Department of
Information Engineering
and Computer Science
DISI

UNIVERSITY
OF TRENTO - Italy

DISI - Via Sommarive 14 - 38123 Povo - Trento (Italy)
http://www.disi.unitn.it

A FORMAL PERSPECTIVE ON RELATION BASED ACCESS CONTROL

Alessandro Artale, Bruno Crispo,
Fausto Giunchiglia and Rui Zhang

December 2009

Technical Report # DISI-09-069

A Formal Perspective on Relation Based Access Control *

Alessandro Artale[1], Bruno Crispo[2], Fausto Giunchiglia[2] and Rui Zhang[3]

[1]KRDB Research Centre, Free University of Bozen-Bolzano, Italy
[2]DISI, University of Trento, Italy
[3]CCST, Jilin University, China
{artale}@inf.unibz.it {crispo,fausto}@disi.unitn.it {rui}@jlu.edu.cn

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 typical access control policies with Description Logics. Important security properties, i.e., Separation of Duties (*SoD*) and Chinese Wall constraints are studied and formally represented in *RelBAC* with the expressive DL $\mathcal{ALCQIBO}$. To meet the needs of automated tools for administrators, *RelBAC* can formalize and answer queries about access control requests and administrative checks resorting to the reasoning services of the underlying Description Logic.

Keywords: Access Control models, Description Logics

1 Introduction

A key aspect of any information system is access control. Access control allows organizations to discipline which resources can be used, by whom and how they must be used. Traditionally, the task of specifying correctly all rules and conditions that regulate accesses to resources, referred with the term *access control policies*, has been difficult. Furthermore, the increasing complexity of modern information systems combined with some new paradigms like SOA stressing on aspects such as re-usability and sharing, exacerbates the task of writing access control policies. Since context, domain and entities of the system keep changing and evolving so do the policies. In particular, the problem of supporting security administrators in their task is of paramount importance. While organizational textbooks describe very clean organizational models with well defined responsibility, the real world is quite different. Many organizations find difficult if not impossible to identify a single point of coordination for policy changes and control of their consistency. Furthermore, for not trivial information systems with thousands of policies their management requires automated tools to lower the incidence of human errors.

* A short version of this paper has been published at the 22nd annual Description Logic workshop as 'Using Description Logics in Relation Based Access Control'.

To address such new challenges, security researchers have recently proposed new approaches [1, 12] with the aim of offering richer expressivity in term of supported policies and at the same time easing the management of such policies with the help of automated tools. One such approach is *RelBAC* [8]. By modeling permissions as binary relations between (set of) subjects and (set of) objects *RelBAC* allows to express a richer set of policies compared to existing models (e.g. cardinality is a basic feature of the model [8]). An aspect that however has not yet been fully investigated is the ability of *RelBAC* to offer a set of powerful services to help administrators in doing their job. This paper fills this gap by describing *RelBAC*'s reasoning ability that derive by its formalization in DL.

In general the use of well established reasoners has two important advantages. First, it allows to rigorously verify dynamic conditions at runt-time (e.g. dynamic separation of duties or Chinese Wall constraints). Second, it provides a very important support for security administrators in managing the policies and making sure that while changing policies, inconsistencies are not introduced or general properties are not violated. The paper defines the class of available reasoning services and for which types of policy management problems they can be used. It also provides a complexity classification for such services to evaluate how feasible is their practical use.

The contributions of this paper are thus as follows: *i*) it completes the characterization of policies and properties that can be expressed using *RelBAC*; *ii*) it specifies the class of queries supported by a *RelBAC* system and its reasoning ability in term of access control policy management; *iii*) it provides an analysis on the complexity of such reasoning services.

The rest of the paper is organised as follows. Section 2 specifies the *RelBAC* model; the Description Logic $\mathcal{ALCQIBO}$ and how it is domain specified in *RelBAC*. Section 3 presents in details the formalization of policies, properties and queries using the DL $\mathcal{ALCQIBO}$; Section 4 summarizes the corresponding complexity results depending on the expressiveness of the adopted representation language; then we describe related works in Section 5 and conclude with final remarks in Section 6.

2 The *RelBAC* Approach to Access Control

2.1 The *RelBAC* Model

Relation-based Access Control (*RelBAC*) is an access control model originated in [8]. As shown in the ER Diagram of Figure 1, what distinguishes *RelBAC* from other access control models is the way it models `PERMISSION` in addition to the basic components such as `SUBJECT` and `OBJECT`. A `PERMISSION` is modeled as an operation that users (`SUBJECT`s) can perform on certain resources (`OBJECT`s). To capture this intuition a `PERMISSION` is named with the name of the operation it refers to, e.g., *Write*, and *Read* operation—in *RelBAC* we adopt the convention that `PERMISSION` names are capitalized verbal forms. The *generalization* (loops) on each components represent `IS-A` relations. Not only `SUBJECT` and `OBJECT` are

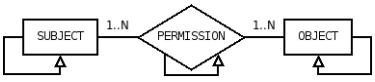

Fig. 1. The ER Diagram of the *RelBAC* Model.

organized along `IS-A` hierarchies but also `PERMISSION`. Thus, by imposing the constraint that '*Update is more powerful than Read*' allows those people who can access some data with the permission *Update* to have already been assigned the permission *Read*.

2.2 The Description Logic $\mathcal{ALCQIBO}$

The logic $\mathcal{ALCQIBO}$ extends the description logic $\mathcal{ALC}$ [3] 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:*

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

where $A \in \mathsf{N_C}$, $P \in \mathsf{N_R}$, $a_i \in \mathsf{N_I}$ *and* $n \in \mathrm{N}$.
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 $C^{\mathcal{I}} \neq \emptyset$ ($R^{\mathcal{I}} \neq \emptyset$).

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.

2.3 The *RelBAC* Logic

In *RelBAC* we distinguish 5 different kinds of specifications that, altogether, contitute an access control system: the hierarchy information, the general assignment, the ground level assignment, the properties of a hierarchy and the queries. *RelBAC* uses the description logic $\mathcal{ALCQIBO}$ to express each specification by associating a concept name to each `SUBJECT` and `OBJECT` while permissions are described by means of role names.

1. **Hierarchies**
   In the access control domain, `SUBJECT`, `OBJECT` and `PERMISSION` are organized in a taxonomy along the `IS-A` relation [7]. An `IS-A` relation is represented as a concept or role *inclusion* axiom in *RelBAC*:

   $$C \sqsubseteq D \quad \text{or} \quad P \sqsubseteq Q$$

   where C, D are either `SUBJECT`s or `OBJECT`s and P, Q are both `PERMISSION`s.
2. **General Assignments**
   They specify the permissions existing between a `SUBJECT` and an `OBJECT` expressing different forms of constraints that we will describe in Section 3.1. Policies can also be specified from the `OBJECT` perspective by using the inverse role construct.
3. **Ground Assignments**
   Access policies can also be expressed at the instance level as we will describe in Section 3.2. At this level, constraints on access control are expressed at the level of single domain instances.
4. **Properties**
   They specifies general constraints and conditions over a given hierarchy. Some of the most recurrent and useful properties are:
   (a) **Separation of Duties**
   They regulate the mutual exclusiveness of permissions. Section 3.3 investigates different cases of this property.
   (b) **Chinese Wall**
   The Chinese Wall property regulates conflict of interest when accessing different objects. This is discussed in Section 3.4.
5. **Queries**
   Queries related to subjects, permissions and objects are distinguished as

access control queries and administrative queries that we will describe in Section 3.5. *RelBAC* uses the reasoning facilities of the underling DL formalization to answer queries.

In the next sections we will investigate in more details the different kinds of specifications that can be expressed in *RelBAC*. In particular, we will present how *RelBAC* formally expresses the general assignments, the ground assignments and the properties. Finally, we briefly discuss how *RelBAC* can enforce the policies at run-time. Together hierarchies, properties, ground and general assignments form what are called access control policies, the set of rules that regulate the access control in an information system.

3 Using *RelBAC*

3.1 General Assignments

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

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 Assignments

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'.
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. '*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'.
5. '*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'.
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, *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 restrictions rules are important especially in those scenarios where a specific number limit is a key factor for access control.

3.3 Separation of Duties

Security policies in an access control system should satisfy some properties in order to meet the constraints and real world requirements. Separation of Duties

(*SoD*) is an important security property in modern access control systems. It enforces a mutual exclusiveness constraints on either subjects or permissions. In this section, we will formalize the different kinds of *SoD*s used in the security literature.

Definition 2 (Separation of Duties *SoD*).

***SoD*1.** *Mutually Exclusive Positions*[1] *(MEP).*
A 'position' is an organizational role denoting a group of subjects such as employees, managers, CEOs, etc. Given a set of positions $\mathcal{P} = \{P_1, \ldots, P_n\}$, *where each* P_i *is a concept name:*

1. *To enforce that a subject can be assigned to at most one position among the MEP, we write:*

$$P_i \sqcap P_j \sqsubseteq \bot, \ for\ i = 1, \ldots, n-1, \quad j > i.$$

2. *To enforce that a subject can not be assigned to all the positions among the MEP, we write:*

$$\prod_{i=1}^{n} P_i \sqsubseteq \bot.$$

3. *To enforce that a subject can not be assigned more than* m *positions among the MEP* ($m \leq n$), *we write:*

$$\bigsqcup_{i=1}^{C_n^{m+1}} \left(\prod_{j=1}^{m+1} P_{i_j}\right) \sqsubseteq \bot.$$

where C_n^{m+1} *is the binomial coefficient of 'n choose* $m+1$*'.*

***SoD*2.** *Mutually Exclusive Operations (MEO).*
An 'operation' is a kind of permission that subjects may be allowed to perform some 'act' on objects, such as Read, Download, etc. Given a set of operations giving rise to a MEO, $\mathcal{OP} = Op_1, \ldots, Op_n$ *(where each* Op_i *is a DL role name), then, we distinguish two different kinds of MEO:*

1. *To enforce that a subject cannot perform two operations in the MEO, we write:*

$$\exists Op_i.\top \sqcap \exists Op_j.\top \sqsubseteq \bot, \ for\ i = 1, \ldots, n-1, \quad j > i.$$

2. *To enforce that a subject cannot perform two operations in the MEO on the same object, we write:*

$$Op_i \sqcap Op_j \sqsubseteq \bot, \ for\ i = 1, \ldots, n-1, \quad j > i.$$

Note that, also for MEO we can distinguish the same three sub-cases as for the MEP case using similar axioms.

[1] We do not use the word '*Role*' to avoid confusion with the notion of a DL role.

SoD3. *Functional Access (FA).*
If each user in U, has an FA, P, to an object in O, then:

$$U \sqsubseteq (\leq 1\ P.O).$$

SoD4. *Inverse Functional Access (IFA).*
If each object in O, has an IFA, P^-, from an user in U, then:

$$O \sqsubseteq (\leq 1\ P^-.U).$$

We now discuss the above defined *SoD*s. Considering the mutually exclusive positions (MEP), a position is usually defined together with some permissions that those subjects belonging to such a position may perform, e.g.:

$$\begin{aligned} Customer &\sqsubseteq \exists Sign.Check \\ Clerk &\sqsubseteq \exists Cashout.Check \\ Manager &\sqsubseteq \exists Monitor.Check \end{aligned}$$

To enforce that the three positions are mutually exclusive (as in *SoD1.1*) we write $Customer \sqcap Clerk \sqsubseteq \bot$, $Customer \sqcap Manager \sqsubseteq \bot$ and $Manager \sqcap Clerk \sqsubseteq \bot$, meaning that every subject in the domain can be assigned at most one of the three positions. For the *SoD1.2* case, by writing $Customer \sqcap Clerk \sqcap Manager \sqsubseteq \bot$, we enforce that subjects can not be assigned to all positions. Finally, according to the *SoD1.3* case, by writing $(Customer \sqcap Clerk) \sqcup (Customer \sqcap Manager) \sqcup (Manager \sqcap Clerk) \sqsubseteq \bot$, we enforce that subjects can not be assigned to more than 2 positions.

Defining a set of operations as mutually exclusive (MEO), as in the *Sod2.1*, forbids a given user to use any combination of them. For example, if we want to enforce that an user cannot read and update at the same time, then, $\mathcal{OP} = \{Read, Update\}$, and thus, $\exists Read.\top \sqcap \exists Update.\top \sqsubseteq \bot$. We can restrict this constraint to be applied only to users belonging to a certain class. For example, let's assume that *Secretary, Manager, Administrator* are classes of users that can either *Read* or *Update* while the MEO rule applies only on users which are either secretaries or managers (and not on administrators), then, we change the MEO rule as: $(Secretary \sqcup Manager) \sqcap (\exists Read.\top \sqcap \exists Update.\top) \sqsubseteq \bot$. In contrast, the MEO as in *SoD2.2* forbids a given user to use all the MEO operations on the same object. For example, suppose in a scenario of Sales Force Automation[2] (SFA), we want to enforce the MEO rule that to initiate, process, check and archive a given order should not be completed by only one user. Assuming that $\mathcal{OP} = \{Initiate, Process, Check, Archive\}$, then, the policy $Initiate \sqcap Process \sqcap Check \sqcap Archive \sqsubseteq \bot$ restricts any pair (u, o) from belonging to all four operations *Initiate, Process, Check* and *Archive*.

The last two *SoD*s are special cases of cardinality restrictions as described in Section 3.1. An example of a functional access is the case where employees have the right to access a single printer, expressed by the following FA, $Employee \sqsubseteq$

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

($\leq 1\ Access.Printer$). On the other hand, an example of an inverse functional access is the case of a version file that can be updated-by a single user, expressed by the following IFA rule: $Version\text{-}File \sqsubseteq (\leq 1\ Update^{-}.User)$, thus a version file is a classical example of a mutually exclusive accessible resource.

Remark 3. The *SoD1.3* (and the similar one in the MEO case) can require a more refined form as mentioned in [11]. Let n the cardinality of the MEP (MEO) set, then we may require at least k ($k \leqslant n$) users to be involved. Suppose now that we further specify that any of the k users can fulfill at-most m positions (operations). If everyone can fulfill an equal number of positions (operations), then the number m in *SoD1.3* must satisfy the following inequation: $(k-1)*m < n$, i.e., $m \leqslant \lceil n/(k-1) \rceil - 1$. This means intuitively that any $m+1$ of these positions (operations) should not be assigned to one user. Then, the *SoD1.3* (and the corresponding one for *SoD2*) can be refined as:

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

In the MEO example above, given the 4 duties in the SFA scenario, a MEO *SoD* requiring that *at least 3 users should be involved* 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 $C_n^{\lceil n/(k-1)\rceil} = C_4^{\lceil 4/2\rceil} = C_4^2 = 6$.

High Level Security Policies on *SoD*s. Some application domains may require further constraints on *SoD*s like, e.g., they may need to constraint that users participating in a *SoD* are from certain classes and that precise cardinality constraints are also necessary for the task to be accomplished.

Li and Wang [12] studied the requirements for specific attributes of the users in addition to cardinality constraints of each kind of users. An algebra was proposed in [12] to specify complex *SoD* based on these extra constraints, called *high-level* security policies. For example, suppose in a scenario of Sales Force Automation[3] (SFA), we want to enforce the MEO *SoD* that to initiate, process, check and archive an order should not be completed by only one user. Assuming that *Initiate, Process, Check* and *Archive* are four operations forming a MEO set, then the following further high-level policies can be enforced:

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

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

3. Exactly two users, one sales manager and one customer, are involved in the operations.

Policy 1 means that a customer must be involved to initiate the order and a manager must be involved to check the order, while the administrator does not care who processes and archives the order. Policy 2 specifies that only customers and managers can be involved in every operation. Policy 3 is the most strict one 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, the three constraints expressed above for the SFA domain can be formalized as follows, respectively:

1. $Order \sqsubseteq (\geqslant 1\ Initiate^{-}.Customer) \sqcap (\geqslant 1\ Check^{-}.Manager)$,
2. $Order \sqsubseteq \forall Involve.(Customer \sqcup Manager) \sqcap$
   $(\geqslant 1\ Initiate^{-}.Customer) \sqcap (\geqslant 1\ Check^{-}.Manager)$,
3. $Order \sqsubseteq (= 2\ Involve.\top) \sqcap \forall\ Involve.(Customer \sqcup Manager) \sqcap$
   $(= 1\ Initiate^{-}.Customer) \sqcap (= 1\ Check^{-}.Manager)$

where $Involve$ is an operation that is more general than any of the 4 operations composing the SFA task, i.e., $Initiate^{-} \sqcup Process^{-} \sqcup Check^{-} \sqcup Archive^{-} \sqsubseteq Involve$.

3.4 Chinese Wall

The *Chinese Wall* property refers to policies that deals with *Conflict of Interest* (CoI). An example of CoI in the financial world is that of a market analyst working for a financial institution providing corporate business services. Such an analyst must keep the confidentiality of information: she cannot advise a corporation if she has knowledge of plans, status and standing of a competitor corporation. The motivation is to avoid sensitive information to be disclosed to competitor companies through the work of the financial consultants. Thus, suppose that there are three companies competing each other. A consultant may offer advice to any company before she commits with a specific customer. But once a choice is made, she cannot offer advice to the other two competitors.

RelBAC can enforce this property as follows, given a set of resources forming a CoI, $\mathcal{R} = \{A_1, \ldots, A_n\}$, accessible via the permissions $P_1, \ldots, P_m$, then:

$$\exists P_k.A_i \sqcap \exists P_k.A_j \sqsubseteq \bot,\ \text{for } k = 1, \ldots, m,\ i = 1, \ldots, n-1,\ j > i.$$

3.5 Queries as Reasoning Services

Access control policies constitute the body of knowledge against which queries need to be answered. The queries mainly fall into two categories: *access control requests* and *administrative checks*. We will show how the DL formalization is able to capture common queries in the access control domain resorting to the well known DL reasoning services.

Access Control Request. These requests take the form of queries such as '*whether someone is allowed to access something with some permission*' during the system operation. Queries are expressed as logical implications starting from a set, Σ, of *RelBAC* specifications. For example, the following queries

1. '*Is a* `SUBJECT` *u allowed to access an* `OBJECT` *o with* `PERMISSION` *P?*',
2. '*Is a* `SUBJECT` *u allowed to access more than n* `OBJECT`*s in O with* `PERMISSION` *P?*',
3. '*Is a* `SUBJECT` *u allowed to access all the* `OBJECT`*s in O with* `PERMISSION` *P?*',

correspond to the, so called, *instance checking* reasoning service:

$$\Sigma \models P(u, o), \quad \Sigma \models (\geq n+1\ P.O)(u), \quad \Sigma \models (\forall \neg P.\neg O)(u).$$

Administrative Check. From the administration point of view, some queries are so called *intensional*, i.e., whether a particular access control policy holds. This kind of queries check whether an axiom expressing a particular policy is logically implied by the current set Σ of specifications. For example, the following queries

1. '*List the* `SUBJECT`*s allowed to access a given* `OBJECT` *o with* `PERMISSION` *P.*'
2. '*List the* `OBJECT`*s that a* `SUBJECT` *u is allowed to access with* `PERMISSION` *P.*'
3. '*Before adding a policy* `Pol` *check if it is already implied by the current policy.*'
4. '*Check if the addition of a policy* `Pol` *is contradictory w.r.t. the current policy.*'
5. '*Does the addition of a policy* `Pol` *violate some security properties* `Pro`*?*'

correspond to the following well known DL reasoning services:

1&2. *Instance Retrieval*: Retrieve all the individuals u and o s.t. $\Sigma \models (\exists P.\{o\})(u)$ and $\Sigma \models (\exists P^{-}.\{u\})(o)$, respectively.
3. *Logical Implication*: $\Sigma \models$ `Pol`.
4. *Satisfiability Check*: $\Sigma \cup \{\texttt{Pol}\} \models \bot$.
5. If $\Sigma \cup \{\texttt{Pol}\} \models \bot$, then, $\forall$`Pro` $\in \Sigma$ check whether $\Sigma \backslash \{\texttt{Pro}\} \cup \{\texttt{Pol}\} \not\models \bot$.

4 Expressivity of *RelBAC* Vs. Complexity

In this section we investigate the trade-off between expressive power and computational complexity. In particular, we will describe the complexity boundaries of the reasoning services in *RelBAC* depending on the kinds of constraints that we want to impose on the domain of application. In the following, the constraints are grouped according to their complexity class with the assumption that the grouping with higher complexity contains all the constraints from the lower complexity groups.

NP-Complete. The description logic *DL-Lite*$^{\mathcal{N}}_{bool}$ [2] allows to express axioms ($C_1 \sqsubseteq C_2$) where concept expressions have the following syntax:

$$C, D \ ::= \ A \ \mid \ \neg C \ \mid \ C \sqcap D \ \mid \ \geq n\, R$$

where the concept expression $\geq n\, R$ is the so called *unrestricted cardinality restriction* and it is equivalent to $\geq n\, R.\top$[4], while roles can also be inverse. Furthermore, we can express disjointness between two role names (i.e., $R_1 \sqcap R_2 \sqsubseteq \bot$).

Since *DL-Lite*$^{\mathcal{N}}_{bool}$ cannot express hierarchical constraints between roles, then we can just express **Hierarchies** only at the level of `SUBJECT` and `OBJECT` but not at the level of `PERMISSION`. Concerning the general assignments, we can capture both `Domain` and `Range Restrictions` ($\exists P \sqsubseteq U$ and $\exists P^{-} \sqsubseteq O$, respectively). We can express a weaker form of minimum and maximum `Cardinality Restrictions`. For example, assuming that, by a range restriction, we define the range of a permission P to be the objects in C_P then with the axiom $U \sqsubseteq \leq n\ P$ we say that *'users in U are allowed to access (with P) at most n objects in* C_P*'* but we cannot express in *DL-Lite*$^{\mathcal{N}}_{bool}$ different cardinality constraints for different subclasses of C_P.

As for *SoD*s, we can fully capture MEP and MEO (even though, in the latter case, we can represent just simple mutual disjointness between two 'operations' for the *SoD2.2* case). Concerning functionality *SoD*s, as for the case of cardinality restrictions, we can represent the weaker unrestricted case (e.g., the FA case reduces to axioms of the form $U \sqsubseteq \leq 1\ P$).

ExpTime-Complete. The description logic $\mathcal{ALCQI}$ is the sub-language of $\mathcal{ALCQIBO}$ without nominals (i.e., the concept expression $\{a_i\}$) and where roles are just atomic or inverse roles. Using $\mathcal{ALCQI}$ we regain hierarchies over permissions, full cardinality restrictions and `Universal Restrictions`. As for *SoD*s, we can now fully capture both FA and IFA. Furthermore, the *Chinese Wall* can now be represented using $\mathcal{ALCQI}$.

NExpTime-Complete. To fully capture all the constraints in *RelBAC* we need the full power of the description logics $\mathcal{ALCQIBO}$. In particular, w.r.t. $\mathcal{ALCQI}$, we add the possibility to capture the `TAC` rule and what we called *High Level Security Policies*, in particular, the ones concerning the complex axioms over DL roles (i.e., the more involved cases in the *SoD2.2* case). Furthermore, the availability of 'nominals' allow us to fully capture ground assignments and to pose queries involving named individuals.

5 Related Work

Description Logics [3] 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. [4, 6, 8, 9, 17, 18]).

Giunchiglia et al. introduced in [8] the *RelBAC* model together with a domain specific Description Logic as the underlying formalism. *RelBAC* captures, with inclusion axioms, the dynamic hierarchies of the subjects, objects and permissions and provides, with cardinality restrictions, powerful cardinality related

[4] We will denote with $\exists R$ the concept expression $\geq 1\, R$

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 [18] an early attempt was made by 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 Chae and Shiri [4] 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 extended RBAC with the object hierarchy similar to the user hierarchy which facilitates the permission propagation.

Recently, Finin et al. proposed to use OWL[5] language as the underlying formalization of the RBAC model in [6]. They provide two ways to formalize an RBAC role, either 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*s done by Li and Wang in [12]. They propose an algebra to specify the composition of the users that share the duties. In [11] Li et al. modeled the problem of an *SoD* of n duties among k users with a first order logic formula as

$$\forall u_1 ... u_{k-1} \in U((\bigcup_{i=1}^{k-1} auth\_perms_\gamma[u_i]) \not\supseteq \{p_1 ... p_n\}) \tag{2}$$

with universal quantifier on arbitrary $k-1$ users in the space of overall users U. Formula (2) specifies that the collection of all the permissions explicitly/implicitly

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

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 1, 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 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 [5] which is an XML based access control policy language without a formal semantics (such as the logic-based formalization proposed here). Kolovski et al. used DLs 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 Conclusions

In this paper, we showed the formal perspective of a novel access control model, *RelBAC*. A domain specific Description Logic, i.e., $\mathcal{ALCQIBO}$, has been proposed to formalize access control policies about knowledge hierarchies, permission assignments, security properties and queries. While reasoning in $\mathcal{ALCQIBO}$ is NExpTime-complete we provided useful scenarios where sub-languages of the full $\mathcal{ALCQIBO}$ can be used giving rise to lower complexity results.

We studied the different semantics of an important security property, i.e., *SoD*s and provided a formalization for different aspects of *SoD*s. The Chinese Wall property, already introduced in the original *RelBAC* proposal, has been formalized here using $\mathcal{ALCQIBO}$. Last but not least, from the administration point of view, we showed how to formalize access control requests and administrative checks resorting to the well known reasoning services of $\mathcal{ALCQIBO}$.

As a further work we intend to use off the shelves DL reasoners to reason over *RelBAC* specification. The first experiments using general purpose DL reasoners such as Pellet and FaCT++ did not perform well for *RelBAC*. The next steps of this experimental work aim to investigate the efficiency issues along two directions. On one side, we should consider complete algorithms for interesting subsets of *RelBAC* with nicer computational complexities along the lines presented in Section 4. On the other hand, we are interested in investigating uncomplete algorithms that nicely approximate the complete reasoning but with a lower complexity.

References

1. S. Agarwal and B. Sprick. Access control for semantic web services. In *ICWS '04: Proceedings of the IEEE International Conference on Web Services*, page 770, Washington, DC, USA, 2004. IEEE Computer Society.

2. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The DL-Lite family and relations. *Journal of Artificial Intelligence Research (JAIR)*, 36:1–69, 2009.
3. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The description logic handbook: theory, implementation, and applications*. Cambridge University Press, New York, NY, USA, 2003.
4. J.-H. Chae and N. Shiri. Formalization of rbac policy with object class hierarchy. In E. Dawson and D. S. Wong, editors, *ISPEC*, volume 4464 of *Lecture Notes in Computer Science*, pages 162–176. Springer, 2007.
5. O. eXtensible Access Control Markup Language (XACML) TC. http://www.oasis-open.org/committees/tc_home.php? wg_abbrev=xacml.
6. 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.
7. F. Giunchiglia and I. Zaihrayeu. Lightweight ontologies. In *Encyclopedia of Database Systems*. Springer, 2008.
8. F. Giunchiglia, R. Zhang, and B. 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. F. Giunchiglia, R. Zhang, and B. Crispo. Ontology driven community access control. In *SPOT2009 - Trust and Privacy on the Social and Semantic Web*, 2009.
10. V. Kolovski, J. Hendler, and B. 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. N. Li, M. V. Tripunitara, and Z. Bizri. On mutually exclusive roles and separation-of-duty. *ACM Transactions on Information System Security*, 10(2):5, 2007.
12. N. Li and Q. 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 F. Wolter, H. Wansing, M. de Rijke, and M. 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. I. 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. R. Zhang. *RelBAC: Relation Based Access Control*. PhD thesis, University of Trento, March 2009.
18. C. Zhao, N. Heilili, S. Liu, and Z. Lin. Representation and reasoning on rbac: A description logic approach. In D. V. Hung and M. Wirsing, editors, *ICTAC*, volume 3722 of *Lecture Notes in Computer Science*, pages 381–393. Springer, 2005.