Using Description Logics in Relation Based Access Control

Rui Zhang[1], Alessandro Artale[2], Fausto Giunchiglia[1] and Bruno Crispo[1]

[1] Faculty of Information Engineering and Computer Science
I-38100 Trento, Italy
{zhang,fausto,crispo}@disi.unitn.it

[2] Faculty of Computer Science
Free University of Bozen-Bolzano
I-39100 Bolzano, Italy
artale@inf.unibz.it

Abstract. Relation Based Access Control (*RelBAC*) is an access control model designed for the new scenarios of access control on Web 2.0. Under this model, we discuss in this paper how to formalize with Description Logics the typical authorization problems of access control together with the enforcement of an important security property: Separation of Duties (*SoD*) and some high level security policies about the composition of those subjects on which to separate the duties.

1 Introduction

Access control is an important branch of computer security. Nowadays, access control models become more and more complex in order to represent the dynamics of the subject, i.e., the new scenario where objects and permissions of access control are located on the web. Early access control models, such as *Mandatory Access Control* and *Discretionary Access Control*, have evolved to more complex models such as Role-Based Access Control (RBAC) [4] and, most recently, Relation-Based Access Control (*RelBAC*) [8]. Complex models, such as RBAC, have been deeply studied and many logical formalisms have been proposed to formalize their access control policies. In particular, it has been shown how *RelBAC* can be used to model access control in terms of lightweight ontologies [6, 7, 18] of users, objects and permissions, and how this can be exploited to automatically manage permissions.

In this paper, we discuss how to formalize the typical authorization problems in the *RelBAC* model with a specific Description Logic (DL), i.e., the DL $\mathcal{ALCQIBO}$ [13, 14, 16]. *RelBAC* is a new model designed specially for the various dynamic scenarios of Web 2.0 in [8]. It is natural and flexible such that it is easy to understand and use for personal resources of an ordinary web user (see, e.g., photos on Flickr[1]). The main feature of *RelBAC* is that a permission, intuitively the operation allowed to be performed on some resource, is modeled as a binary relation between a set of subjects and a set of objects. Furthermore, it is flexible enough to enforce cardinality related access such as to restrict that 'anonymous users can view no more than 5 of my photos'. *RelBAC* is formalized

[1] http://www.flickr.com

with an access control domain specific Description Logic in order to provide formal syntax and semantics plus an automated reasoning mechanism to facilitate the access control management.

The contributions of this paper are as follows:

- Focusing on authorization problem of *RelBAC*, we discuss the usage of DL to formalize access control policies.
- Different formalizations of Separation of Duties (*SoD*) are analyzed as an important security property.
- Dynamic *SoD* and *high-level* security policies about *SoD* are also formally represented and discussed.

The rest of the paper is organized as follows. Section 2 describes the DL exploited to capture the access control problems. Section 3 discusses the expressiveness of the proposed DL to faithfully capture the *RelBAC* application domain. In Section 4 we discuss the logical formalization of a security property, i.e., Separation of Duties. The related work is summarized in Section 5 and we make our concluding remarks in Section 6.

2 The Description Logic $\mathcal{ALCQIBO}$

The logic $\mathcal{ALCQIBO}$ extends the description logic $\mathcal{ALC}$ [1] with qualified cardinalities, inverse roles, nominals and Boolean for roles (see [16, 14, 13] for extensions of DLs with Booleans between roles). We define the syntax of $\mathcal{ALCQIBO}$ as follows.

Definition 1 ($\mathcal{ALCQIBO}$ Syntax). *Let* $\mathsf{N_C}$, $\mathsf{N_R}$ *and* $\mathsf{N_I}$ *be pairwise disjoint and countably infinite sets of* concept names, role names *and* individual names. *Then* concept expressions *and* role expressions *are defined as follows:*

$$C, D \ ::= \ A \ \mid \ \neg C \ \mid \ C \sqcap D \ \mid \ \geq n\, R.C \ \mid \ \{a_i\}$$
$$R, S \ ::= \ P \ \mid \ R^- \ \mid \ \neg R \ \mid \ R \sqcap S$$

where $A \in \mathsf{N_C}$, $P \in \mathsf{N_R}$, $a_i \in \mathsf{N_I}$ *and* $n \in \mathbb{N}$.
A Knowledge Base (KB) is a pair $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ *where* $\mathcal{T}$, *called* TBox, *is a finite set of* general concept inclusions (GCIs) *of the form* $C \sqsubseteq D$ *and a finite set of* general role inclusions (GRIs) *of the form* $R \sqsubseteq S$, *while* $\mathcal{A}$, *called* ABox, *is a finite set of concept and role assertions of the form* $C(a_i)$ *and* $R(a_i, a_j)$, *with* $a_i, a_j \in \mathsf{N_I}$.

An $\mathcal{ALCQIBO}$-*interpretation*, $\mathcal{I}$, is a pair $(\Delta, \cdot^{\mathcal{I}})$ where Δ is a non-empty set called the *domain* of $\mathcal{I}$ and $\cdot^{\mathcal{I}}$ is a function mapping each $A \in \mathsf{N_C}$ to a subset $A^{\mathcal{I}} \subseteq \Delta$ and each $P \in \mathsf{N_R}$ to a relation $P^{\mathcal{I}} \subseteq \Delta \times \Delta$. Furthermore, $\cdot^{\mathcal{I}}$ applies also to individuals by mapping each individual name $a_i \in \mathsf{N_I}$ into an element $a_i^{\mathcal{I}} \in \Delta$ such that $a_i^{\mathcal{I}} \neq a_j^{\mathcal{I}}$, for all $i \neq j$, i.e., we adopt the so called *unique name assumption* (UNA). We extend the mapping $\cdot^{\mathcal{I}}$ to complex roles and concepts as follows:

$$
\begin{aligned}
(R^{-})^{\mathcal{I}} &:= \{(y,x) \in \Delta \times \Delta \mid (x,y) \in R^{\mathcal{I}}\},\\
(\neg R)^{\mathcal{I}} &:= \Delta \times \Delta \setminus R^{\mathcal{I}}, \quad (\neg C)^{\mathcal{I}} := \Delta \setminus C^{\mathcal{I}},\\
(R \sqcap S)^{\mathcal{I}} &:= R^{\mathcal{I}} \cap S^{\mathcal{I}}, \quad (C \sqcap D)^{\mathcal{I}} := C^{\mathcal{I}} \cap D^{\mathcal{I}},\\
(\geqslant n\ R.C)^{\mathcal{I}} &:= \{x \in \Delta \mid \sharp\{y \in \Delta \mid (x,y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \geq n\}, \quad \{a_i\}^{\mathcal{I}} := \{a_i^{\mathcal{I}}\}.
\end{aligned}
$$

An $\mathcal{ALCQIBO}$-interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ is said a *model* of a KB, $\mathcal{K}$, iff it satisfies $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, for all $C \sqsubseteq D \in \mathcal{K}$, $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$, for all $R \sqsubseteq S \in \mathcal{K}$, $a_i^{\mathcal{I}} \in C^{\mathcal{I}}$, for all $C(a_i) \in \mathcal{A}$, and $(a_i^{\mathcal{I}}, a_j^{\mathcal{I}}) \in R^{\mathcal{I}}$, for all $R(a_i, a_j) \in \mathcal{A}$. In this case we say that $\mathcal{K}$ is satisfiable and write $\mathcal{I} \models \mathcal{K}$. A concept C (role R) is *satisfiable w.r.t.* $\mathcal{K}$ if there exists a model $\mathcal{I}$ of $\mathcal{K}$ such that $C^{\mathcal{I}} \neq \emptyset$ ($R^{\mathcal{I}} \neq \emptyset$).

As usual, we can define a number of useful abbreviations:

$$
\begin{aligned}
C \sqcup D &\text{ for } \neg(\neg C \sqcap \neg D)\\
(\leqslant n\ R.C) &\text{ for } \neg(\geqslant n+1\ R.C)\\
(= n\ R.C) &\text{ for } \neg(\geqslant n+1\ R.C) \sqcap (\geqslant n\ R.C)\\
\exists R.C &\text{ for } (\geqslant 1\ R.C)\\
\forall R.C &\text{ for } (\leqslant 0\ R.\neg C)\\
\top &\text{ for } A \sqcup \neg A \text{ (for some concept } A)\\
\bot &\text{ for } \neg\top\\
\mathcal{U} &\text{ for } R \sqcup \neg R \text{ (for some role } R)
\end{aligned}
$$

Note that, TBox axioms can be internalized. Indeed, we can encode each axiom $C \sqsubseteq D$ as the concept expression $\forall \mathcal{U}.(\neg C \sqcup D)$, while each role axiom $R \sqsubseteq S$ can be encoded as the concept expression $\forall \mathcal{U}.\forall(R \sqcap \neg S).\bot$. (To encode ABox assertions as concept expressions see [16]).

Concerning the complexity of $\mathcal{ALCQIBO}$, KB satisfiability can be reduced to reason over the two-variable first-order fragment with counting quantifiers which is NExpTime-complete [15]. On the other hand, Boolean modal logic is a proper sub-language of $\mathcal{ALCQIBO}$ and it is NExpTime-complete [13]. Summing up, reasoning in $\mathcal{ALCQIBO}$ is NExpTime-complete.

3 The Authorization Problem in *RelBAC*

Here we discuss the authorization problem, which deals with questions like 'who is authorized to access what'. We distinguish two different phases: *general authorizations* on group of users and sets of resources, and *ground authorizations* onto individual users and/or resource instances.

3.1 General Authorizations

In *RelBAC*, a generic permission P (e.g., Write) is modeled as a binary relation between a class of users U (e.g., SW-Developer) and a class of objects O (e.g., Java-Code). The following general constraints can be captured in $\mathcal{ALCQIBO}$.

1. '*The permission P applies only between users U and objects O*'.
   This is a form of domain and range constraint for the binary relation P and can be modeled in $\mathcal{ALCQIBO}$ with the following axioms:

   $\exists P.\top \sqsubseteq U$ `Domain Restriction`
   $\exists P^{-}.\top \sqsubseteq O$ `Range Restriction`
2. '*Users in U can access just objects in O with P*'
   '*Objects in O can be accessed just from users in U with P*'.
   We can represent these constraint using universal restrictions as:

   $U \sqsubseteq \forall P.O$ `Universal Restriction`
   $O \sqsubseteq \forall P^{-}.U$ `Universal Restriction`
3. '*Users in U are allowed to access (with P) at most n objects in O*'
   '*At most n users in U are allowed to access any given object in O with P*'.
   We can represent these constraints using cardinality constraints as:

   $U \sqsubseteq (\leqslant n\ P.O)$ `Cardinality Restriction`
   $O \sqsubseteq (\leqslant n\ P^{-}.U)$ `Cardinality Restriction`
4. '*Users in U have access to at least m objects in O with P*'
   '*At least m users in U have access to any object in O with P*'.
   We can represent these constraints using cardinality constraints as:

   $U \sqsubseteq (\geqslant m\ P.O)$ `Cardinality Restriction`
   $O \sqsubseteq (\geqslant m\ P^{-}.U)$ `Cardinality Restriction`
5. '*All users in U have access to all objects in O with P*'
   '*All objects in O are accessed by all users in U with P*'.
   This rule defines a so called *Total Access Control* rule (TAC) and can be captured using the negation of roles constructor in cardinality restriction:

   $U \sqsubseteq \forall \neg P.\neg O$ `TAC Rule`
   $O \sqsubseteq \forall \neg P^{-}.\neg U$ `TAC Rule`

3.2 Ground Authorizations

Using the ABox mechanism of $\mathcal{ALCQIBO}$ we can assert particular facts associated to given individuals of the domain. The following is a list of the most common assignments concerning single individuals that can be captured in $\mathcal{ALCQIBO}$ (in the following u and o are individuals in $\mathsf{N_I}$, P is a role in $\mathsf{N_R}$, U and O are concepts in $\mathsf{N_C}$).

1. '*The user u is allowed to access the object o with P*'.
   This is represented as: $P(u, o)$.
   For example, $Update(david, mb903ll/a)$ says that 'David is allowed to update the entry MB903LL/A'.
2. '*The user u is allowed to access maximum n objects in O with P*'.
   This is represented as: $(\leqslant n\ P.O)(u)$.
   For example, $(\leqslant 5\ Update.Digital)(David)$ says that 'David is allowed to update maximum 5 entries of Digital'.

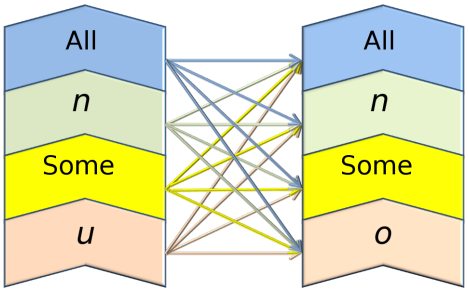

Fig. 1. Combination of `RULE` cardinalities in *RelBAC*

3. '*The user u is allowed to access minimum n objects in O with P*'.
   This is represented as: $(\geqslant n\ P.O)(u)$.
   For example, $(\geqslant 5\ Update.Digital)(David)$ says that 'David is allowed to update minimum 5 entries of Digital'.
4. '*The user u is allowed to access all objects in O with P*'.
   This is represented as: $(\forall \neg P.\neg O)(u)$.
   For example, $(\forall \neg Update.\neg Digital)(David)$ says that 'David is allowed to update all entries in Digital'.
5. '*Minimum n users in U are allowed to access the object o with P*'.
   This is represented as: $(\geqslant nP^{-}.U)(o)$.
   For example, $(\geqslant 3\ Update^{-}.Apple)(mb903ll/a)$ says that 'At least 3 friends from Apple are allowed to update the entry MB903LL/A'.
6. '*All users in U are allowed to access the object o with P*'.
   This is represented as: $U \sqsubseteq \exists P.\{o\}$.
   For example, $Apple \sqsubseteq \exists Update.\{mb903ll/a\}$ says that 'All friends from Apple are allowed to update the entry MB903LL/A'.

Besides the traditional access control rules which can specify only 'one-to-one' and 'one-to-many' mappings about individuals, we see that the ABox of $\mathcal{ALCQIBO}$ provides many ways to write a ground access control rule in *RelBAC*. Altogether, as is shown in Figure 1 (taken from [18]), *RelBAC* can specify 15 kinds of access control rules with a single DL axiom (counting minimum/maximum/equal cardinality restriction as one). These rules show, both the power of *RelBAC* as an access control model and the expressiveness of the logic to be able to capture this scenario. The cardinality related rules (formed with cardinality restriction) are important especially in those scenarios where number is a key factor for access control.

So far we discussed the permission assignments at design time. When an access control request arrives at run time the access control should decide whether to accept or deny these requests according to the assignments done at design time. Since a *RelBAC* system is an $\mathcal{ALCQIBO}$ KB, while a request is captured

by an ABox, then at run time the system decides to grant an access if by adding the ABox to the current KB the resulting KB is satisfiable.

4 Separation of Duties

Separation of Duties *SoD* is an important security property in modern access control systems. It enforces that more than one person is required to complete a task. In this section, we will discuss the general meaning of an *SoD*, its enforcement at design and run time and the *high-level* security policy [12] for an *SoD*.

4.1 General *SoD*

Definition 2 (Separation of Duties *SoD*). *An* SoD *states that, if a sensitive task consists of two steps, then two different users should perform mutually exclusive steps. More generally, when a sensitive task is composed of n steps, an* SoD *constraint requires the cooperation of at least k (for some $k \leqslant n$) different users to complete the task.*

In one of the most well-known access control model, RBAC[4], *SoD* is enforced with the help of restrictions on the 'roles[2]'. The *SoD* that 'different users should perform two different steps of a task' can be enforced at design time by restricting any user from the assignments to the two roles, each of which is assigned the permission to carry out a step of the two. For a more general *SoD* that 'different users should perform n different steps of a task', each step is modeled as an RBAC role R_i, and the *SoD* is formally expressed as the following formula: $R_1 \sqcap \ ... \sqcap R_i \sqcap \ ... \ \sqcap R_n \sqsubseteq \bot$.

In *RelBAC*, a permission is a relation which links a subject with an object. To enforce this *SoD* in *RelBAC*, we can simply assert an axiom about the permissions. For example, suppose in a scenario of sales force automation[3], *to initiate, process, check and archive an order should not be completed by only one user*. Suppose *Initiate, Process, Check* and *Archive* are four permissions with the same domain as users and co-domain as orders. The *SoD* above can be expressed in *RelBAC* as

$$Initiate \sqcap Process \sqcap Check \sqcap Archive \sqsubseteq \bot$$

This policy restricts any pair (u, o) from belonging to all four sets *Initiate, Process, Check* and *Archive*.

In general, given n steps of a task $step_1, ..., \ step_n$, the *SoD* requires at least k ($k \leqslant n$) users can fulfill all these n steps. Suppose any of the k users can fulfill

[2] To be differentiated from a DL role, the 'role' in the RBAC model is a component simulating the real world enterprise organism. A user can only execute a permission assigned to the role that s/he can activate.

[3] http://www.salesforce.com

maximum m steps. In the worst case, everyone can fulfill equal number of steps, and satisfies the following DL axiom:

$$(k-1) * m < n \quad i.e. \quad m \leqslant \lceil n/(k-1) \rceil - 1 \tag{1}$$

because $k,\ m,\ n$ are all integers. This means intuitively that any user can be assigned to at most m of these duties as restricted in Formula 1. Thus, any $m+1$ of these duties should not be assigned to one user. Then *RelBAC* can enforce the *SoD* with the following axiom:

$$\bigsqcup_{i=1}^{C_n^{\lceil n/(k-1) \rceil}} \left(\prod_{j=1}^{\lceil n/(k-1) \rceil} P_{ij} \right) \sqsubseteq \bot \tag{2}$$

in which P_{ij} stands for one of the m permissions for each step, and C_n^k is the binomial coefficient of 'n choose k'.

Following the example above, given the 4 duties in the SFA scenario, an *SoD* requires that *at least 3 users should be involved*. This *SoD* can be enforced as follows.

$$(Initiate \sqcap Process) \sqcup (Check \sqcap Initiate) \sqcup (Process \sqcap Archive) \sqcup$$
$$(Process \sqcap Check) \sqcup (Archive \sqcap Initiate) \sqcup (Check \sqcap Archive) \sqsubseteq \bot$$

as $C_n^{\lceil n/(k-1) \rceil} = C_4^{\lceil 4/2 \rceil} = C_4^2 = 6$.

An *SoD* can be enforced both at design and at run time. Up to now, we have discussed the *SoD* designed by the administrator off-line. An *SoD* enforced at run time intuitively means that the duties to be separated can be assigned to one user during design, but cannot be executed by her simultaneously.

RBAC fulfills it with the concept of *session* as representatives of the user at run time by restricting sessions from activation of separated roles. To enforce an *SoD* at run time, the concept of 'session' has been introduced in the RBAC model. At run time, a user activates sessions to execute permissions. The duties assigned to the user at design time and restricted to be executed simultaneously are separated by enforcing that no session can execute them at the same time.

In *RelBAC*, we do not need the concept of 'session', but use a run time permission to describe the state of a permission in execution to enforce *SoD* at run time.

Definition 3 (Run Time Permission - *RTP*). *A* RTP *describes the current execution of a permission. We assume that the name of a* RTP *is formed by alternating the verb (phrase) denoting the permission into the present participle of the verb (phrase). A user cannot be assigned to a* RTP *unless she has the permission for the original permission.*

For each *RelBAC* permission, a *RTP* is introduced. For example, assume that *Initiating* is the *RTP* for the permission *Initiate*. As said above, a user cannot execute the permission *Initiating* without having the permission *Initiate*. Now,

the *SoD* '*a user cannot initiate and process an order at the same time*' is enforced as follows (we assume that both permissions have the same range, $Order$):

$$Initiating \sqcap Processing \sqsubseteq \bot$$

In the real world, a user can be granted the permission to initiate an order (as a customer) and to process an order (as a sales agent), but cannot perform the two permissions (duties) simultaneously in order to avoid the process of one's own order.

Although an *SoD* can be enforced at design and run time with similar role axioms in $RelBAC$, the mechanisms regulating it are different. To enforce an *SoD* at run time, the monitoring mechanism should inform the access control system in real time, so that the current state such as *Alice is initiating an order 'Bolzano'* should be recognized. Then the knowledge base should be updated with the new assertion $Initiating(alice, bolzano)$. Therefore, the above *SoD* (ruling out the possibility of initiating and processing an order at the same time) in the KB $\mathcal{K}$ entails the following:

$$\mathcal{K} \sqcup \{Initiating(alice, bolzano)\} \models \neg Processing(alice, bolzano)$$

although Alice might have permissions to both initiate and process some order.

4.2 High Level Security Policy about *SoD*

For the general *SoD* property, the composition of the k users to complete the task is sometimes important. The administrator may like to constraint first that these users are from certain groups, cardinality related constraints are also necessary for the composition.

N.Li et al. studied *SoD* in detailed requirements for specific attributes of the users in addition to numbers of each kind of users. An algebra was proposed in [12] to specify complex policies combining requirements on user attributes and number. On top of the cardinality constraints for given duties, the algebra can specify the composition of the users for the *SoD* which they regard as *high-level* security policy. For example, beyond to restrict that at least 2 users should be involved for the 4 steps in the *SFA* schema of Section 4.1, it further enforces that the set of users that can complete the order fulfillment task involve customer(s) to initiate the order and sales manager(s) to check the order. For example, the composition of the user set should be as follows.

1. At least one sales manager to check orders and at least one customer to initiate orders.
2. At least one sales manager and at least one customer and maybe some other sales manager or customer involved, but no others than those two kinds of users.
3. Exactly two users, one sales manager and one customer.

Case 1 means that a customer should be involved to initiate the order and a manager should be involved to check the order, for the other users, the administrator does not care who processes and archives the order. Case 2 specifies that only customer and manager can be involved which means the same manager (or some other manager) will take charge of the duties to process and to archive. Case 3 is the most strict by allowing only one customer and one manager to be involved.

RelBAC can achieve this kind of constraints with *object-centric* rules with the *cardinality restriction* constructor. For example, as for the cases above, the three constraints for the set of users can be formalized as follows.

$$
\begin{aligned}
Order &\sqsubseteq (\geqslant 1\ Initiate^{-}.Customer) \sqcap (\geqslant 1\ Check^{-}.Manager) \\
Order &\sqsubseteq \forall Involve.(Customer \sqcup Manager) \sqcap \\
&\quad (\geqslant 1\ Initiate^{-}.Customer) \sqcap (\geqslant 1\ Check^{-}.Manager) \\
Order &\sqsubseteq (= 2\ Involve.(Customer \sqcup Manager)) \sqcap \\
&\quad (= 1\ Initiate^{-}.Customer) \sqcap (= 1\ Check^{-}.Manager) \\
&\quad \sqcap \neg\exists\ Involve.(Customer \sqcap Manager)
\end{aligned}
$$

Where $Involve$ is a permission more general than any of the 4 permissions for the 4 duties in the above schema, i.e.

$$Initiate^{-} \sqcup Process^{-} \sqcup Check^{-} \sqcup Archive^{-} \sqsubseteq Involve$$

This kind of *high-level* security policy complements the general *SoD* policy as discussed in Section 4.1 because it has the full power to describe the composition of the set of subjects including the exact cardinality.

5 Related Work

Description Logics [1] arouse the interests in the AI community for their expressiveness and decidability of the reasoning services. Various papers describe the use of Description Logics to formalize access control models and use state of the art Description Logic reasoners to formalize security properties and check their consistency (see, e.g. [2, 5, 8, 9, 18, 19]).

F.Giunchiglia et al. introduced in [8] the *RelBAC* model together with a domain specific Description Logic as the formalism. *RelBAC* captures, with subsumption axioms, the dynamic hierarchies of the subjects, objects and permissions and provides, with cardinality restrictions, powerful cardinality related specifications of access control rules. The theory of Lightweight Ontology [7] can be used to model the social community into ontologies as discussed in [9] and can facilitate the management of knowledge hierarchies and access control rules.

In [19] an early attempt was made by C.Zhao et al. to apply DLs to the representation of policies in the RBAC model. In their proposal, *users, roles, sessions* and *permissions* are formalized as DL concepts but *objects* are regarded as encapsulated inside *permissions* together with *operations*. This results in an

explosion in the number of permissions and the corresponding difficulty to specify policies about *objects*. Moreover, they proposed to use only the *existential restriction* constructor for *permission assignments*.

Another formalization of RBAC in DLs was proposed by J.Chae et al. [2], where an *operation* is represented by a DL role. Their system has several critical points, and in particular:

- Miss use of *existential quantifier*. In the semantics of their formalization, the assignment with the formula '$Admin \sqsubseteq \exists CanRead.Log$' assigns to *all* administrators the read access to *all* log files. But the DL semantics of this formula enforces only the existence of *some* connections between administrators and log files. In our case, the `TAC` rules are introduced to cover precisely this case (see Sect. 3.1).
- The formalization of 'assign' and 'classify' into DL roles seems redundant. These DL roles are supposed to connect users to RBAC roles or object to object classes. We explicitly use an ABox mechanism to better deal with individuals.

However, their work is relevant for our approach since:

- They inspired us in the way they formalized *operation*, i.e., by introducing a binary relation from a *subject* to an *object*.
- They extended RBAC with the object hierarchy similar to the user hierarchy which facilitates the permission propagation.

Recently, T.Finin et al. proposed to use OWL[4] language as the formalization of the RBAC model in [5]. They provide two ways to formalize a RBAC role, as a class or as an attribute. N3Logic is used together with DL subsumption reasoning. Authorization decision queries can be answered using DL reasoners in their system.

Another important work is related with the formalization of *SoD*. N.Li et al. studied *SoD* in [12], and proposed an algebra to specify the composition of the users that share the duties. In [11] N.Li et al. modeled the problem of an *SoD* of n duties among k users with a first order logic formula as

$$\forall u_1 ... u_{k-1} \in U((\bigcup_{i=1}^{k-1} auth\_perms_\gamma[u_i]) \not\supseteq \{p_1 ... p_n\}) \tag{3}$$

with universal quantifier on arbitrary $k-1$ users in space of U. Formula 3 specifies that the collection of all the permissions explicitly/implicitly assigned to this $k-1$ users should not be a superset of all the n steps of duties. Their solution has the complexity of $(|U|^{k-1} * n)$ which explodes to the cardinality of the subject space. In *RelBAC*, as shown in Formulas 2, our solution enforces a sufficient but not necessary condition of the *SoD* because the 'ceiling' operator ($\lceil \cdot \rceil$) is an approximation of the exact value for $n/(k-1)$. For example, in the schema above, the representation are the same for $k = 3$ and $k = 4$. However the

[4] http://www.w3.org/TR/owl-guide/

computational complexity is only $(n^{n/k})$. Considering that the number of steps which is n, is far less than the number of users in the system, which is $|U|$, our method is more efficient than [11].

An existing industry standard is XACML [3] which is an XML based access control policy language without formal semantics such as in a logic. Kolovski et al. used DL to provide formal semantics for XACML in [10]. *RelBAC*, in contrast, is not only a new access control model, but also a logic with well defined syntax and semantics to express web-based access control policies.

6 Conclusion

In this paper, we discussed a domain specific Description Logic, i.e. $\mathcal{ALCQIBO}$ for access control, in which the reasoning is NExpTime-complete. Exploiting $\mathcal{ALCQIBO}$ to formalize the *RelBAC* model, we have been able to formalize the typical authorization problem of access control. Besides, we studied the formalization of an important security property which is the Separation of Duties (*SoD*). Furthermore, different aspects of *SoD* can be formalized in $\mathcal{ALCQIBO}$ such as dynamic *SoD* and high level security policy of *SoD*.

A basic version of *RelBAC* has been implemented as described in [18]. But the evaluation results with a general purpose DL reasoner, Pellet [17], are not good enough to achieve real-time response. The future work of this direction is to adapt such general purpose DL reasoners to access control knowledge bases for more efficient reasoning.

References

1. Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The description logic handbook: theory, implementation, and applications*. Cambridge University Press, New York, NY, USA, 2003.
2. Jung-Hwa Chae and Nematollaah Shiri. Formalization of rbac policy with object class hierarchy. In Ed Dawson and Duncan S. Wong, editors, *ISPEC*, volume 4464 of *Lecture Notes in Computer Science*, pages 162–176. Springer, 2007.
3. OASIS eXtensible Access Control Markup Language (XACML) TC. http://www.oasis-open.org/committees/tc_home.php? wg_abbrev=xacml.
4. David F. Ferraiolo, Ravi S. Sandhu, Serban I. Gavrila, D. Richard Kuhn, and Ramaswamy Chandramouli. Proposed NIST standard for role-based access control. *Information and System Security*, 4(3):224–274, 2001.
5. T. Finin, A. Joshi, L. Kagal, J. Niu, R. Sandhu, W. Winsborough, and B. Thuraisingham. Rowlbac: representing role based access control in owl. In *SACMAT '08: Proceedings of the 13th ACM symposium on Access control models and technologies*, pages 73–82, New York, NY, USA, 2008. ACM.
6. Fausto Giunchiglia, Maurizio Marchese, and Ilya Zaihrayeu. Encoding classifications into lightweight ontologies. In *ESWC*, pages 80–94, 2006.
7. Fausto Giunchiglia and Ilya Zaihrayeu. Lightweight ontologies. In *Encyclopedia of Database Systems*. Springer, 2008.

8. Fausto Giunchiglia, Rui Zhang, and Bruno Crispo. Relbac: Relation based access control. In *SKG '08: Proceedings of the 2008 Fourth International Conference on Semantics, Knowledge and Grid*, pages 3–11, Washington, DC, USA, 2008. IEEE Computer Society.
9. Fausto Giunchiglia, Rui Zhang, and Bruno Crispo. Ontology driven community access control. In *SPOT2009 - Trust and Privacy on the Social and Semantic Web*, 2009.
10. Vladimir Kolovski, James Hendler, and Bijan Parsia. Analyzing web access control policies. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 677–686, New York, NY, USA, 2007. ACM.
11. Ninghui Li, Mahesh V. Tripunitara, and Ziad Bizri. On mutually exclusive roles and separation-of-duty. *ACM Transactions on Information System Security*, 10(2):5, 2007.
12. Ninghui Li and Qihua Wang. Beyond separation of duty: an algebra for specifying high-level security policies. In *CCS '06: Proceedings of the 13th ACM conference on Computer and communications security*, pages 356–369, New York, NY, USA, 2006. ACM.
13. C. Lutz and U. Sattler. The complexity of reasoning with boolean modal logics. In Frank Wolter, Heinrich Wansing, Maarten de Rijke, and Michael Zakharyaschev, editors, *Advances in Modal Logics Volume 3*. CSLI Publications, Stanford, 2001.
14. C. Lutz and D. Walther. Pdl with negation of atomic programs. *Journal of Applied Non-Classical Logic*, 15(2):189–214, 2005.
15. Ian Pratt-Hartmann. Complexity of the two-variable fragment with counting quantifiers. *J. of Logic, Lang. and Inf.*, 14(3):369–395, 2005.
16. R. A. Schmidt and D. Tishkovsky. Using tableau to decide expressive description logics with role negation. In K. Aberer, K.-S. Choi, N. Fridman Noy, D. Allemang, K.-I. Lee, L. J. B. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber, and P. Cudré-Mauroux, editors, *The Semantic Web, 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC 2007 + ASWC 2007, Busan, Korea, November 11–15, 2007*, volume 4825 of *Lecture Notes in Computer Science*, pages 438–451. Springer, 2007.
17. E. Sirin, B. Parsia, B. Cuenca Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical owl-dl reasoner. *Submitted for publication to Journal of Web Semantics.*, 2003.
18. Rui Zhang. *RelBAC: Relation Based Access Control*. PhD thesis, University of Trento, March 2009.
19. Chen Zhao, NuerMaimaiti Heilili, Shengping Liu, and Zuoquan Lin. Representation and reasoning on rbac: A description logic approach. In Dang Van Hung and Martin Wirsing, editors, *ICTAC*, volume 3722 of *Lecture Notes in Computer Science*, pages 381–393. Springer, 2005.