An Ontology-Based Approach for Handling Inconsistency in Explainable and Prioritized Access Control Models

Ahmed Laouar, Toky Raboanary, Salem Benferhat

An Ontology-Based Approach for Handling Inconsistency in Explainable and Prioritized Access Control Models

Ahmed Laouar[1][0009−0002−0028−3234], Toky Raboanary[2][0000−0001−6133−4643], and Salem Benferhat[1][0000−0002−4853−3637]

[1] CRIL, Univ. Artois & CNRS, Lens, France {laouar,benferhat}@cril.fr
[2] University of Cape Town, Cape Town, South Africa traboanary@cs.uct.ac.za

Abstract. The development of secure and efficient solutions for access control is an important issue in a variety of applications. One of the main challenges is to avoid situations that make access control decision-making impossible. However, avoiding such situations hampers the evolution of the model, as it means either adding a large set of constraints or dealing with each conflict situation. It is, therefore, important to explore methods that deal with conflicts as they arise while providing explanations of the decision taken. In this work, we develop an ontology to manage dynamic and abstract access control rules based on the OrBAC (Organization-Based Access Control) model and integrate an ordering relation over instances of the ontology. Our method takes advantage of the application of inconsistency-tolerant semantics to resolve conflicts and generate explanations for transparency in decisions made. Our results show that the approach efficiently preserves the consistency of the decision taken and provides potentially useful and human-friendly explanations.

Keywords: Access Control · Ontology · Inconsistency · Explainability.

1 Introduction

In computer security, various mechanisms are needed to make sure that information systems are properly accessed by authorized users. Some of these mechanisms focus on user authentication and identification, while other mechanisms focus on the modelling and the specification of access rights according to the required security policy [19]. Access control models are the main mechanisms to achieve this objective. They serve as a tool of implementing and enforcing security policies, by providing an access decision (grant or deny) based on some predefined access control security rules. A lot of research efforts focused on proposing different types of models [15,19], like the role-based model (RBAC) [21] and the organisation-based model (OrBAC) [16].

In this paper, we focus on dealing with conflicts that arise in generic and dynamic access control models. We chose the OrBAC model, mainly because it is a dynamic model that has the capacity of inferring access rights and also

because uncertainty in the model is natural and can be represented in multiple ways, like using possibilistic logic weights [2]. We propose to deal with conflicts on their occurrence rather than defining rules and constraints to avoid them as it is usually used in the literature [8].

This can be achieved by applying inconsistency-tolerant semantics [4], which are the formal way to reason in the presence of conflicts in ontologies [5].

Therefore, OrBAC must be adapted (or encoded) in some formal ontology fragment in order to apply such methods. More precisely, the main contributions of this paper are as follows:

- we provide an ontology that encodes the OrBAC model, alongside the use of hierarchies and a specific representation of uncertainty;
- we show how a partial order is propagated in the ontology using SWRL rules;
- we apply repair methods inspired from partially ordered possibilistic knowledge base repairs [17] to handle conflicts;
- we provide explanations for the model decisions, with and without conflicts;
- we show that reasoning services are done with simple SPARQL queries.

Section 2 provides a refresher on OrBAC. Section 3 introduces the OrBAC ontology and the representation of uncertainty. Section 4 introduces an efficient conflict handling method. Section 5 provides mechanisms to generate explanations of the decisions taken, before concluding the paper.

2 Refresher on OrBAC

The basic concepts in access control are subjects, which represent users, objects, which are passive entities like resources and files, and actions which are the basic operations performed by a subject on an object, like read and write. OrBAC [16] considers these concepts as concrete entities and abstracts each of them into an abstract entities. Subjects that fulfill same functions are grouped in roles, objects that satisfy common properties are grouped in views and actions that share the same principles are grouped in activities. In addition, OrBAC considers organisations as concrete concepts, and always defines rules in an organisation.

Concrete entities are linked to abstract entities using the connection relations Employ, Use and Consider. Employ(org,s,r) means that the subject "s" plays the role "r" within the organization "org". Use(org,o,v) means that the object "o" is used in the view "v" within the organization "org". Consider(org,α,a) means that the organization "org" considers that the action "α" fall within the activity "a". Moreover, OrBAC is said to be dynamic because it uses contexts. A privilege may be granted in a context and denied in another. It can represent environmental variables, like timeOfDay, or a particular state like emergency. Contexts are defined between a subject, an action and an object in an organisation using the relation Define such that Define(org,s,α,o,c) means that in the organization "org", the context "c" holds between the subject "s", the object "o" and the action "α".

Policy rules, which are also called abstract privileges, are defined using the relations Permission, Prohibition, Obligation and Recommendation. Each rule admits five arguments; Permission(org, r, a, v, c) means that the organization "org" grants

the role "r" the permission to perform the activity "a" on the view "v" if the context "c" holds. The remaining rules are defined similarly.

Access rights, or concrete privileges, are derived for the concrete entities, namely subjects, actions and objects. They are represented with the relations Is-permitted, Is-prohibited, Is-obliged, and Is-recommended. The concrete privilege Is-permitted(s,α,o) means that the subject "s" is allowed to perform the action "α" on the object "o". The derivation is given as follows:

Definition 1 (Derivation of concrete rules). *Concrete permissions are derived from an OrBAC policy using the following rule:*
$\forall$org, $\forall$s, $\forall\alpha$, $\forall$o, $\forall$r, $\forall$a, $\forall$v, $\forall$c,
Permission(org,r,a,v,c) $\wedge$ Employ(org,s,r) $\wedge$ Use(org,o,v) $\wedge$ Consider(org, α, a) $\wedge$ Define(org,s,α,o,c) $\vDash$ Is-permitted(s,α,o)
The privileges Is-prohibited, Is-obliged, and Is-recommendedare derived similarly.

3 The OrBAC ontology

3.1 Preliminaries

An ontology can be encoded in OWL [9] using multiple syntaxes. In this paper, RDF triples are used to represent the axioms of the proposed ontology, in addition to some properties from OWL. Note that OWL is not limited to the elements described in this paper, as it allows representing other notions like complex concepts, subproperties, etc. The ontology is constructed from finite sets of classes C, object properties P and individuals I. The class triples are of the form $\langle x$,rdf:type,owl:Class$\rangle$, where $x \in C$ and relationships between classes are of the form $\langle x$,rdfs:subClassOf$, y\rangle$, where $x, y \in C$. Object property triples have the form $\langle x$,rdf:type,owl:ObjectProperty$\rangle$, where $x \in P$ and $\langle x, p, y\rangle$ s.t. $p \in \{$rdfs:domain,rdfs:range$\}$, $x \in P$ and $y \in C$. Transitive object property triples are of the form $\langle x$,rdf:type,owl:TransitiveProperty$\rangle$. It should be noted that properties like TransitiveProperty can be encoded in RDF triples, but are not part of RDF/RDFS. Axioms about individuals are called assertions, and have the form $\langle x, p, y\rangle$, where $x \in I$ is an individual, p is a property and y can be an individual or a literal. In addition, Semantic Web Rule Language (SWRL) rules [13] and SPARQL [12] queries are used to implement reasoning services: SWRL rules can be represented in the form $\varphi \rightarrow \phi$, where φ and ϕ are conjunction of atoms involving classes and object properties. SPARQL ASK queries check for the existence of triples that match a specified query pattern and return True if the pattern matches, and False otherwise. Then, SELECT queries retrieve specific data based on the provided patterns, and the result is a set of variable bindings matching the patterns. UNION queries combine results of multiple patterns, and the result set includes all matches from each pattern, eliminating duplicates by default.

3.2 Ontology classes and object properties

Representing OrBAC concrete and abstract entities. Objects, subjects and actions are represented using the classes orbac:Object, orbac:Subject and or-

bac:Action, respectively. The orbac:Organisation class represents organisations in the model, and is also a sub class of the orbac:Subject class. They are all sub classes of the class orbac:ConcreteConcept. Activities, views and roles are represented using the classes orbac:Activity, orbac:View and orbac:Role, respectively. They are all sub classes of the class orbac:AbstractConcept. Moreover, the orbac:Context class represents contexts.

Representing policy rules. The abstract policy rules, called also access types in the following, are represented using the orbac:Permission, orbac:Prohibition, orbac:Obligation and orbac:Recommendation classes, which are all sub classes of the class orbac:AccessType. Each access type is connected with its constructing elements using the object properties: orbac:accessTypeOrg to link an access type with an organisation. orbac:accessTypeRole to link an access type with a role. orbac:accessTypeActivity to link an access type with an activity. orbac:accessTypeView to link an access type with a view. orbac:accessTypeContext to link an access type with a context. Each access type instance must be connected with exactly one entity for each of the above object properties.

Representing Connection rules. The rules linking concrete entities with their abstractions, and the rule that defines whether a context holds or not for a subject, an action and an object in an organisation are represented using the classes orbac:Consider, orbac:Use, orbac:Employ and orbac:Define, which are all sub classes of orbac:ConnectionRule. These rules are connected with their entities using the following object properties. The class orbac:Consider uses the orbac:considersActivity, orbac:considersAction and orbac:considersOrg object properties to connect its instances to an activity, an action and an organisation respectively. The class orbac:Use uses the object properties orbac:usesView, orbac:usesObject and orbac:usesEmployer to connect its instances to a view, an object and an organisation, respectively. The class orbac:Employ uses the orbac:employesRole, orbac:employesEmployee and orbac:employesEmployer object properties to connect its instances to a role, a subject and an organisation, respectively. The class orbac:Define uses the orbac:definesSubject, orbac:definesAction, orbac:definesObject and orbac:definesContext properties to connect its instances to a subject, an action, an object and a context, respectively.

Running example. In collaborative research projects, sharing resources involves access to sensitive data by users from multiple organisations, which requires proper access control mechanisms. We consider an example of a consortium of universities. The consortium defines a permission in the secondment context, a secondee can perform modification to the reports view, which contains secondment reports. It also defines a prohibition for a staff member to not modify the same reports, in the default context. Moreover, the consortium considers that the action edit falls under the activity modify and the file report1 is used within the view reports. Assume a user called Bob, who is employed within the

```
:perm1 rdf:type orbac:Permission.
:perm1 :accessTypeOrg :consortium.
:perm1 :accessTypeContext :secondment.
:perm1 :accessTypeActivity :modify.
:perm1 :accessTypeView :reports.
:perm1 :accessTypeRole :secondee.

:emp1 rdf:type orbac:Employ.
:emp1 :employesEmployer :univ1.
:emp1 :employesRole :secondee.
:emp1 :employesEmployee :Bob.

:use1 rdf:type orbac:Use.
:use1 :usesEmployer :univ1.
:use1 :usesView :reports.
:use1 :usesObject :report1.

:def2 rdf:type orbac:Define.
:def2 :definesOrg :univ1.
:def2 :definesContext :secondment.
:def2 :definesSubject :Bob.
:def2 :definesAction :edit.
:def2 :definesObject :report1.
```

```
:prohi1 rdf:type orbac:Prohibition.
:prohi1 :accessTypeOrg :consortium.
:prohi1 :accessTypeContext :default.
:prohi1 :accessTypeActivity :modify.
:prohi1 :accessTypeView :reports.
:prohi1 :accessTypeRole :staffMember.

:emp2 rdf:type orbac:Employ.
:emp2 :employesEmployer :consortium.
:emp2 :employesRole :staffMember.
:emp2 :employesEmployee :Bob.

:cons1 rdf:type orbac:Consider.
:cons1 :considersOrg :consortium.
:cons1 :considersActivity :Modify.
:cons1 :considersAction :edit.

:def1 rdf:type orbac:Define.
:def1 :definesOrg :consortium.
:def1 :definesContext :default.
:def1 :definesSubject :Bob.
:def1 :definesAction :edit.
:def1 :definesObject :report1.
```

Fig. 1: Running example triples.

consortium as a staff member. In addition, Bob is on a secondment in univ1, thus they are employed as a secondee there. The context default holds between Bob, edit and report1, in the consortium, since Bob is a staff member. Moreover, the context secondment also holds for Bob, edit and report1 since Bob is on a secondment. The representation of this example using the proposed orbac ontology is given in Figure 1.

Hierarchies in the OrBAC ontology. In access control models, hierarchies define how privileges are inherited in the model. We propose the use of hierarchies of organisations and roles, as defined in [7]. For roles, two types of relationships are implemented: a sub-role inherits both permissions and prohibitions of a parent role, and a senior-role inherits permissions of a parent role and the parent role inherits its prohibitions. This separation results from the difference in significance of a relationship between roles. A senior role is considered as more privileged, thus inheritance goes upward.

- orbac:SubRole and orbac:SeniorRole classes as sub classes of the orbac:Role class.
- The orbac:hasParent property links a sub-role or a senior-role to a parent role.
- Sub-roles and senior-roles are defined in an organisation using the property orbac:subRoleOrg, which links a sub-role or a senior-role to an organisation.
- Organisational hierarchy is represented using the orbac:subOrganisationOf object property, which links two organisations. Every rule or relation that is defined in a parent organisation holds for the sub-organisation, for example: if $\langle$perm1, orbac:accessTypeOrg, consortium$\rangle \wedge \langle$univ1, orbac:subOrganisationOf, consortium$\rangle \rightarrow \langle$perm1, orbac:accessTypeOrg, univ1$\rangle$. The inverse does not hold.

We use SPARQL queries to check if a rule is inferred from the hierarchy, without adding the inferred axioms (triples) to the ontology.

Example 1. The following are examples of role hierarchies: the three triples on the left indicate that a secondee is a sub-role of employee in univ1. Similarly, on the right, the consortium considers that an employee is a sub-role of staffMember.

```
:secondee rdf:type orbac:subRole.
:secondee orbac:hasParent :employee.
:secondee orbac:subRoleOrg :univ1.
```

```
:employee rdf:type orbac:subRole.
:employee orbac:hasParent :staffMember.
:employee orbac:subRoleOrg :consortium.
```

3.3 Representation of uncertainty

The representation of priorities in OrBAC has been widely studied [3,8]. However, the proposed solutions focus on providing a preference of application between policy rules. The automatic assignment of preferences has been discussed in [8], but the authors did not elaborate on how it can be achieved. In this paper, a new representation of uncertainty is suggested, making the following separation:
- Fully certain rules: the policy rules (or access types) are established by policy experts and must be enforced, hence, they are given the highest priority.
- Uncertain rules: are the rules connecting concrete and abstract concepts.

Uncertainty is encoded using a preference relation defined between the individuals of the connection relations, namely, Employ, Consider, Use, and Define. Policy rules are assumed to be fully certain. Defining a preference between individuals of the uncertain rules is achieved using the transitive object property $\langle x,$ orbac:isPreferredTo, $y\rangle$, which means x is more preferred to y. Equivalence holds when both $\langle x,$ orbac:isPreferredTo, $y\rangle$ and $\langle y,$ orbac:isPreferredTo, $x\rangle$ hold and incomparability is the absence of both relations.

The following describes how preferences are automatically assigned to the connection rules. The main method proceeds by ordering abstract concepts (e.g. roles). The order is then propagated using SWRL rules. Rule R1 in Table 1 is an example on how ordering is propagated from orbac:Role individuals to orbac:Employ individuals, other rules are defined similarly. Another method relies on hierarchies to define preferences. Mainly, a sub-role and a senior-role are considered more preferred than their parent roles. In addition, rules defined in sub-organisations have a higher preference. Rules R2 and R3 in Table 1 are examples of rules used to propagate ordering from hierarchies. All the rules are provided in the appendix. SWRL rules are executed efficiently using Drools [20].

Example 2. In our example, if we have $\langle$secondment, orbac:isPreferredTo,default$\rangle$ and $\langle$secondee, orbac:isPreferredTo, staffMember$\rangle$, then the SWRL rules result in $\langle$emp1, orbac:isPreferredTo, emp2$\rangle$ and $\langle$def2, orbac:isPreferredTo, def1$\rangle$. In addition, $\langle$univ1, orbac:subOrganisationOf, consortium$\rangle$, $\langle$use1, orbac:usesEmployer, univ1$\rangle$ and $\langle$emp2, orbac:employesEmployer, consortium$\rangle$, result in $\langle$use1, orbac:isPreferredTo,emp2$\rangle$.

Definition 2. *An* orbac *KB is a tuple* $\mathcal{K} = \langle \mathcal{T}, \mathcal{R}, \mathcal{T}_I \rangle$ *s.t.* $\mathcal{T}$ *is a finite set of axioms of the* orbac *ontology.* $\mathcal{R}$ *is a finite set of* SWRL *order propagation rules.* $\mathcal{T}_I$ *is a finite set of assertions (dataset) and* I *is the set of individuals.*

Table 1: SWRL rules for order propagation. From R2. consider that r1 and r2 are roles, if r2 is a parent role of r1, then r1 is preferred to r2.

R1: orbac:isPreferredTo(?role1,?role2) ∧ orbac:employesRole(?employ1,?role1) ∧ orbac:employesRole(?employ2,?role2) → orbac:isPreferredTo(?employ1,?employ2)
R2: orbac:hasParent(?role1, ?role2) → orbac:isPreferredTo(?role1, ?role2)
R3: orbac:usesEmployer(?use1, ?org1) ∧ orbac:employesEmployer(?employ2, ?org2) ∧ orbac:subOrganisationOf(?org1, ?org2) → orbac:isPreferredTo(?use1, ?employ2)

In the following, $\mathcal{T}_I = \mathcal{T}_U \cup \mathcal{T}_F$ *is used to separate the assertions involving uncertain* (U) *and fully certain* (F) *individuals.* $\mathcal{T}_U = \{\langle x, p, y\rangle \in \mathcal{T}_I | x = i \text{ or } y = i \text{ s.t. } \langle i, \mathsf{rdf:type}, t\rangle \in \mathcal{T}_I \text{ and } t \in U\}$*, where* $U = \{$orbac:Employ*,* orbac:Consider*,* orbac:Use*,* orbac:Define$\}$*.*

The derivation of access rights, given by Definition 1, is encoded using SPARQL queries, called Is-permitted(s, α, o) and Is-prohibited(s, α, o), for a concrete permission and prohibition, respectively, where s, α, o are a subject, action, and object, respectively. Due to space limitations, the orbac ontology, alongside all the queries used in this paper are provided in the appendix [3].

Example 3. In the running example, both Is-permitted(Bob,edit,report1) and Is-prohibited(Bob,edit,report1) return true.

4 Inconsistency handling

The main objective of encoding OrBAC as an ontology is to allow inconsistency handling as defined for formal ontologies [4]. The separation between fully certain and uncertain rules plays a key role in this application. In this section, we introduce the different reasoning services applied to the orbac ontology. Then, we discuss the application of partially ordered possibilistic repairs [17] to it.

4.1 Consistency checking and computing conflicts

In access control, a knowledge base is considered as inconsistent if both a permission and a prohibition are inferred for the same subject, object and action. In the access control literature, inconsistency is also referred to by the presence of redundant rules [8]. The SPARQL query InconsistencyChecking($\mathcal{K}$) checks if at least for one of the classes orbac:Employ, orbac:Consider, orbac:Use and orbac:Define, two different instances are defined for the same subject, action and object, respectively. And whether instances of both orbac:Permission and orbac:Prohibition are defined for their consecutive orbac:Role, orbac:Activity, orbac:View and orbac:Context. The query also checks if every instance is linked with the same organisation, or with one of its parent organisations and whether the permission is linked with a parent role or the prohibition is linked with a senior role. The definition following establishes inconsistency in the sense of the orbac ontology:

[3] All paper appendices are available at `https://github.com/ahmedlaouar/orbac.owl`

Definition 3. *Let $\mathcal{K}$ be an* orbac *KB, $\mathcal{K}$ is inconsistent if both of the queries* Is-permitted(s, α, o) and Is-prohibited(s, α, o) return true for the same s, α, and o (both a concrete permission and prohibition are derived).

Moreover, a conflict is the minimal set of individuals deriving both a permission and a prohibition for the same subject, action and object.

Definition 4. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{R}, \mathcal{T}_U \cup \mathcal{T}_F \rangle$ be an* orbac *KB. Let I_U denote the uncertain individuals involved in $\mathcal{T}_U$. $\mathcal{C} \subseteq I_U$ is a conflict of $\mathcal{K}$ if:*
– InconsistencyChecking$(\langle \mathcal{T}, \mathcal{R}, \mathcal{T}_\mathcal{C} \cup \mathcal{T}_F \rangle)$ = True, and;
– $\forall i \in \mathcal{C}$, InconsistencyChecking$(\langle \mathcal{T}, \mathcal{R}, \mathcal{T}_{\mathcal{C}\setminus\{i\}} \cup \mathcal{T}_F \rangle)$ = False.
Where $\mathcal{T}_{\mathcal{C}\setminus\{i\}} = \mathcal{T}_\mathcal{C} \setminus \{\langle x, p, y \rangle \in \mathcal{T}_\mathcal{C} | x = i \text{ or } y = i\}$.

The individuals in $\mathcal{I}_U$ ($I_U = \{i \in I | \langle i, \text{rdf:type}, t \rangle \in \mathcal{T}_U$ and $t \in \{$orbac:Employ, orbac:Consider, orbac:Use, orbac:Define$\}\}$) are considered uncertain by the representation in Section 3.3. These individuals are those included in conflicts. The remaining individuals are fully certain or irrelevant to conflicts. Computing all the conflicts of a KB is achieved using a SPARQL query called ComputeConflicts($\mathcal{K}$), which is, to some extent, similar to the InconsistencyChecking($\mathcal{K}$) query.

Example 4. The query InconsistencyChecking($\mathcal{K}$) returns True from our example, because it infers both a permission and prohibition for (Bob, edit, report1). The conflict resulting from the query ComputeConflicts($\mathcal{K}$) contains the following individuals: $\mathcal{C} = \{$emp1, use1, cons1, emp2, def1, def2$\}$.

Proposition 1. *Checking the consistency and computing the conflicts of an* orbac *KB $\mathcal{K}$ using the queries* InconsistencyChecking($\mathcal{K}$) and ComputeConflicts($\mathcal{K}$) are both tractable. This follows from the efficiency of the used queries.

4.2 Repair-based method

An orbac knowledge base, as given by Definition 2 is partially preordered, thanks to the use of the property orbac:isPreferredTo. To resolve inconsistency in the orbac ontology, we propose to adapt the notion of accepted assertion used for DL-Lite$_\mathcal{R}$ partially preordered possibilistic knowledge bases in [1,17]. To achieve that, we start by introducing two notions: support and dominance. Schematically, a request for a permission is accepted if it admits at least one support which dominates all the supports for a prohibition [18].

Support: a support of a privilege for a given subject, action and object is the minimal set of individuals participating in its derivation. A privilege may have multiple supports. To compute the supports of a privilege, a SPARQL query is used. ComputeSupports$_p$($\mathcal{K}$,s, α, o), s.t. $p \in \{$Permission, Prohibition$\}$, and s, α and o are a subject, action and object, respectively, returns all the supports of a permission involving s, α and o. A similar query to compute supports of a prohibition is achieved by changing the requested access type with Prohibition, considering the fact that querying for a prohibition is slightly different, as it must check for senior roles prohibitions.

Definition 5. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{R}, \mathcal{T}_U \cup \mathcal{T}_F \rangle$ be an* orbac *KB and s, α and o be a subject, an action and an object respectively. Let I_U denote the uncertain individuals in $\mathcal{T}_U$. $\mathcal{S} \subseteq I_U$ is a support of a privilege involving s, α and o if:*
– $\text{ComputeSupports}_p(\langle \mathcal{T}, \mathcal{R}, \mathcal{T}_\mathcal{S} \cup \mathcal{T}_F \rangle, s, \alpha, o) \neq \emptyset$, and;
– $\forall i \in \mathcal{S}$, $\text{ComputeSupports}_p(\langle \mathcal{T}, \mathcal{R}, \mathcal{T}_{\mathcal{S}\setminus\{i\}} \cup \mathcal{T}_F \rangle, s, \alpha, o)) = \emptyset$.
*Where $p \in \{$*Permission, Prohibition*$\}$ and $\mathcal{T}_{\mathcal{S}\setminus\{i\}} = \mathcal{T}_\mathcal{S} \setminus \{\langle x, p, y \rangle \in \mathcal{T}_\mathcal{S} | x = i \text{ or } y = i\}$. and $\mathcal{T}_\mathcal{S}$ is the set of triples involving elements of $\mathcal{S}$.*

Example 5. A support of the permission of (Bob, edit, report1) contains the individuals $\mathcal{S}_{perm} = \{$emp1, def2, use1, cons1$\}$, and a support of the prohibition of (Bob, edit, report1) contains the individuals $\mathcal{S}_{proh} = \{$emp2, def1, use1, cons1$\}$.

Dominance: the object property orbac:isPreferredTo encodes a partial preorder over a knowledge base. The dominance relation as defined in [17] extends the partial preorder defined over the elements of a set into a partial preorder defined over its subsets. We adapt this definition to the relation defined in this paper.

Definition 6. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{R}, \mathcal{T}_U \cup \mathcal{T}_F \rangle$ be an* orbac *KB. Let I_U denote the uncertain individuals in $\mathcal{T}_U$. Let $\mathcal{B}_1$ and $\mathcal{B}_2$ be two subsets of I_U.*
$\mathcal{B}_1$ dominates $\mathcal{B}_2$ if $\forall x \in \mathcal{B}_1$, $\exists y \in \mathcal{B}_2$ s.t. ⟨x, orbac:isPreferredTo, y⟩ holds and ⟨y, orbac:isPreferredTo, x⟩ does not hold.

Accepted permission. In the case where, for a given query, both concrete permission and prohibition are derived, we adapt the notion of acceptance defined in [17] to decide whether a permission is granted or not. The idea is to verify for each support of a prohibition, if there exists a support of a permission dominating it. This way, multiple supports of a permission can participate in granting an access, while ensuring that there is no support of a prohibition that is more preferred to all the supports of a permission. Our approach is local and query-driven, and does not involve repairing the knowledge base. It is clearly in the spirit of argumentation methods. The method relies on the notions of support and dominance from previous section, and avoids inferring all privileges. Instead it checks if an access is granted using the following definition.

Definition 7. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{R}, \mathcal{T}_U \cup \mathcal{T}_F \rangle$ be an inconsistent* orbac *KB. An access is accepted for a given subject s, action α and object o, if for each $\mathcal{S}_{proh}$ a support of a prohibition for (s, α, o) (as per Definition 5), $\exists \mathcal{S}_{perm}$ a support of a permission for (s, α, o) s.t. $\mathcal{S}_{perm}$ dominates $\mathcal{S}_{proh}$ (as per Definition 6).*

Example 6. Let us now complete our example. The consortium agreed that the connection rule Consider is fully certain. For the privileges of Bob to edit the object report1, in Example 5, we derived a single support for a permission $\mathcal{S}_{perm}$ and a single support $\mathcal{S}_{proh}$. The new supports are: $\mathcal{S}_{perm} = \{$emp1, use1, def2$\}$, and $\mathcal{S}_{proh} = \{$emp2, use1, def1$\}$. From Example 2, ⟨emp1, orbac:isPreferredTo, emp2⟩, ⟨def2, orbac:isPreferredTo, def1⟩ and ⟨use1, orbac:isPreferredTo, emp2⟩. Therefore, according to Definition 6, $\mathcal{S}$ dominates $\mathcal{C}$, hence a permission is accepted for Bob to edit the object report1.

Since the computation of supports is done in polynomial time, and under the assumption that the size of the support sets of permissions and prohibitions (for an access request) is polynomial, the decision whether or not to grant a permission (Definition 7) is also done in polynomial time w.r.t. the size of the dataset (data complexity).

5 Text-based generation of explanations

Explainability in AI systems is becoming crucial nowadays because transparency is needed so that we can trust AI prediction. In Example 6, not only knowing that Bob can edit the object report1 is important, but also understanding the reasons for that decision made by the AI system. Different types of explanations can be provided, such as trace-based, counterfactual, contrastive and scientific explanations [6]. Our interest is trace-based and contrastive explanations. The former consists of finding the rationale that leads to the decision made by following the steps that were taken by the AI system [6], and the latter focuses on emphasising the features that make the differences between facts [6]. Our method leverages InferenceQuery($\mathcal{K}$) (presented in Section 3.3), ComputeConflicts($\mathcal{K}$) (Definition 4), ComputeSupports($\mathcal{K}$,s, α, o) (Definition 5) and the notion of dominance (Definition 6) to generate the explanations. It makes use of SimpleNLG [11] and nltk[4] libraries for natural language generation. This last section briefly presents our method for generating the explanations.

Algorithm 1 presents our method for generating the logic-based explanations (L) and the explanations in natural language (V) by receiving in inputs: OrBAC knowledge base $\mathcal{K}$, subject s, action α and object o. It starts with querying $\mathcal{K}$ for (s, α, o) to determine the different privileges with their justifications (stored in t), and it computes the conflicts (lines 1-2). If there is no conflict, the access is either permitted or prohibited where the contents of L and V explain it (lines 3-6). Otherwise, the process assumes that the access is denied at the beginning. In this case, the method continues with separating the 2 opposite traces: t_{perm} for permission and t_{proh} for prohibition, which form L and are verbalised in V (lines 9-12). Thereafter, it computes the supports of permission and prohibition, emphasises the contrasts regarding the conflict by means of $Cont = \mathcal{S}_{perm} \Delta \mathcal{S}_{proh}$ and shows this using logic-based explanations and natural language explanations (lines 13-17). The preferences related to (s, α, o) are computed to be part of the explanations (lines 18-19) to help understand the automatic resolution of the conflict. Lines 21-34 determine the access following Definition 6 and Definition 7. The access for (s, α, o) is granted if for each $s_{proh} \in \mathcal{S}_{proh}$, a support of prohibition (line 21), $\exists s_{perm} \in \mathcal{S}_{perm}$, a support of permission for (s, α, o), s.t. s_{perm} dominates s_{proh} (line 23). In line 24, Trace selects the preferences responsible for the dominance, which is used to constitute L_2 and V_2. Then, the access is permitted if $i = |\mathcal{S}_{proh}|$ holds, making L_2 and V_2 be part of the explanations.

[4] https://www.nltk.org/

Algorithm 1: Generating explanations for the access control

Require: $\mathcal{K}$, s, α, o {$\mathcal{K}$ = orbac $\mathcal{KB}$, s = subject, α = action and o = object}
  $t \leftarrow$ InferenceQuery$(\mathcal{K}, s, \alpha, o)$ {t traces the inferred privileges.}
  $C \leftarrow$ ComputeConflicts$(\mathcal{K}, s, \alpha, o)$
  if $C = \emptyset$ **then**
    {**NO access conflict**}
    $Access \leftarrow$ GetAccess(t) {$Access$ is either permitted or prohibited.}
    $L \leftarrow$ Formulate$(\mathcal{K}, t)$ {L: Logic-based explanations}
    $V \leftarrow V_{\text{simple}}(\mathcal{K}, t)$ {V: Verbalised explanations, V_{simple}: Trace verbalisation (t)}
  else
    {**Access conflict**}
    $Access \leftarrow$ false {Access is prohibited.}
      {***Trace-based explanations through the traces:*** t_{perm} ***and*** t_{perm}}
    $t_{perm} \leftarrow$ SelectPermission$(\mathcal{K}, t)$ {$t_{perm} \in t$ where t_{perm} concerns permission.}
    $t_{proh} \leftarrow$ SelectProhibition$(\mathcal{K}, t)$ {$t_{proh} \in t$ where t_{proh} concerns prohibition.}
    $L \leftarrow$ Formulate$(\mathcal{K}, t_{perm}) \cup$ Formulate$(\mathcal{K}, t_{proh})$ {L from t_{perm} and t_{proh}}
    $V \leftarrow$ append$(V_{\text{simple}}(\mathcal{K}, t_{perm}), V_{\text{Simple}}(K, t_{proh}))$ {Verbalisation}
      {***Trace-based and contrastive explanations from supports***}
    $\mathcal{S}_{perm} \leftarrow$ ComputeSupports$(\mathcal{K}, t_{perm})$ {Set of supports of permission}
    $\mathcal{S}_{proh} \leftarrow$ ComputeSupports$(\mathcal{K}, t_{proh})$ {Set of supports of prohibition}
    $Cont \leftarrow \mathcal{S}_{perm} \Delta \mathcal{S}_{proh}$ {$Cont$: contrasts between permission and prohibition}
    $L \leftarrow L \cup \mathcal{S}_{perm} \cup \mathcal{S}_{proh} \cup Cont$ {$Cont$ presents the contrastive explanation.}
    $V \leftarrow$ append$(V_{\text{support}}(Cont))$ {V_{support}: Verbalisation for supports}
      {***Presenting the preference relations***}
    $L \leftarrow L \cup$ pref$(\mathcal{K}, s, \alpha, o)$ {Presenting all preferences related to (s, α, o)}
    $V \leftarrow$ append$(V_{\text{pref}}($pref$(\mathcal{K}, s, \alpha, o)))$ {V_{pref}: Verbalisation for preferences}
      {***Trace-based explanations based on dominance***}
    $i \leftarrow 0$, $L_2 \leftarrow \emptyset, V_2 \leftarrow \emptyset$
    for all $s_{proh} \in \mathcal{S}_{proh}$ **do**
      for all $s_{perm} \in \mathcal{S}_{perm}$ **do**
        if dominates(s_{perm}, s_{proh}) {Following Definition 5} **then**
          $L_2 \leftarrow L_2 \cup$ Trace(dominates(s_{perm}, s_{proh})) {Presenting the preferences}
          $V_2 \leftarrow$ append(Trace(V_{pref}(dominates(s_{perm}, s_{proh})))) {Verbalisation}
          $i \leftarrow i + 1$
          break
        end if
      end for
    end for
    if $i = |\mathcal{S}_{proh}|$ **then**
      $Access \leftarrow$ true {By following Definition 6, the access is granted.}
      $L \leftarrow L \cup L_2$, $V \leftarrow$ append(V_2) {Providing the explanations}
    end if
  end if
Ensure: $Access$, L, V {$Access$ = Access control, L = logic-based explanations, V = explanations in natural language}

Access	**Conflict**: Bob is permitted and prohibited to edit report1. **Outcome**: Bob can edit report1.
Logic-based explanation	perm1(consortium, secondee, modify, reports, secondment) ∧ emp1(univ1, bob, secondee) ∧ use1(univ1, report1, reports) ∧ cons1(univ1, edit, modify) ∧ def2(univ1, bob, edit, report1, secondment) ∧ subOrganisationOf(univ1, consortium) ⊨ Is-permitted(Bob, edit, report1) prohib1(consortium, staffMember, modify, reports, default) ∧ emp2(consortium, bob, staffMember) ∧ use1(univ1, report1, reports) ∧ cons1(univ1, edit, modify) ∧ def1(consortium, bob, edit, report1, default) ∧ subOrganisationOf(univ1, consortium) ⊨ Is-prohibited(Bob, edit, report1) **Supports** $\mathcal{S}_{perm}$ = {emp1, def2, use1} $\mathcal{S}_{proh}$ = {emp2, def1, use1} $Cont = \mathcal{S}_{perm} \Delta \mathcal{S}_{proh}$ = {emp1, emp2, def1, def2} **Outcome** Is-permitted(Bob, edit, report1) because: ⟨emp1, orbac:isPreferredTo, emp2⟩, ⟨def2, orbac:isPreferredTo, def1⟩, ⟨use1, orbac:isPreferredTo, emp2⟩
Explanation in English	There is a conflict. Bob, a secondee at consortium, can edit report1. Bob is permitted to modify reports in a secondment context, where report1 is considered as reports, edit it is classified as a modify activity, and univ1 is part of consortium. Bob, a staffMember at consortium, cannot edit report1. Bob is prohibited to modify reports in a default context, where report1 is considered as reports, edit it is classified as a modify activity, and univ1 is part of consortium. There are contrasts: (a) Bob is a secondee at consortium, and Bob is a staffMember at univ1. (b) In univ1, the context secondment holds between Bob, report1 and edit, and in consortium, the default context holds between Bob, report1 and edit. Bob can edit report1 because: (1) 'Bob is a secondee at consortium' is preferred to 'Bob is a staff member at univ1'. (2) 'In univ1, the secondment context holds between Bob, report1 and edit' is preferred to 'In consortium, the default context holds between Bob, report1 and edit'. (3) 'report1 is used in reports at univ1' is preferred to 'Bob is a staffMember at univ1'.

Table 2: Example of a conflict of access. The explanations show why access is permitted and prohibited and why the final outcome is granted. The presentation of preferences is omitted for the sake of space.

Logic-based explanations. Formulate$(\mathcal{K}, t)$ translates the results t from querying $\mathcal{K}$ to logic-based representation. See the second line in Table 2 as an illustration. $\mathcal{S}_{perm} \cup \mathcal{S}_{proh} \cup Cont$, where $Cont = \mathcal{S}_{perm} \Delta \mathcal{S}_{proh}$ shows the contrast

between the supports. pref($\mathcal{K}, s, \alpha, o$) presents the list of preference relations that are involved in determining the privilege for (s, α, o).

Explanations in natural language. Concerning verbalisation, we used a template-based approach to generate explanations in natural language because the structure of an ontology is fixed. A template is a linguistic structure with gaps designed to be completed to form a sentence [10]. A linguistic template has been developed for each case, and the selected individuals from $\mathcal{K}$ fill in the gaps. See the third line, in Table 2 as an illustration. $V_{\text{simple}}(\mathcal{K}, t)$ uses the following template: "[subject], [role] at [Organisation], [ability] [action] the [object], and [Organisation] is part of [Organisation2]. This is because [subject] is [decision] to [activity] [view] in [context] context, where [object] is considered as [view], and [action] it is classified as a [activity] activity.", where "[x]" represent a gap to be replaced. "[ability]" is either 'can' or 'cannot' if the access is permission or prohibition, respectively. "[decision]" is either 'permitted' or 'prohibited' if the access is permission or prohibition, respectively. The other gaps are replaced by the individuals in t. V_{support} and V_{pref} follow the same principle. $V_{\text{support}}(\mathcal{V}) = V_{\text{ind}}(v_1) + \cdots + V_{\text{ind}}(v_n)$, where $v_1, \ldots, v_n \in \mathcal{V}$ are individuals, and V_{ind} is a template-based verbaliser like V_{simple}. We defined a template for each entity, and the algorithm selects the individuals to render the explanations. As an illustration, the template "[subject] is [role] at [Organisation]" is a template for the entity Employ. Then, using it, V_{ind}(emp1) generates 'Bob is a secondee at univ1'. In the case of contrast (line 17), the supports with the same classes are verbalised in parallel. Lastly, $V_{\text{pref}}(\mathcal{P}) = V_{\text{pref'}}(p_1(a_1, b_1)) + \cdots + V_{\text{pref'}}(p_m(a_m, b_m))$, where $p_1(a_1, b_1), \ldots, p_m(a_m, b_m) \in \mathcal{P}$ are the preferences responsible for the dominance, $a_1, \ldots, a_m, b_1, \ldots, b_m$ are individuals, and $V_{\text{pref'}}(p_i(a_i, b_i)) = V_{\text{ind}}(a_i) +$ 'is preferred to' $+ V_{\text{ind}}(b_i)$, with $1 \leq i \leq m$.

Discussions. We generated all privileges from our knowledge base, and the authors judge that the generated explanations are useful to help understand the outcome. We used a grammar checker: language_tool_python[5] for getting an insight into the grammaticality, and it is positive. We also used automatic metrics to know the readability of the explanations in natural language. As a result, the reader needs to have at least some college education, which is in line with the education of administrators. However, the best method to evaluate the quality of generated texts is through human evaluation [14], and the usefulness of the explanations has been checked by the authors. We can just say that the method has the potential to be useful and human-friendly. Details of this simple evaluation are available in the appendix [6].

6 Concluding discussions

In this paper, we propose an access control system relying on an ontology of the OrBAC model, on a SPARQL query engine to perform reasoning services and

[5] https://pypi.org/project/language-tool-python/

[6] All paper appendices are available at `https://github.com/ahmedlaouar/orbac.owl`

on a SWRL rules engine to automatically assign priorities in the ontology. We showed the efficiency of the different reasoning services and provided a new method to resolve conflicts, relying on inconsistency-tolerant semantics, which were initially defined for lightweight ontologies. Furthermore, trace-based and contrastive explanations are generated to support the decision made. Through a simple study, we found that the generated explanations are potentially useful and human-friendly. There is room for improvement. The importance of permission and prohibition are equal in our explanation generation, whereas prohibition might be essential. In addition, discriminating supports and conflicts may reduce the number of explanations while focusing on the relevant ones.

Acknowledgments. This research was supported by the European Union's Horizon research and innovation programme under the MSCA-SE (Marie Skłodowska-Curie Actions Staff Exchange); Call: HORIZON-MSCA-2021-SE-01; Project title: STARWARS (STormwAteR and WastewAteR networkS heterogeneous data AI-driven management) [grant agreement 101086252]. This research has also received support from the ANR project EXPIDA (EXplainable and parsimonious Preference models to get the most out of Inconsistent DAtabases), [grant number ANR-22-CE23-0017]. THR acknowledges support from the Hasso Plattner Institute for Digital Engineering through the HPI Research School at UCT. A. Laouar's PhD is supported by the ANR project Vivah (Vers une intelligence artificielle à visage humain) [grant number ANR-20-THIA-0004].

References

1. Belabbes, S., Benferhat, S.: Computing a possibility theory repair for partially preordered inconsistent ontologies. IEEE Transactions on Fuzzy Systems **30**(8), 3237–3246 (2021)
2. Benferhat, S., Bouriche, K., Ouzarf, M.: On the possibilistic handling of priorities in access control models. In: Foundations and Applications of Intelligent Systems: Proceedings of the Seventh International Conference on Intelligent Systems and Knowledge Engineering, Beijing, China, Dec 2012 (ISKE 2012). pp. 275–285. Springer (2014)
3. Benferhat, S., El Baida, R.: A prioritized-based approach to handling conflicts in access control. In: 16th IEEE International Conference on Tools with Artificial Intelligence. pp. 286–293. IEEE (2004)
4. Bienvenu, M., Bourgaux, C.: Inconsistency-tolerant querying of description logic knowledge bases. Reasoning Web: Logical Foundation of Knowledge Graph Construction and Query Answering: 12th International Summer School 2016, Aberdeen, UK, September 5-9, 2016, Tutorial Lectures 12 pp. 156–202 (2017)
5. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. Journal of Automated Reasoning **39**(3), 385–429 (2007)
6. Chari, S., Seneviratne, O., Gruen, D.M., Foreman, M.A., Das, A.K., McGuinness, D.L.: Explanation ontology: a model of explanations for user-centered ai. In: International Semantic Web Conference. pp. 228–243. Springer (2020)
7. Cuppens, F., Cuppens-Boulahia, N., Miège, A.: Inheritance hierarchies in the orbac model and application in a network environment. Proc. Foundations of Computer Security (FCS04) pp. 41–60 (2004)

8. Cuppens, F., Cuppens-Boulahia, N., Ghorbel, M.B.: High level conflict management strategies in advanced access control models. Electronic Notes in Theoretical Computer Science **186**, 3–26 (2007), proceedings of the First Workshop in Information and Computer Security (ICS 2006)
9. Dean, M., (eds.), G.S.: OWL Web Ontology Language Reference. World Wide Web Consortium, Recommendation REC-owl-ref-20040210 (February 2004)
10. Gatt, A., Krahmer, E.: Survey of the state of the art in natural language generation: Core tasks, applications and evaluation. Journal of Artificial Intelligence Research **61**, 65–170 (2018)
11. Gatt, A., Reiter, E.: Simplenlg: A realisation engine for practical applications. In: Proceedings of the 12th European workshop on natural language generation (ENLG 2009). pp. 90–93 (2009)
12. Harris, S., (eds.), A.S.: SPARQL 1.1 Query Language. World Wide Web Consortium, Recommendation REC-sparql11-query-20130321 (March 2013)
13. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A Semantic Web Rule Language Combining OWL and RuleML. World Wide Web Consortium, Member Submission SUBM-SWRL-20040521 (May 2004)
14. Howcroft, D.M., Belz, A., Clinciu, M., Gkatzia, D., Hasan, S.A., Mahamood, S., Mille, S., Van Miltenburg, E., Santhanam, S., Rieser, V.: Twenty years of confusion in human evaluation: Nlg needs evaluation sheets and standardised definitions. In: 13th International Conference on Natural Language Generation 2020. pp. 169–182. Association for Computational Linguistics (2020)
15. Jabal, A.A., Davari, M., Bertino, E., Makaya, C., Calo, S., Verma, D., Russo, A., Williams, C.: Methods and tools for policy analysis. ACM Computing Surveys (CSUR) **51**(6), 1–35 (2019)
16. Kalam, A.A.E., Baida, R.E., Balbiani, P., Benferhat, S., Cuppens, F., Deswarte, Y., Miege, A., Saurel, C., Trouessin, G.: Organization based access control. In: Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks. pp. 120–131. IEEE (2003)
17. Laouar, A., Belabbes, S., Benferhat, S.: Tractable closure-based possibilistic repair for partially ordered DL-Lite ontologies. In: European Conference on Logics in Artificial Intelligence. pp. 353–368. Springer (2023)
18. Laouar, A., Belabbes, S., Benferhat, S.: Conflict handling strategies for partially ordered access control security policies. In: 2024 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM). pp. 1–6 (2024)
19. Mohamed, A.K.Y.S., Auer, D., Hofer, D., Küng, J.: A systematic literature review for authorization and access control: definitions, strategies and models. International Journal of Web Information Systems **18**(2/3), 156–180 (2022)
20. Proctor, M.: Drools: a rule engine for complex event processing. In: Applications of Graph Transformations with Industrial Relevance: 4th International Symposium, AGTIVE 2011, Budapest, Hungary, October 4-7, 2011, Revised Selected and Invited Papers 4. pp. 2–2. Springer (2012)
21. Sandhu, R., Ferraiolo, D., Kuhn, R.: The nist model for role-based access control: Towards a unified standard. In: Proceedings of the Fifth ACM Workshop on Role-Based Access Control. p. 47–63. RBAC '00, Association for Computing Machinery, New York, NY, USA (2000)