCA Methods*

Frithjof Dau and Martin Knechtel

SAP AG, SAP Research CEC Dresden, Germany
{frithjof.dau, martin.knechtel}@sap.com

Abstract. Role Based Access Control (RBAC) is a methodology for providing users in an IT system specific permissions like *write* or *read* to users. It abstracts from specific users and binds permissions to user roles. Similarly, one can abstract from specific documents and bind permission to document types.
In this paper, we apply Description Logics (DLs) to formalize RBAC. We provide a thorough discussion on different possible interpretations of RBAC matrices and how DLs can be used to capture the RBAC constraints. We show moreover that with DLs, we can express more intended constraints than it can be done in the common RBAC approach, thus proving the benefit of using DLs in the RBAC setting. For deriving additional constraints, we introduce a strict methodology, based on attribute exploration method known from Formal Concept Analysis. The attribute exploration allows to systematically finding unintended implications and to deriving constraints and making them explicit. Finally, we apply our approach to a real-life example.

1 Introduction

1.1 Access control matrix, RBAC, Description Logics

An access control matrix M, first introduced by Lampson [1], is an abstract formal computer security model which consists of a set of objects O, subjects S and actions A. Each matrix row represents a subject and each column represents an object. Each matrix element $M[s, o] \subseteq A$ is the set of actions which a subject $s \in S$ is allowed to perform on object $o \in O$. For any type of access control system it can model the static access permissions, ignoring further definitions of a policy like rules and dynamic behavior in time. One type of access control system is Role Based Access Control (RBAC) [2], which abstracts from specific users and binds permissions to user roles. The permission set of a specific user is the union of all permissions of the roles he is assigned to. Flat RBAC comprises a set of users U, a set of roles R and a set of permissions P. Users are assigned to roles via a relation $UA \subseteq U \times R$, and permissions are assigned to roles via

* This research was funded by the German Federal Ministry of Economics and Technology under the promotional reference 01MQ07012 and the German Federal Ministry of Education and Research under grant number 01IA08001A.

a relation $PA \subseteq R \times P$. One extension to this simple model is Hierarchical RBAC, which introduces a hierarchy of user roles for permission inheritance. The partial order $\geqslant_R \subseteq R \times R$ defines the role dominance relation. If a senior role $r_s \in R$ dominates a junior role $r_j \in R$, it inherits all permission from it (i.e. $\forall p \in P : (r_j, p) \in PA \wedge (r_s, r_j) \in \geqslant_R \rightarrow (r_s, p) \in PA$).

The relationship between RBAC and other access control systems which can be modeled with the access control matrix has been elaborated in [3]. For our paper we straightforwardly interpret the set of user roles as the set of subjects ($S = R$) and we define permissions as a set of tuples of action and object ($P \subseteq A \times O$). We call this an *RBAC matrix*. An *RBAC policy* can not completely be described by an RBAC matrix, since it contains further constraints, e.g. rules, dynamic behavior in time, user role hierarchy, implications between the allowed actions etc. Objects do not need to be individuals but could also be abstract groups. As an example for the RBAC matrix, each row represents a user role and each column a document type.

Description Logic (DL) [4] systems are formal knowledge representation and reasoning systems which provide inference services that deduce implicit knowledge from the explicitly represented knowledge. For these inference services to be feasible the underlying inference problems must at least be decidable, since DL is a decidable fragment of First Order Logic this is provided. Some proposals are available to model an RBAC policy with a DL ontology, in order to reduce authorization decision to standard reasoning services. Some of these approaches contained modeling flaws which we discussed in [5] and [6].

1.2 Our contributions

Our paper discusses how FCA can be applied in order to provide services to a security policy designer. In our approach, a role-based access control matrix is formalized as triadic formal context $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}} := (\mathbf{R}, \mathbf{D}, \mathbf{P}, I)$, with a set $\mathbf{R}$ of role names, a set $\mathbf{D}$ of document type names and a set $\mathbf{P}$ of permission names.

Although it is quite straightforward to use an access control matrix as a model for RBAC, the interpretation of the matrix is not a priori clear. The paper contains a *discussion of three interpretations of an RBAC matrix*.

Up to now the DL modeling was done with ad hoc approaches. In [6] we discussed a flawed approach and proposed a reworked version. In this paper, we show how in each of the three possible interpretations, the information contained in the RBAC matrix is correctly modeled by DL general concept inclusions (GCIs). The used DL is $\mathcal{ALERO\mathcal{I}}$ which is a subset of $\mathcal{SRO\mathcal{I}Q}$, the basis for the coming W3C OWL 2 standard. This DL is required to simulate the concept product expressions $\mathsf{R}^{\mathcal{I}} \times \mathsf{D}^{\mathcal{I}} \subseteq \mathsf{P}^{\mathcal{I}}$ and $(\mathsf{R}^{\mathcal{I}} \times \mathsf{D}^{\mathcal{I}}) \cap \mathsf{P}^{\mathcal{I}} = \emptyset$.

Using DLs, it will turn out that we can model additional constraints which are intended by the RBAC engineer, but which cannot be modd in role-based access control matrix alone. For example for a review process, it is not allowed that the same person who writes a document also approves it. The inclusion of axiom $mayWrite \sqcap mayApprove \sqsubseteq \bot$ in the DL model defines that both permissions are disjoint. The DL model allows *consistency checks of the RBAC*

policy with given additional restrictions. Both the higher expressiveness of a DL based modelling approach and the consistency check clearly show the benefit of using DLs for RBAC.

Generally, one can say that ontology editors provide reasoning facilities, where for example the consistency of an DL knowledge base can be checked. Roughly speaking, ontology editors support checking the *soundness* of a DL knowledge base. In this paper, we do not only target the soundness of the DL formalization of an RBAC matrix, but also the *completeness* (compare to [7]). We introduce a *strict methodology*, based on the attribute exploration method of FCA, for deriving additional constraints in RBAC setting. Our methodology derives such constraints not explicitly contained in the RBAC matrix in a computer supported dialog with the RBAC engineer. This helps the engineer to create the policy as DL model based on the matrix.

The paper is structured as follows: In Sec. 2, all relevant notions are formally defined, and the running example we will use is introduced. Moreover, in this section the tree possible interpretations of an RBAC matrix are discussed. In Sec. 3, we show how the information of an RBAC matrix can be expressed by means of DL GCIs. In Sec. 4, we thoroughly discuss how attribute exploration can be used in order to obtain additional constraints from the RBAC matrix, and how these constraints are then modeled with DL GCIs. In Sec. 4, we apply our approach to a real-life example. Finally, in Sec. 6 we summarize this paper and discuss future research.

2 Basic Definitions

In this section, all relevant notions which will be used in this paper are formally defined, and our working example is introduced.

Vocabulary: As already mentioned, our starting point is a three-dimensional matrix, where the three dimensions are the *roles*, *document types* and *permissions*. In order not to mix up user roles and DL roles, with "role" we always refer to a user role, whereas we use the OWL terminology "property" for a DL role. In our ongoing formalization, both roles and document types will be modeled as concept names of a (appropriately chosen) DL, and each permission will be modeled as a property between roles and document types. That is, we consider a DL vocabulary which consists of a set $\mathbf{R}$ of role names, a set $\mathbf{D}$ of document type names, and of a set $\mathbf{P}$ of permission names. The vocabulary of these names will be denoted $\mathbf{V}$. We will use a working example with specific roles, document types and permissions. We consider the permissions mayApprove, mayWrite and mayOpen, which are abbreviated by MA, MW and MO, respectively. The document types are user manual, marketing document, customer contract document, term of use document, installation guide, external technical interface document, design document and rating entry, abbreviated by UM, MD, CCD, ToUD, IG, ETID, DD, RE. The roles are marketplace visitor, service consumer, software development engineer, service vendor, legal department employee, service provider, marketing employee, technical editor and customer service employee, abbreviated by MV, SC, SDE, SV,

LDE, SP, ME, TE and CSE. This example stems from the research project Theseus/Processus from a scenario where documents describe aspects of services offered in the Internet of Services. The documents are accessible by different roles with different permissions.

Formal Contexts: The three-dimensional matrix of roles, document types and permissions is formalized as a triadic formal context $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}} := (\mathbf{R}, \mathbf{D}, \mathbf{P}, I)$. The example we use in this paper is provided in Tab. 1.

	mayOpen								mayWrite								mayApprove							
	UM	MD	CCD	ToUD	IG	ETID	DD	RE	UM	MD	CCD	ToUD	IG	ETID	DD	RE	UM	MD	CCD	ToUD	IG	ETID	DD	RE
MV		×		×				×																
SC	×	×	×	×		×		×								×			×	×				
SDE	×	×		×	×	×	×	×	×				×	×	×			×						
SV	×	×	×	×	×	×	×	×									×	×	×	×	×	×	×	
LDE	×	×	×	×	×	×	×	×			×	×												
SP	×	×		×	×	×		×																
ME	×	×		×	×	×	×	×		×														
TE	×	×		×	×	×	×	×	×				×	×	×									
CSE	×	×	×	×	×	×	×	×			×													

Table 1. Our example RBAC matrix

Our aim is to conduct an attribute exploration in order to explore dependencies between different roles, different document types, or different permissions. As attribute exploration is applied to dyadic contexts, we have do derive such contexts from the given triadic context. This can be done in several ways.

1. First, we can consider "slices" of the triadic context. For our goal, it is most useful to consider the "slice" for each $\mathsf{P} \in \mathbf{P}$. That is, for a given $\mathsf{P} \in \mathbf{P}$, we consider $\mathbb{K}^{\mathsf{P}}_{\mathbf{R},\mathbf{D}} := (\mathbf{R}, \mathbf{D}, I^{\mathsf{P}})$, where $(\mathsf{R}, \mathsf{D}) \in I^{\mathsf{P}} :\Leftrightarrow (\mathsf{R}, \mathsf{D}, \mathsf{P}) \in I$.
2. Next, we can consider the dyadic contexts, where the set of attributes is one of the sets $\mathbf{R}$, $\mathbf{D}$, $\mathbf{P}$, and the set of objects is the cross-product of the remaining two sets. E.g. we can consider the context $\mathbb{K}_{\mathbf{R}\times\mathbf{P},\mathbf{D}} := (\mathbf{R} \times \mathbf{P}, \mathbf{D}, I^{\mathbf{R}\times\mathbf{P},\mathbf{D}})$ with $((\mathsf{R}, \mathsf{P}), \mathsf{D}) \in I^{\mathbf{R}\times\mathbf{P},\mathbf{D}} \Leftrightarrow (\mathsf{R}, \mathsf{D}, \mathsf{P}) \in I$. This is a straight-forward transformation. To simplify notations, we will denote the incidence relation again by I, thus writing $(\mathbf{R} \times \mathbf{D}, \mathbf{P}, I)$. We can construct six dyadic contexts this way, namely $\mathbb{K}_{\mathbf{R}\times\mathbf{D},\mathbf{P}}$, $\mathbb{K}_{\mathbf{P}\times\mathbf{R},\mathbf{D}}$, $\mathbb{K}_{\mathbf{D}\times\mathbf{P},\mathbf{R}}$ and the respective named variants with identical cross table $\mathbb{K}_{\mathbf{D}\times\mathbf{R},\mathbf{P}}$, $\mathbb{K}_{\mathbf{R}\times\mathbf{P},\mathbf{D}}$, $\mathbb{K}_{\mathbf{P}\times\mathbf{D},\mathbf{R}}$.
3. For a given context $\mathbb{K} := (G, M, I)$, when attribute exploration is conducted, sometimes it is sensible to add an additional attribute $\perp$ (which satisfies $\neg\exists g \in G : (g, \perp) \in I$) to M. We use $\mathbb{K}_{\perp} := (G, M \cup \{\perp\}, I)$ to denote this context (again, we simply 'reuse' the symbol 'I' for the incidence relation).

In our example no agent will be allowed to write and approve the same document, thus $\mathsf{mayApprove} \wedge \mathsf{mayWrite} \to \bot$.

As each of the formal context only deals with *names* for roles, document types, and permissions, but not with instances of these names (in some DL interpretations, see below), all these formal contexts are called $\mathcal{T}$-context.

Interpretations: The DL-interpretations for RBAC matrices are straightforwardly defined: For our setting, a DL-interpretation for $\mathbf{V}$ is a pair $(\Delta, \cdot^{\mathcal{I}})$ with a non-empty universe (of discourse) Δ and an interpretation function $\cdot^{\mathcal{I}}$ which satisfies:

- $\mathsf{R}^{\mathcal{I}} \subseteq \Delta$ for each $\mathsf{R} \in \mathbf{R}$. Moreover, we set $\mathbf{R}^{\mathcal{I}} := \bigcup_{\mathsf{R} \in \mathbf{R}} \mathsf{R}^{\mathcal{I}}$. The elements $r \in \mathbf{R}^{\mathcal{I}}$ are called agents.
- $\mathsf{D}^{\mathcal{I}} \subseteq \Delta$ for each $\mathsf{D} \in \mathbf{D}$. Moreover, we set $\mathbf{D}^{\mathcal{I}} := \bigcup_{\mathsf{D} \in \mathbf{D}} \mathsf{D}^{\mathcal{I}}$.
- $\mathsf{P}^{\mathcal{I}} \subseteq \mathbf{R}^{\mathcal{I}} \times \mathbf{D}^{\mathcal{I}}$ for each $\mathsf{P} \in \mathbf{P}$
- $\mathbf{R}^{\mathcal{I}} \cap \mathbf{D}^{\mathcal{I}} = \emptyset$ (nothing is both agent and document)
- $\mathbf{R}^{\mathcal{I}} \cup \mathbf{D}^{\mathcal{I}} = \Delta$ (everything is either agent or document)

Note that the first two conditions are standard conditions for DL interpretations, whereas the last 3 condition are additional constraints.

Permissive, Prohibitive and Strict Interpretations: As each formal object and attribute of $(\mathbf{R}, \mathbf{D}, \mathbf{P}, I)$ stands in fact for a whole class of agents resp. documents, it is not a priori clear what the semantics of the incidence relation I with respect to an interpretation $(\Delta, \cdot^{\mathcal{I}})$ is. So we have to clarify the meaning of I. First we might assume that a relationship $(\mathsf{R}, \mathsf{D}, \mathsf{P}) \in I$ means that *each* agent $r \in \mathsf{R}^{\mathcal{I}}$ has the permission $\mathsf{P}^{\mathcal{I}}$ for *each* document $d \in \mathsf{D}^{\mathcal{I}}$. So a cross in the cross-table of the context $(\mathbf{R}, \mathbf{D}, I^{\mathsf{P}})$ grants permissions to agents on documents, and we can read from the context which permissions are *at least* granted to agents. Vice versa, we might assume that a missing relationship $(\mathsf{R}, \mathsf{D}, \mathsf{P}) \notin I$ means that *no* agent $r \in \mathsf{R}^{\mathcal{I}}$ has the permission $\mathsf{P}^{\mathcal{I}}$ for *any* document $d \in \mathsf{D}^{\mathcal{I}}$. So a missing cross in the cross-table of the context $(\mathbf{R}, \mathbf{D}, I^{\mathsf{P}})$ prohibits that permissions are granted to agents, and we can read from the context which permissions are *at most* granted to agents. And finally, we could of course assume that both conditions hold. That is, we can read from the context which permissions are *precisely* granted to agents.

interpretation	cross	no cross
strict	permission for all individuals	prohibition for all individuals
permissive	permission for all individuals	permission for some individuals
prohibitive	permission for some individuals	prohibition for all individuals

Table 2. Variants how to interpret a cross in the context

These three understandings lead to the notion of permissive, prohibitive and strict interpretations (with respect to the formal context) summarized in Tab. 2. They are formally defined as follows:

- An interpretation $(\Delta, \cdot^{\mathcal{I}})$ is called permissive (with respect to $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$), and we write $(\Delta, \cdot^{\mathcal{I}}) \models_{+} (\mathbf{R}, \mathbf{D}, \mathbf{P}, I)$, iff. for all role names $\mathsf{R} \in \mathbf{R}$, all document type names $\mathsf{D} \in \mathbf{D}$ all permission names $\mathsf{P} \in \mathbf{P}$ we have:

$$(\mathsf{R}, \mathsf{D}, \mathsf{P}) \in I \quad \Longrightarrow \quad \forall r \in \mathsf{R}^{\mathcal{I}}\ \forall d \in \mathsf{D}^{\mathcal{I}} : (r, d) \in \mathsf{P}^{\mathcal{I}}$$

  In other words, if $(\mathsf{R}, \mathsf{D}, \mathsf{P}) \in I$, we have $\mathsf{R}^{\mathcal{I}} \times D^{\mathcal{I}} \subseteq \mathsf{P}^{\mathcal{I}}$.
- An interpretation $(\Delta, \cdot^{\mathcal{I}})$ is called prohibitive (with respect to $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$), and we write $(\Delta, \cdot^{\mathcal{I}}) \models_{-} (\mathbf{R}, \mathbf{D}, \mathbf{P}, I)$, iff. for all role names $\mathsf{R} \in \mathbf{R}$, all document type names $\mathsf{D} \in \mathbf{D}$ all permission names $\mathsf{P} \in \mathbf{P}$ we have:

$$(\mathsf{R}, \mathsf{D}, \mathsf{P}) \notin I \quad \Longrightarrow \quad \forall r \in \mathsf{R}^{\mathcal{I}}\ \forall d \in \mathsf{D}^{\mathcal{I}} : (r, d) \notin \mathsf{P}^{\mathcal{I}}$$

  In other words, if $(\mathsf{R}, \mathsf{D}, \mathsf{P}) \notin I$, we have $(\mathsf{R}^{\mathcal{I}} \times D^{\mathcal{I}}) \cap \mathsf{P}^{\mathcal{I}} = \emptyset$.
- An interpretation $(\Delta, \cdot^{\mathcal{I}})$ is called strict (with respect to $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$), iff. it is both permissive and prohibitive.

We say that we use the permissive approach (prohibitive approach, strict approach), if we assume that each interpretation is permissive (prohibitive, strict).

Instantiations of Contexts: As already said in the introduction, it will turn out that for running attribute exploration on the context, it is reasonable not to consider the $\mathcal{I}$-context, but contexts where on the side of the objects, roles are replaced by "real" users resp. document types are replaced by "real" documents. Essentially, instantiations of a context contain at least all rows of the given context, and there might be more rows, but these additional rows must be extensions of rows in the given context. These contexts are now introduced.

Let one of the contexts $\mathbb{K}^{\mathsf{P}}_{\mathbf{R},\mathbf{D}} := (\mathbf{R}, \mathbf{D}, I^{\mathsf{P}})$ $(\mathsf{P} \in \mathbf{P})$ be given. An instantiation of $\mathbb{K}^{\mathsf{P}}_{\mathbf{R},\mathbf{D}}$ is a context $(R, \mathbf{D}, J^{\mathsf{P}})$, where R is a set of agents such that

- $\forall \mathsf{R} \in \mathbf{R}\ \exists r \in R\ \forall \mathsf{D} \in \mathbf{D} : (\mathsf{R}, \mathsf{D}) \in I^{\mathsf{P}} \Leftrightarrow (r, \mathsf{D}) \in J^{\mathsf{P}}$
- $\forall r \in R\ \exists \mathsf{R} \in \mathbf{R}\ \forall \mathsf{D} \in \mathbf{D} : (\mathsf{R}, \mathsf{D}) \in I^{\mathsf{P}} \Rightarrow (r, \mathsf{D}) \in J^{\mathsf{P}}$

Such a context will be denoted $\mathbb{K}^{\mathsf{P}}_{R,\mathbf{D}}$. We define similarly the instantiations $\mathbb{K}_{R\times\mathbf{P},\mathbf{D}}$ of $\mathbb{K}_{\mathbf{R}\times\mathbf{P},\mathbf{D}}$, and $\mathbb{K}_{\mathbf{P}\times R,\mathbf{D}}$ of $\mathbb{K}_{\mathbf{P}\times\mathbf{R},\mathbf{D}}$ (where again the role names are replaced by agents), as well as the instantiations $\mathbb{K}^{\mathsf{P}}_{D,\mathbf{R}}$ of $\mathbb{K}^{\mathsf{P}}_{\mathbf{D},\mathbf{R}}$ $(\mathsf{P} \in \mathbf{P})$, $\mathbb{K}_{D\times\mathbf{P},\mathbf{R}}$ of $\mathbb{K}_{\mathbf{D}\times\mathbf{P},\mathbf{R}}$, and $\mathbb{K}_{\mathbf{P}\times D,\mathbf{R}}$ of $\mathbb{K}_{\mathbf{P}\times\mathbf{D},\mathbf{R}}$ (where now the document type names are replaced by documents).

Instantiations of the contexts where the permissions are the attributes, i.e. instantiations $\mathbb{K}_{D\times R,\mathbf{P}}$ of $\mathbb{K}_{\mathbf{D}\times\mathbf{R},\mathbf{P}}$ (resp. $\mathbb{K}_{R\times D,\mathbf{P}}$ of $\mathbb{K}_{\mathbf{R}\times\mathbf{D},\mathbf{P}}$) are defined similarly (where on the side of the objects, both document type names and role names are replaced by "real" documents and "real" agents, respectively).

An example for an instantiation of $\mathbb{K}^{\mathsf{mayWrite}}_{\mathbf{R},\mathbf{D}}$ is given in Tab. 3.

	UM	MD	CCD	ToUD	IG	ETID	DD	RE
MV								
SC								×
SDE	×				×	×	×	
SV								
LDE			×	×				
SP								
ME		×						
TE	×				×	×	×	
CSE			×					

	UM	MD	CCD	ToUD	IG	ETID	DD	RE
$agent_1$								
$agent_2$								×
$agent_3$	×				×	×	×	
$agent_4$								
$agent_5$			×	×				
$agent_6$								
$agent_7$		×						
$agent_8$	×				×	×	×	
$agent_9$			×					
$agent_{10}$			×	×	×			
$agent_{11}$	×		×		×	×	×	×
$agent_{12}$		×	×					

Table 3. The context $\mathbb{K}^{\mathsf{mayWrite}}_{\mathsf{R},\mathsf{D}}$ and one possible instantiation.

3 Expressing the Cross-Table by GCIs

In this section, we scrutinize how the information of the context $\mathbb{K}_{\mathsf{R},\mathsf{D},\mathsf{P}}$ can be expressed by means of DLs. Besides the standard DL quantifications $\exists R.C$ (the set of entities which stand in relation R *to at least one* instance of C) and $\forall R.C$ (the set of entities which stand in relation R *only to* instances of C), we will use the non-standard constructor $\forall C.R$ (the set of entities which stand in relation R *to all* instances of C). This constructor can be expressed by means of negation of relations, as $\forall C.R$ is equivalent to $\forall \neg R.\neg C$ (see [8] for a thorough discussion of the constructor). Adding it to $\mathcal{ALC}$ still yields a decidable DL, but as this constructor is certainly non-standard, is it not supported by common DL reasoners.

For the permissive approach, we have to capture the condition $\mathsf{R}^{\mathcal{I}} \times D^{\mathcal{I}} \subseteq \mathsf{P}^{\mathcal{I}}$. The left expression is a *concept product*. It can not be expressed in $\mathcal{SHOIN}(\mathrm{D})$, which is the underlying DL of OWL DL. In OWL 2.0, there does not exist a native language construct for the concept product, but Krötzsch, Rudolph, Hitzler provide in [9] a workaround to express it in OWL 2.0. Using the constructor $\forall C.R$, the condition $\mathsf{R}^{\mathcal{I}} \times D^{\mathcal{I}} \subseteq \mathsf{P}^{\mathcal{I}}$ can be expressed with the GCIs

$$\mathsf{R} \sqsubseteq \forall \mathsf{D}.\mathsf{P} \quad (\text{i.e. } \mathsf{R} \sqsubseteq \forall \neg \mathsf{P}.\neg \mathsf{D}) \qquad \text{and} \qquad \mathsf{D} \sqsubseteq \forall \mathsf{R}.\mathsf{P}^{-1} \quad (\text{i.e. } \mathsf{D} \sqsubseteq \forall \neg \mathsf{P}^{-1}.\neg \mathsf{R})$$

For the prohibitive approach, the condition $(\mathsf{R}^{\mathcal{I}} \times \mathsf{D}^{\mathcal{I}}) \cap \mathsf{P}^{\mathcal{I}} = \emptyset$ has to be captured. This can be expressed by the two GCIs

$$\mathsf{R} \sqsubseteq \forall \mathsf{P}.\neg \mathsf{D} \qquad \text{and} \qquad \mathsf{D} \sqsubseteq \forall \mathsf{P}^{-1}.\neg \mathsf{R}$$

Note that this condition is precisely the condition for the permissive approach, when we replace each permission P by its complement $\neg \mathsf{P}$. This duality principle will be discussed in the next section.

If we knew that $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$ is correct, and if we know which type of approach (permissive, prohibitive, strict) we use, then we can describe the information of $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$ by DL GCIs. We first set $\mathsf{R}_{\text{all}} := \bigsqcup_{\mathsf{R}\in\mathbf{R}} \mathsf{R}$ and $\mathsf{D}_{\text{all}} := \bigsqcup_{\mathsf{D}\in\mathbf{D}} \mathsf{D}$. Now we define the following knowledge base:

$$KB_0 := \{\mathsf{R}_{\text{all}} \sqsubseteq \forall \mathsf{P}.\mathsf{D}_{\text{all}}\,, \mathsf{D}_{\text{all}} \sqsubseteq \forall \mathsf{P}^{-1}.\mathsf{R}_{\text{all}} \mid \mathsf{P} \in \mathbf{P}\} \cup \{\mathsf{R}_{\text{all}} \sqsubseteq \neg \mathsf{D}_{\text{all}}\} \cup \{\mathsf{R}_{\text{all}} \sqcup \mathsf{D}_{\text{all}} \equiv \top\}$$

Obviously, a general DL-interpretation $(\Delta, \cdot^{\mathcal{I}})$ is a DL-interpretation of $\mathbf{V}$ iff. it satisfies KB_0. According to the chosen approach, we can now capture the information of $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$ as follows:

$$\begin{aligned} KB_+ &:= KB_0 \cup \{\mathsf{R} \sqsubseteq \forall \neg \mathsf{P}.\neg \mathsf{D}\,,\ \mathsf{D} \sqsubseteq \forall \neg \mathsf{P}^{-1}.\neg \mathsf{R} \mid (\mathsf{R},\mathsf{D},\mathsf{P}) \in I\} \\ KB_- &:= KB_0 \cup \{\mathsf{R} \sqsubseteq \forall \mathsf{P}.\neg \mathsf{D}\,,\ \mathsf{D} \sqsubseteq \forall \mathsf{P}^{-1}.\neg \mathsf{R} \mid (\mathsf{R},\mathsf{D},\mathsf{P}) \notin I\} \\ KB_\pm &:= KB_+ \cup KB_- \end{aligned}$$

Again, a DL-interpretation is obviously an permissive (prohibitive, strict) interpretation of $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$, if it satisfies KB_+ (KB_-, $KB_\pm$).

4 Using Attribute Exploration for RBAC matrices

In this section, we discuss how attribute exploration can be utilized in order to create a DL knowledge base which captures as best as possible the dependencies between roles, documents, and permissions. It is crucial which approach (permissive, prohibitive, strict) we use, thus we first elaborate the differences between these approaches with respect to attribute exploration. In the second and third part of this section, we go into the details of an attribute exploration for instantiations of contexts in the permissive approach.

4.1 General discussion

We first compare the permissive and the prohibitive approach. In the permissive approach, the crosses in a cross-table carry information, whereas missing crosses are not informative. In the prohibitive approach, the situation is converse: Missing crosses carry information, and crosses are not informative. Missing crosses in a relation correspond to crosses in the complement of the relation. Thus if we replace in the prohibitive approach the relations $\mathsf{mayOpen}$,$\mathsf{mayWrite}$ and $\mathsf{mayApprove}$ by their complements $\mathsf{mayOpen}^c = \mathsf{mustNotOpen}$, $\mathsf{mayWrite}^c = \mathsf{mustNotWrite}$, $\mathsf{mayApprove}^c = \mathsf{mustNotApprove}$, we have a situation similar to the permissive approach. That is, we have the following duality principle: Any account to the permissive approach can be turned into an account to the prohibitive approach (and vice versa) by replacing each permission by its complement.[1] For this reason, we do not target the prohibitive approach in this paper.

[1] But keep in mind that switching between the permissive and prohibitive approach requires changing the underlying DL-language, including the need for non-standard constructors in the permissive approach.

We assume that the set of role names, document type names, and permission names is fixed. Conducting an attribute exploration on one of the $\mathcal{T}$-contexts seems for this reason to some extent pointless, as we cannot add new objects (counterexamples for implications which do not hold). We can still use attribute exploration in order to check that the information in $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$ is *correct*, but this idea does not tap the full potential of attribute exploration and will not be carried out in this paper (we assume that the matrix $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$ is correct). But notice that this check for correctness would have avoided the inconsistency between role hierarchy, DL model and access matrix discussed in [6]. Anyhow, we emphasized that in the formal context, the formal objects (the elements of $\mathbf{R}$) and attributes (the elements of $\mathbf{D}$) stand in turn for complete classes (of agents and documents). This can be used to apply attribute exploration to RBAC matrices. Assume we stick to the permissive approach. Assume moreover that we consider a permissive interpretation $(\Delta, \cdot^{\mathcal{I}})$ with respect to $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$. Then for a given permission $\mathsf{P} \in \mathbf{P}$, agent $r \in \mathsf{R}^{\mathcal{I}}$ for a role $\mathsf{R} \in \mathbf{R}$, and document $d \in \mathsf{D}^{\mathcal{I}}$ for a document type $\mathsf{D} \in \mathbf{D}$, we might have that r has permission P to d (i.e., $(r, d) \in \mathsf{P}^{\mathcal{I}}$), though we do not have $(\mathsf{R}, \mathsf{D}, \mathsf{P}) \in I$. That is, it is sensible to run an attribute exploration on the *instantiations* of the $\mathcal{T}$-contexts. As we will see in the next section, with attribute exploration we can in fact infer constraints for the dependencies between roles, documents and permissions which are not captured by $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$.

In the strict approach, the situation is different. If we consider a strict interpretation $(\Delta, \cdot^{\mathcal{I}})$ with respect to $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$, then for a given permission $\mathsf{P} \in \mathbf{P}$, agent $r \in \mathsf{R}^{\mathcal{I}}$ and document $d \in \mathsf{D}^{\mathcal{I}}$, we have $(r, d) \in \mathsf{P}^{\mathcal{I}} \Leftrightarrow (\mathsf{R}, \mathsf{D}, \mathsf{P}) \in I$. That is, based on the given assumption that the sets of roles, documents and permissions are fixed, all possible constraints for the dependencies between these entities are already captured by $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$. This observation has two consequences: First, no DL formalization of the strict approach can extend the information of $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$, i.e., a DL formalization of $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$ is somewhat pointless. Second, the instantiations of $\mathcal{T}$-context are nothing but the $\mathcal{T}$-context themselves (instantiations might duplicate some rows, but this is of course of no interest), thus conduction attribute exploration in the strict approach is pointless as well.

To summarize: As the permissive and prohibitive approach are mutually dual, and as addressing the strict approach with DLs or attribute exploration is pointless, it is sufficient that we here address only the permissive approach.

4.2 Attribute Exploration for Instantiations of $\mathcal{T}$-contexts

In the last part we argued why we will run attribute exploration on instantiations of $\mathcal{T}$-contexts. Before doing so, we first have to discuss how implications in $\mathcal{T}$-contexts and their instantiations are read, and then we will scrutinize some peculiarities for applying attribute exploration in our setting. In fact, due to the fact that the objects and attributes of $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$ stand for whole classes, the existing approaches for conducting attribute explorations on triadic contexts (e.g, [10]) cannot be applied to our framework.

Reading Implications We consider the two contexts of Tab. 3. In both contexts, term of use document→customer contract document holds. For the $\mathcal{T}$-context $\mathbb{K}^{\mathsf{mayWrite}}_{\mathbf{R},\mathbf{D}}$, the objects are classes, thus this implication is read as follows:

> **$\mathcal{T}$-reading:** For each role we have that whenever every agent of that role may write all terms of use documents, then every agent of that role may write all customer contract documents as well.

For the instantiation of $\mathbb{K}^{\mathsf{mayWrite}}_{\mathbf{R},\mathbf{D}}$, the objects are now instances instead of classes, thus we have a different reading of the implication. It is:

> **$\mathcal{I}$-reading:** Whenever every agent may write all terms of use documents, then the agent may write all customer contract documents as well.

Implications like this cannot be read from any $\mathcal{T}$-context, thus running attribute exploration on instantiations can indeed be used to obtain new knowledge.

Please note that none of the above readings conforms to the concept inclusion term of use document$\sqsubseteq$customer contract document. This is due to in both implications we quantify over *all* term of use documents and *all* customer contract documents. For the latter reading, we now show how it is correctly translated into a GCI. The implication means that for any permissive interpretation $(\Delta, \cdot^{\mathcal{I}})$, we have that $\forall r \in \mathbf{R}^{\mathcal{I}} : (\forall d \in \mathsf{ToUD}^{\mathcal{I}} : (r,d) \in \mathsf{MW}^{\mathcal{I}} \rightarrow \forall d \in \mathsf{CCD}^{\mathcal{I}} : (r,d) \in \mathsf{MW}^{\mathcal{I}})$ holds. This condition is now transformed into a GCI as follows:

$$\begin{aligned}
&\forall r \in \mathbf{R}^{\mathcal{I}} : \Big(\forall d \in \mathsf{ToUD}^{\mathcal{I}} : (r,d) \in \mathsf{MW}^{\mathcal{I}} \rightarrow \forall d \in \mathsf{CCD}^{\mathcal{I}} : (r,d) \in \mathsf{MW}^{\mathcal{I}}\Big) \\
&\iff \forall r \in \mathbf{R}^{\mathcal{I}} : \big(r \in (\forall \mathsf{ToUD}.\mathsf{MW})^{\mathcal{I}} \rightarrow r \in (\forall \mathsf{CCD}.\mathsf{MW})^{\mathcal{I}}\big) \\
&\iff (\Delta, \cdot^{\mathcal{I}}) \models \forall \mathsf{ToUD}.\mathsf{MW} \sqsubseteq \forall \mathsf{CCD}.\mathsf{MW}
\end{aligned}$$

(we have to emphasize that the direction "$\rightarrow$" of the last equivalence is only valid if we assume that $dom(\mathsf{MW}^{\mathcal{I}}) \subseteq \mathbf{R}^{\mathcal{I}}$ holds, but we assume that out interpretation satisfies KB_0, which models this additional condition).

In general, any implication of the form $\mathsf{D}_1 \wedge \ldots \wedge \mathsf{D}_{n-1} \rightarrow \mathsf{D}_n$ in an instantiation of one of the contexts $\mathbb{K}^{\mathsf{P}}_{\mathbf{R},\mathbf{D}}$ can be translated into the following GCI:

$$\forall \mathsf{D}_1.\mathsf{P} \sqcap \ldots \sqcap \forall \mathsf{D}_{n-1}.\mathsf{P} \sqsubseteq \forall \mathsf{D}_n.\mathsf{P}$$

Similarly, any implication of the form $\mathsf{R}_1 \wedge \ldots \wedge \mathsf{R}_{n-1} \rightarrow \mathsf{R}_n$ in an instantiation of one of the contexts $\mathbb{K}^{\mathsf{P}}_{\mathbf{D},\mathbf{R}}$ can be translated into the following GCI:

$$\forall \mathsf{R}_1.\mathsf{P}^- \sqcap \ldots \sqcap \forall \mathsf{R}_{n-1}.\mathsf{P}^- \sqsubseteq \forall \mathsf{R}_n.\mathsf{P}^-$$

If we consider an instantiation of a context where the attributes of the context are neither document type names nor role names, but instead permission names, the situation is different, as now the attributes do not stand for classes of instances, but for properties between instances. In Sec. 5.1, we consider a context $\mathbb{K}_{D \times R,\mathbf{P}}$. In this context, mayWrite $\rightarrow$ mayOpen holds. The reading of this implication is

> Whenever some agent has the permission to write some document, then this agent may open this document as well.

So we see that in this case, the implication can be translated to a simple inclusion axiom between properties, namely mayWrite $\sqsubseteq$ mayOpen.

4.3 Conducting Attribute Exploration on Instantiations

We consider the instantiation of a $\mathcal{T}$-context, where we want to run attribute exploration on. Obviously, for any $\mathcal{T}$-context $\mathbb{K}$, there exists a smallest instantiation $\mathbb{K}_{\min}$, which is isomorphic to $\mathbb{K}$, and a largest instantiation $\mathbb{K}_{\max}$. The basic idea is that we start the attribute exploration with $\mathbb{K}_{\min}$, and for implications which do not hold, we add (as usual) counterexamples to the context, until we finally reach a context $\mathbb{K}_{\text{ae}}$. Anyhow, in this process, we cannot add counterexamples in an arbitrary manner, as the context $\mathbb{K}_{\text{ae}}$ we obtain must still be an instantiation. The question is how this additional constraint can be captured by attribute exploration. First of all, we trivially have the following subset relations between the implications which hold in the contexts:

$$Imp(\mathbb{K}_{\max}) \subseteq Imp(\mathbb{K}_{\text{ae}}) \subseteq Imp(\mathbb{K}_{\min})$$

So if we run an attribute exploration on $Imp(\mathbb{K}_{\min})$, we could use $Imp(\mathbb{K}_{\max})$ as a set of additional background implications. Anyhow, a closer observation yields that $Imp(\mathbb{K}_{\max})$ only contains all implications of the form $\emptyset \to m$, where m is an attribute of $\mathbb{K}_{\min}$ which applies to all objects. This can easily be seen as follows: Let $\mathbb{K}_{\min} := (O_{\min}, M, I_{\min})$, let $\mathbb{K}_{\max} := (O_{\max}, M, I_{\max})$, let $M_1 := \{m \in M \mid \forall o \in O_{\min} : (o, m) \in I_{\min}\}$ and $M_2 := M - M_1$ be the complement of M_1. First of all, we obviously have that $\emptyset \to m_1$ holds in $\mathbb{K}_{\min}$, thus in $\mathbb{K}_{\max}$ as well, for each $m_1 \in M_1$. Now let $m_2 \in M_2$. Then there exists an object $o \in O_{\max}$ with $(o, m) \in I_{\max} \Leftrightarrow m \neq m_2$ for all $m \in M$. That is, there cannot exist any (nontrivial) implication in $Imp(\mathbb{K}_{\max})$ with m_2 in its conclusion.

4.4 Choice of Instantiation Contexts for Attribute Exploration

Theoretically, we could conduct an attribute exploration on the minimal instantiation of $\mathbb{K}_{\mathbf{R}\times\mathbf{P},\mathbf{D}}$. Anyhow, we observe that any instantiation of $\mathbb{K}_{\mathbf{R}\times\mathbf{P},\mathbf{D}}$ is the subposition of instantiations of the contexts$\mathbb{K}^{\mathsf{P}}_{\mathbf{R},\mathbf{D}}$, $\mathsf{P} \in \mathbf{P}$. Generally, for any contexts $\mathbb{K}_1, \ldots, \mathbb{K}_n$ with identical attribute sets, an implication holds in each context $\mathbb{K}_1, \ldots, \mathbb{K}_n$ if and only if it holds in the subposition of these contexts. Thus if the RBAC engineer runs an attribute exploration on the minimal instantiation of all contexts $\mathbb{K}^{\mathsf{P}}_{\mathbf{R},\mathbf{D}}$, $\mathsf{P} \in \mathbf{P}$, there is no need to run an attribute exploration on the minimal instantiation of $\mathbb{K}_{\mathbf{R}\times\mathbf{P},\mathbf{D}}$.

The discussion above applies to the context $\mathbb{K}_{\mathbf{D}\times\mathbf{P},\mathbf{R}}$ as well. To summarize: For a *complete* investigation of $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$, the RBAC engineer should run an attribute exploration on the minimal instantiations of the following contexts:

- $\mathbb{K}^{\mathsf{P}}_{\mathbf{R},\mathbf{D}}$ for each permission $\mathsf{P} \in \mathbf{P}$ to infer document implications
- $\mathbb{K}^{\mathsf{P}}_{\mathbf{D},\mathbf{R}}$ for each permission $\mathsf{P} \in \mathbf{P}$ to infer role implications
- $\mathbb{K}_{\mathbf{R}\times\mathbf{D},\mathbf{P}}$ to infer permission implications

For the context $\mathbb{K}_{\mathbf{R}\times\mathbf{D},\mathbf{P}}$, one could add the additional attribute $\perp$ in order to obtain constraints which express the disjointness of some permissions.

5 Evaluation of the Approach for a Real-Life-Example

In this section, we apply our approach to the example introduced in Tab. 1. Due to space limitations, we do not conduct a complete attribute exploration: Instead we consider only the contexts $\mathbb{K}_{\mathbf{D}\times\mathbf{R},\mathbf{P}}$ and $\mathbb{K}^{\mathsf{MO}}_{\mathbf{D},\mathbf{R}}$.

5.1 Attribute exploration for $\mathbb{K}_{D\times R,\mathbf{P}}$.

In this section, we conduct the attribute exploration on the minimal instantiation $\mathbb{K}_{\min}$ of $\mathbb{K}_{D\times R,\mathbf{P}}$. For this exploration, as discussed in Sec. 2, we added an additional attribute $\bot$ to the set of attributes. An excerpt of $\mathbb{K}_{\min}$, together with its concept lattice, is provided in Fig. 1. This is the context the RBAC engineer starts the attribute exploration on. In $\mathbb{K}_{D\times R,\mathbf{P}}$, thus in $\mathbb{K}_{\min}$, we have the following implications:

1. $\mathsf{MW} \to \mathsf{MO}$
2. $\mathsf{MA} \to \mathsf{MO}$
3. $\bot \to \mathsf{MO} \wedge \mathsf{MW} \wedge \mathsf{MA}$
4. $\mathsf{MO} \wedge \mathsf{MW} \wedge \mathsf{MA} \to \bot$.

The first implication is read: Whenever some agent can write some document, then this agent can open this document as well. It can easily be verified that this implication should indeed hold in any interpretation $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$, so we add the property inclusion $\mathsf{mayWrite} \sqsubseteq \mathsf{mayOpen}$ to our DL knowledge base. This is the first example of a statement which can be modeled with a DL statement, but not with matrix $\mathbb{K}_{\mathbf{R},\mathbf{D},\mathbf{P}}$ alone.

The next implication can be handled analogously, and we add the inclusion $\mathsf{mayApprove} \sqsubseteq \mathsf{mayOpen}$ to the knowledge base.

The third implication trivially holds due to the definition of $\bot$.

The last implication can, due to the first two implications, be simplified to $\mathsf{MW} \wedge \mathsf{MA} \to \bot$. Due to the definition of $\bot$, this is read: No agent can both write and approve some document. Again, the engineer decides that this implication is valid. Thus she adds the disjoint property axiom $\mathsf{MW} \sqcap \mathsf{MA} \sqsubseteq \bot$ to the knowledge base.

If it is later verified that the complete RBAC policy is consistent, which can be done with a DL reasoner, then each document which can be written or can be approved has to be readable and furthermore no document can be written and approved by the same agent. These are constraints which have not been contained in the matrix but where derived by our methodology.

5.2 Attribute Exploration for $\mathbb{K}^{\mathsf{mayOpen}}_{D,\mathbf{R}}$.

For a second example, attribute exploration is performed on the minimal instantiation context $\mathbb{K}_{\min}$ of $\mathbb{K}^{\mathsf{mayOpen}}_{\mathbf{D},\mathbf{R}}$. The context $\mathbb{K}_{\min}$ looks like the left third of the cross table in Tab. 1 despite that it is transposed and document types are replaced by documents (columns are roles, rows are documents). Due to space

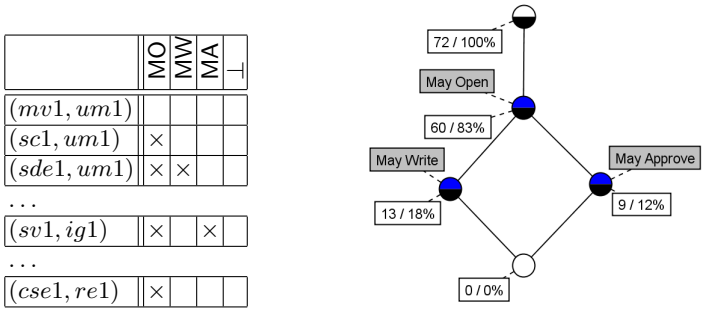

Fig. 1. The instantiation context $\mathbb{K}_{D\times R,\mathsf{P}}$ and its concept lattice

limitation, we do not conduct a complete attribute exploration on $\mathbb{K}_{\min}$, but only provide an example for an valid and an invalid implication.

Let us first note that in $\mathbb{K}^{\mathsf{mayOpen}}_{\mathsf{D},\mathsf{R}}$, the attributes SV, LDE and CSE apply to all objects. So, according to the discussion in the implications $\emptyset \to \mathsf{SV}$, $\emptyset \to \mathsf{LDE}$ and $\emptyset \to \mathsf{CSE}$ hold in all instantiations of $in\mathbb{K}^{\mathsf{mayOpen}}_{\mathsf{D},\mathsf{R}}$, thus we can add the GCIs $\top \sqsubseteq \forall \mathsf{SV}.\mathsf{mayOpen}^-$, $\top \sqsubseteq \forall \mathsf{LDE}.\mathsf{mayOpen}^-$ and $\top \sqsubseteq \forall \mathsf{CSE}.\mathsf{mayOpen}^-$ to our knowledge base.

A example for an implication (of the stem base) of $\mathbb{K}_{\min}$ is $\mathsf{TE} \to \mathsf{ME}$. During the attribute exploration, the RBAC engineer has to decide whether this implication holds in all desired interpretations of $in\mathbb{K}^{\mathsf{mayOpen}}_{\mathsf{D},\mathsf{R}}$. In fact there might be a contract document in preparation by a technical editor which is not allowed to be opened by a marketing employee. Thus the RBAC engineer adds a counterexample to the context $(\mathsf{CCD\_in\_prep}, \mathsf{TE}, \mathsf{MO}) \in I$ and $(\mathsf{CCD\_in\_prep}, \mathsf{ME}, \mathsf{MO}) \notin I$.

Another example for an implication (of the stem base) of is $\mathsf{MV} \to \mathsf{SC}$. In fact, the RBAC engineer realizes that this implication must hold: Any document which can be opened by a marketplace visitor can be opened by a service consumer as well. So she adds the GCI $\forall \mathsf{MV}.\mathsf{mayOpen}^- \sqsubseteq \forall \mathsf{SC}.\mathsf{mayOpen}^-$ to the knowledge base. This is again an example which cannot be derived from $\mathbb{K}_{\mathsf{R},\mathsf{D},\mathsf{P}}$ alone.

6 Conclusion and Future Research

In this paper we used the access control matrix as basic model for the behavior of RBAC and called this an RBAC matrix. We discussed three interpretations of an RBAC matrix and described that for the permissive approach additional constraints can be derived which are not contained in the RBAC matrix. This additional information was added to a so called RBAC policy, modeled in DL.

For obtaining a *complete* RBAC policy, we introduced a strict methodology, based on FCA. The general approach was to derive different dyadic context from RBAC matrix context $\mathbb{K}_{\mathsf{R},\mathsf{D},\mathsf{P}}$ and conduct an attribute exploration on them. The attribute exploration allowed finding unintended implications and to derive constraints and make them explicit.

Our ongoing work comprises several directions. First, we are seeking a smaller DL fragment which meets our modeling requirements. This is particularly for the permissive approach essential, as the DL modelling we used so far is based on some non-standard DL constructors. Next, we want to support positive and negative authorizations in one policy. That is, we want to combine the permissive and prohibitive approach, so we have to investigate how our approach has to be extended in order to do so. Finally, recall that our approach was based on the assumption that sets of roles resp. document types are fixed. In some applications this might be too strict. The three interpretations would have to be adapted and even attribute exploration for the strict approach might make sense if we drop this assumption. This is subject of future research as well. In the long run, we target at a comprehensive methodology for utilizing DLs for RBAC.

References

1. B. Lampson, "Protection," in *Proceedings of the 5th Annual Princeton Conference on Information Sciences and Systems*, pp. 437–443, 1971.
2. R. Sandhu, D. Ferraiolo, and R. Kuhn, "The NIST model for role-based access control: towards a unified standard," in *RBAC '00: Proceedings of the fifth ACM workshop on Role-based access control*, pp. 47–63, ACM, 2000.
3. G. Saunders, M. Hitchens, and V. Varadharajan, "Role-based access control and the access control matrix," *SIGOPS Oper. Syst. Rev.*, vol. 35, no. 4, pp. 6–20, 2001.
4. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2. ed., 2007.
5. M. Knechtel and J. Hladik, "RBAC authorization decision with DL reasoning," in *ICWI '08: Proceedings of the IADIS Int. Conf. WWW/Internet*, 2008.
6. M. Knechtel, J. Hladik, and F. Dau, "Using OWL DL reasoning to decide about authorization in RBAC," in *OWLED '08: Proceedings of the OWLED 2008 Workshop on OWL: Experiences and Directions*, 2008.
7. F. Baader, B. Ganter, U. Sattler, and B. Sertkaya, "Completing description logic knowledge bases using formal concept analysis," in *Proceedings of the Twentieth Int. Joint Conf. on Artificial Intelligence (IJCAI-07)*, AAAI Press, 2007.
8. C. Lutz and U. Sattler, "Mary likes all cats," in *Proceedings of the 2000 Int. Workshop in Description Logics (DL2000)* (F. Baader and U. Sattler, eds.), no. 33 in CEUR-WS, (Aachen, Germany), pp. 213–226, RWTH Aachen, August 2000. Proceedings online available from http://SunSITE.Informatik.RWTH-Aachen.DE/Publications/CEUR-WS/Vol-33/.
9. S. Rudolph, M. Krötzsch, and P. Hitzler, "All elephants are bigger than all mice," in *Proceedings of the 21st International Workshop on Description Logics (DL2008)*, 2008.
10. B. Ganter and S. A. Obiedkov, "Implications in triadic formal contexts.," in *Proceedings of the 12th International Conference on Conceptual Structures (ICCS 2004)* (K. E. Wolff, H. D. Pfeiffer, and H. S. Delugach, eds.), vol. 3127 of *Lecture Notes in Computer Science*, pp. 186–195, Springer, 2004.