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

REASONING ABOUT RELATION BASED ACCESS CONTROL

Alessandro Artale, Bruno Crispo,
Fausto Giunchiglia, Fatih Turkmen
and Rui Zhang

June 2010

Technical Report # DISI-10-040

Also: in the proceedings of the 4th International Conference on Network and System Security (NSS 2010), Melbourne, Australia, September 1-3, 2010.

Reasoning about Relation Based Access Control

Alessandro Artale*, Bruno Crispo†, Fausto Giunchiglia†, Fatih Turkmen† and Rui Zhang‡

*KRDB Research Centre
Free University of Bozen-Bolzano, Italy
Email: {artale}@inf.unibz.it

†DISI, University of Trento, Italy
Email: {crispo,fausto,turkmen}@disi.unitn.it

‡CCST, Jilin University, China
Email:{rui}@jlu.edu.cn

***Abstract*—Relation Based Access Control (*RelBAC*) is an access control model that places *permissions* as first class concepts. 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 are studied and formally represented in *RelBAC*. To meet the needs of automated tools for administrators, we show that *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

I. 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 error-free specification of 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 the organizational textbooks describe very clean organizational models with well defined responsibilities, the real world is quite different. Many organizations find it difficult, if not impossible, to identify a single point of coordination for policy changes and control of their consistency. Furthermore, for non-trivial information systems with thousands of policies, policy management requires automated tools to lower the incidence of human errors.

To address such new challenges, security researchers have recently proposed new approaches [1], [2] with the aim of offering richer expressivity in terms 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* [3]. 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). An aspect that however has not yet been fully investigated is the ability of *RelBAC* to offer a set of powerful services (e.g., Separation of Duty) to help administrators in doing their job. This paper fills this gap by describing *RelBAC*'s reasoning abilities due to its formalization in description logic (DL).

In general the use of well established reasoners has two important advantages. First, it eases the policy management so that while the policies change, inconsistencies are not introduced or general properties are not violated. Such a reasoning service is typically used for checking the correctness of the policies during the system design phase and thus it runs off line. Second and more importantly, it allows to rigorously verify dynamic conditions at run-time (e.g., dynamic separation of duties or Chinese Wall constraints). The paper defines the reasoning services available in *RelBAC* and the different types of policy management problems for which 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 organized as follows. Section II specifies the *RelBAC* model while Section III introduces the Description Logic $\mathcal{ALCQIBO}$. Section IV presents in detail the formalization of policies, properties and queries using the DL $\mathcal{ALCQIBO}$. Section V summarizes the complexity results for the corresponding expressiveness of the adopted representation language. We discuss the related works in Section VII and conclude in Section VIII with final remarks.

II. The *RelBAC* Approach to Access Control

Relation-based Access Control (*RelBAC*) is an access control model introduced in [3]. As shown in the ER Diagram of

Figure 1, what distinguishes *RelBAC* from other access control models is the way it models permissions. A PERMISSION is modeled as an operation that users (SUBJECTs) can perform on certain resources (OBJECTs). To capture this intuition, a PERMISSION is named with the name of the operation it refers to, e.g., *Write*, and *Read* operation[1]. The *generalization*

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

(loops) on each component represents IS−A relations. Not only SUBJECT and OBJECT are organized along IS−A hierarchies but also PERMISSION. Thus, by imposing the constraint that '*Update is more powerful than Read*' *RelBAC* allows those people with the permission *Update* to have the permission *Read*, too.

A. Why RelBAC?

Role Based Access Control (RBAC) has been the dominant access control model [4] in many applications. It has been standardized and integrated into major software products. On the other hand, it has been frequently criticized [5] in terms of expressivity and support of core security principles.

RelBAC addresses the above limitations by allowing to express access control policies and the related properties (e.g., separation of duty) in a simple way. It uses its underlying logic to reason about such properties. In particular

- *RelBAC* supports strong cardinality features. Quantifiers have been very successfully used in databases, but in access control often policies are implicitly universally quantified. By using quantifiers we can express an access control rule which states that *students should be able to use at least one PC*. In principle, the rule allows any student to use all PCs. However, what really matters here is that she has access to at least one. The above policy could be made stronger. For example, by using cardinalities we can state, e.g., that a student can have access to exactly one PC. On the other hand, by using the universal quantifier, we can state, e.g., that a student can only use PCs forbidding the use of other means (e.g., they cannot use personal assistants). A similar behavior can be obtained in the existing models, e.g., RBAC, by checking these constraints at *run-time*.
- Using cardinalities, important properties such as Separation of Duty (SoD, as defined by Li et al. [6]) can be easily expressed with access control rules. In *RelBAC*, these properties sit on top of the model rather than being part of the model as in RBAC. Any change in the rule or the high level policy that enforce the property can be addressed without modifying the core model. As a result, we can reason on them (as shown in Section IV-C) in isolation.

[1]In *RelBAC* we adopt the convention that PERMISSION names are capitalized verbal forms.

- RelBAC splits subjects from objects by defining permissions as relations between them. The role of users and objects is completely symmetric and one can seamlessly define user-centric (e.g., all senior managers can write the file F) or object-centric policies (e.g., the file F can be read only by senior managers).

While these are main benefits of *RelBAC*, further motivations are provided in Section VI.

III. The Description Logic $\mathcal{ALCQIBO}$

The logic $\mathcal{ALCQIBO}$ extends the description logic $\mathcal{ALC}$ [7] with qualified cardinalities, inverse roles, nominals and Boolean for roles (see [8], [9], [10] 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:

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

An $\mathcal{ALCQIBO}$-interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ is said to be 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 [11]. On the other hand, Boolean modal logic is

a proper sub-language of $\mathcal{ALCQIBO}$ and it is NExpTime-complete [10]. Summing up, reasoning in $\mathcal{ALCQIBO}$ is NExpTime-complete.

IV. The *RelBAC* Logic

In *RelBAC* we distinguish 5 different kinds of specifications that, altogether, constitute 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) **Hierarchy:** In the access control domain, `SUBJECT`, `OBJECT` and `PERMISSION` are organized in a taxonomy along the `IS-A` relation [12]. 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 IV-A. 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 IV-B. At this level, constraints on access control are expressed at the level of single domain instances.

4) **Properties**: They specify 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 IV-C 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 IV-D.

5) **Query**: Queries related to subjects, permissions and objects are distinguished as access control queries and administrative queries that we will describe in Section IV-E. *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.

A. 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 only the objects in O with P*'
   '*Objects in O can be accessed only from users in U with P*'.
   We can represent these constraints 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`

B. 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 diverse access control rules with a single DL axiom. 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 restriction rules are important especially in those scenarios where a specific number limit is a key factor for access control.

C. Property 1: Separation of Duties

Separation of Duties (*SoD*) is an important security property for modern access control systems. It enforces, independently of the underlying access control model, a mutual exclusiveness constraint on either subjects or permissions. In this section, we will formalize different types of *SoD*s used in the security literature.

Definition 2 (Separation of Duties SoD*):*

*SoD*1. Mutually Exclusive Positions[2] (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:
$$\exists Op_i.\top \sqcap \exists Op_j.\top \sqsubseteq \bot,\ for\ i = 1, \ldots, n-1,\ j > i$$
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, in this case, we can further distinguish the three sub-cases as for the MEP case using similar axioms.

*SoD*3. Functional Access (FA).
If each user in U, has an FA, P, to an object in O, then:
$$U \sqsubseteq (\leq 1\ P.O)$$

*SoD*4. 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 defined together with some permissions that the subjects belonging to the position may perform, e.g.:
$$Customer \sqsubseteq \exists Sign.Check$$
$$Clerk \sqsubseteq \exists Cashout.Check$$
$$Manager \sqsubseteq \exists Monitor.Check$$
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

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

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 (in this example, such constraint is equivalent to the *SoD1.1* constraint).

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 a 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 us 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[3] (SFA), we want to enforce the MEO rule that initiate, process, check and archive a given purchase 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 IV-A. An example of a functional access is the case where employees have the right to access a single printer, expressed by the following FA, $Employee \sqsubseteq (\leq 1\ Access.Printer)$. 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 1: The *SoD1.3* (and the similar one in the MEO case) can require a more refined form as mentioned in [6]. 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:

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

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

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

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

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

Li and Wang [1] studied the requirements for specific attributes of the users in addition to cardinality constraints of each kind of users. An algebra, Separation of Duty Algebra (SoDA), has been proposed to specify complex *SoD* policies based on these extra constraints, called *high-level* security policies. In addition to *SoD2.2*, the following further high-level policies can be enforced with *RelBAC*:

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

RelBAC can achieve these 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., $Initiate^{-} \sqcup Process^{-} \sqcup Check^{-} \sqcup Archive^{-} \sqsubseteq Involve$.

D. Property 2: Chinese Wall

The *Chinese Wall* property refers to policies that deal 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. A market analyst must keep the certain financial information confidential: 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 financial consultants. Thus, suppose that there are three companies competing with 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:

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

E. 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.

1) 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:

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

2) 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*: $\Sigma \cup \{$`Pol`$\} \models \bot$.
5. If $\Sigma \cup \{$`Pol`$\} \models \bot$, then, $\forall$`Pro` $\in \Sigma$ check whether $\Sigma \backslash \{$`Pro`$\} \cup \{$`Pol`$\} \not\models \bot$.

V. 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*$^{\mathcal{N}}_{bool}$ [13], [14] 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 *hierarchical constraints* only at the level of `SUBJECT` and `OBJECT` but not at the level of `PERMISSION`. Concerning *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

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

$\mathcal{ALCQIBO}$. 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.

VI. Discussion

The strength of RelBAC lies in the rich support of security property specification. Recently, [6] showed that RBAC's core mechanism, statically mutually exclusive role (SMER)[5], in supporting SoD is insufficient. They showed that the enforcement of SoD is intractable (coNP-complete) while RBAC's SoD approach (i.e., SMER constraints) can be enforced in PTime. Li et al. modeled the problem of an *SoD* policy that consists of n duties among k users with a first order logic formula as

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

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 assigned to this $k-1$ users should not be a superset of all the n steps of duties. Their approach has the complexity of $(|U|^{k-1} * n)$ which explodes to the cardinality of the subject space. In *RelBAC*, as shown in Formula 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 is 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 ($|U|$), our method is more efficient than [6].

In a further paper, the authors proposed SoDA for the specification of SoD constraints in which each term (e.g., Manager) refers to a user set (e.g., {Alice, Bob}) from UA relations of RBAC $\langle U, UR \rangle$. A high level policy putting SoD constraints on a payment transaction with two operations (*Create*, *Sign*) can be specified as follows: $(Employee \cup Manager) \otimes Manager$. This policy enforces that the two steps on both sides of the operator $\otimes$ must be completed by different users. However, SoDA provides only a user-based SoD and it does not allow the specification of properties based on actions or objects. On the other hand, *RelBAC* allows to specify SoDs based on different perspectives. As an example, a RelBAC expression on the disjointness of actions can be specified as follows:

$$\begin{aligned} Transaction &\sqsubseteq \exists Create^-.(Employee \sqcup Manager) \sqcap \\ &\quad \exists Sign^-.Manager \\ Create \sqcap Sign &\sqsubseteq \bot \end{aligned}$$

[5]There is also a dynamic version, dynamically mutually exclusive roles (DMER), but it does not enforce SoD[6].

VII. Related Works

Description Logics [7] arouse the interests in the AI community for their expressiveness and decidability of the reasoning services. Various papers describe the use of DL to formalize access control models and use state of the art DL reasoners to formalize security properties and check their consistency (see, e.g., [15], [16], [3], [17], [18], [19]).

Giunchiglia et al. introduced in [3] 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 specifications of access control rules. The theory of Lightweight Ontology [12] can be used to model the social community into ontologies as discussed in [17] and can facilitate the management of knowledge hierarchies and access control rules.

In [19] 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 [15] where an *operation* is represented by a DL role. Their system has several drawbacks, and in particular:

- Misuse 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. IV-A).
- 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, Finin et al. proposed to use OWL[6] language as the underlying formalization of the RBAC model in [16]. They provide two ways to formalize an RBAC role, either as a class or as an attribute. N3Logic is used together with DL

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

subsumption reasoning. Authorization decision queries can be answered using DL reasoners in their system.

An existing industry standard is XACML [20] 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 [21]. *RelBAC*, in contrast, is not only a new access control model, but it comes with a well defined syntax and semantics to express access control policies together with the additional facility to use logic-based reasoning services to help management and concrete use of such policies.

VIII. 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 also 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-shelf DL reasoners to reason over *RelBAC* specification. The preliminary results of our experiments using general purpose DL reasoners such as Pellet and FaCT++ showed that *RelBAC* has a reasonable performance. The next steps of this experimental work aim to investigate the efficiency issues along two directions. On the one hand,we should consider complete algorithms for interesting subsets of *RelBAC* with nicer computational complexities along the lines presented in Section V. On the other hand, we are interested in investigating incomplete algorithms that nicely approximate the complete reasoning but with a lower complexity.

A. Acknowledgement

This work has been supported by funds from the European Commission (contract No 216917 for the FP7-ICT-2007-1 project MASTER).

References

[1] N. Li and Q. Wang, "Beyond separation of duty: An algebra for specifying high-level security policies," in *Journal of the ACM (JACM)*, vol. 55, no. 3. ACM, 2008, pp. 1–46.

[2] L. Hu, S. Ying, X. Jia, and K. Zhao, "Towards an approach of semantic access control for cloud computing," in *Cloud Computing*, vol. 5931/2009. Springer, 2009, pp. 145–156.

[3] 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*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 3–11.

[4] D. F. Ferraiolo, R. S. Sandhu, S. I. Gavrila, D. R. Kuhn, and R. Chandramouli, "Proposed NIST standard for role-based access control," *Information and System Security*, vol. 4, no. 3, pp. 224–274, 2001.

[5] N. Li, J.-W. Byun, and E. Bertino, "A critique of the ANSI standard on role-based access control," *IEEE Security and Privacy*, vol. 5, no. 6, pp. 41–49, 2007.

[6] N. Li, M. V. Tripunitara, and Z. Bizri, "On mutually exclusive roles and separation-of-duty," *ACM Transactions on Information System Security*, vol. 10, no. 2, p. 5, 2007.

[7] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, Eds., *The description logic handbook: theory, implementation, and applications*. New York, NY, USA: Cambridge University Press, 2003.

[8] R. A. Schmidt and D. Tishkovsky, "Using tableau to decide expressive description logics with role negation," in *6th International Semantic Web Conference, 2nd Asian Semantic Web Conference, ISWC + ASWC*, ser. Lecture Notes in Computer Science, vol. 4825, 2007, pp. 438–451.

[9] C. Lutz and D. Walther, "Pdl with negation of atomic programs," *Journal of Applied Non-Classical Logic*, vol. 15, no. 2, pp. 189–214, 2005.

[10] C. Lutz and U. Sattler, "The complexity of reasoning with boolean modal logics," in *Advances in Modal Logics Volume 3*, F. Wolter, H. Wansing, M. de Rijke, and M. Zakharyaschev, Eds. CSLI Publications, Stanford, 2001.

[11] I. Pratt-Hartmann, "Complexity of the two-variable fragment with counting quantifiers," *J. of Logic, Lang. and Inf.*, vol. 14, no. 3, pp. 369–395, 2005.

[12] F. Giunchiglia and I. Zaihrayeu, "Lightweight ontologies," in *Encyclopedia of Database Systems*. Springer, 2008.

[13] A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev, "The DL-Lite family and relations," in *Journal of Artificial Intelligence Research*, 2009, pp. 1–69.

[14] ——, "*DL-Lite* in the light of first-order logic," in *Proc. of the 22nd Nat. Conf. on Artificial Intelligence (AAAI-07)*, 2007.

[15] J.-H. Chae and N. Shiri, "Formalization of RBAC policy with object class hierarchy." in *ISPEC*, ser. Lecture Notes in Computer Science, E. Dawson and D. S. Wong, Eds., vol. 4464. Springer, 2007, pp. 162–176.

[16] 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*. New York, NY, USA: ACM, 2008, pp. 73–82.

[17] F. Giunchiglia, R. Zhang, and B. Crispo, "Ontology driven community access control," in *SPOT2009 - Trust and Privacy on the Social and Semantic Web*, 2009.

[18] R. Zhang, "Relbac: Relation based access control," Ph.D. dissertation, University of Trento, March 2009.

[19] C. Zhao, N. Heilili, S. Liu, and Z. Lin, "Representation and reasoning on rbac: A description logic approach," in *ICTAC*, ser. Lecture Notes in Computer Science, D. V. Hung and M. Wirsing, Eds., vol. 3722. Springer, 2005, pp. 381–393.

[20] O. eXtensible Access Control Markup Language (XACML) TC, http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml.

[21] 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*. New York, NY, USA: ACM, 2007, pp. 677–686.