A Contextual Access Control Model

Narhimene Boustia, Saad Dahlab University of Blida, Algeria
Email: narhimene@hotmail.com
Aicha Mokhtari, USTHB, Algeria
Email: aissani_mokhtari@yahoo.fr

Abstract—The proposed access control model is based on description logic (DL) augmented with a default ($\delta$) and an exception ($\epsilon$) operators to capture context feature. Our model is inspired from OrBAC and has an expressivity almost comparable to it, which means assigning authorization to a user according to its role in an organization in a given context. The most important difference of our model is the allow of composed context, the add of new context and the deduction of new authorization depending on context.

keywords: Access Control, OrBAC model, Context, Description Logic, Defaults and Exceptions, Reasoner.

I. INTRODUCTION

Access control consists in deciding whether the agent that issues a request should be trusted on this request.

Almost all studies on access control and authorization system have assumed the following model: “ a user makes an access request of a system in some context, and the system either authorizes the access requestor or denies it”.

Recently, different formalizations were developed for access control model. These include industry standards such as XACML (eXtensible Access Control Markup Language) [2], but also academic efforts ranging from more practical implemented languages such as Ponder [3], to theoretical languages such as in [4], semantic web based languages such as Rei [5] and KAoS [6], and finally to description logic for RBAC model [18], [19], [20], [21], [22]. In this framework, permissions are assigned to a user according to its role. This is insufficient: authorization can change with the context (exceptional contexts). The lack of exception features in these approaches is not inoffensive, it can leads to an incoherent system.

OrBAC (Organization Based Access Control) [1] is an access control model in which authorization is given to users depending on their role in an organization in a given context. The problem is that OrBAC is formalize in first order logic in which context is represented by an argument in a predicate Permission(organization, role, view, activity, context).

In this paper, we propose another solution to take into account the context. We use the $JClassic_{\delta\epsilon}$ system [23], developed by us for this purpose. It is a description logic based system augmented with two operators $\delta$ (for default) and $\epsilon$ (for exception). This kind of non monotonic reasoning in description logic is not sufficiently developed (see the recommendation of the Handbook of Description logic [9]). Actually, there is no system, in our sens, developed on this kind of reasoning in the web. The well known system as C-Classic, NeoClassic, Loom, Racer, Fact++ ... are all based on monotonic reasoning [10], [11], [12], [13], [14], [15], [16]. The use of our system in dynamic access control is interesting for many reasons: it allows for compounded context, adding of a new context and the deduction of new authorization depending on context.

Moreover, it is known that DLs are well adapted to modelize information system [9], [26]. This confort our choise to use DLs to provide access control with the representation of concepts such as user, role, object,...

The paper is structured as follow. We describe in the first one the key features of our tool $JClassic_{\delta\epsilon}$, which is the reasoner used to infer authorization. In the second part, we provide a new policy language and show how authorizations can be deduce with a case study. Finally, we conclude by prospects of evolutions of our access control model.

II. $JClassic_{\delta\epsilon}$

Description logics are a family of knowledge representation formalisms that stem from KL-ONE[17].

Several systems have been built based on DL[10], [11], [12], [13], [14], [15], [16], and used in real world applications. Besides, DLs facilitate the use of background knowledge and are more expressive than attribute value representations.

In DLs, a concept is defined as a set of properties satisfied by individuals that are instances of the concept. These properties are expressed by terms that are built from atomic concepts and roles and from a set of connectives. Concepts are partially ordered by a subsumption relation which expresses the inclusion relation between concepts and is usually based on a standard model based logical semantics.

C-Classic is a knowledge representation system based on description logic. C-Classic is equivalent to Classic [10], [11] augmented by the connective MIN and MAX which allow set of real to be defined. It is one of the most expressive previously known tractable DL, which preserves its good computational properties for subsumption. C-Classic is represented with an intentional semantic, which is more interesting than an extentional semantic (it is easier to compare two concepts by the use of their set
of properties than by their set of instances), and for our purpose, it is easier to add default and exception characteristics to concepts and roles properties than to the instances.

Our $JClassic_{δ}$ is a combination of $AL_{δ}$, a description logic augmented with default and exception [8] and $C$-Classic formalism [10], [11], it have the particularity to be possibly used in practical case. It not only allows representation of default knowledge and exceptional knowledge, but also infers and deduces new knowledge using tractable algorithms.

The subsumption relation in $JClassic_{δ}$, which is central to deducing access control, is presented in section 2.2. Knowledge is mainly separated into two components: a terminological component (TBox) which contains the terminological definition of concepts and roles and assertional components (ABox) containing statements about individuals.

A. Terminological Language

The connectives of $JClassic_{δ}$ are for the moment, the ones of $AL_{δ}$ [8]. The terminological language of $JClassic_{δ}$ is defined using a set $R$ of atomic roles, a set $P$ of atomic concepts, the constant $\top$ (Top) and the following syntactic rules (C and D are concepts, P an atomic concept, and R an atomic role):

\[
C, D \rightarrow \top \\
| \bot \\
| P \quad \text{primitive concept} \\
| C \cap D \quad \text{concept conjunction} \\
\text{Min}_u \quad u \text{ is a real number} \\
\text{Max}_u \quad u \text{ is a real number} \\
\text{ONE-OF} \{I_1, ..., I_n\} \quad \text{concept in extension} \\
R \text{ FILLS} \{I_1, ..., I_n\} \quad \text{subset of value for R} \\
R \text{ AT-LEAST} n \quad \text{cardinality for R (min)} \\
R \text{ AT-MOST} n \quad \text{cardinality for R (max)} \\
\neg P \quad \text{negation of primitive concept}\quad^2 \\
\forall r : C \quad C \text{ is a value restriction on all roles} \\
\delta C \quad \text{default concept} \\
\epsilon C \quad \text{exception to the concept}
\]

$\delta$ and $\epsilon$ are unary connectives, $\cap$ is a binary conjunction connective and $\forall$ enables universal quantification on role values.

B. Subsumption Algorithm $Sub_{δ}$

$Sub_{δ}$ is composed of 2 stages. The first is a normalisation of descriptions. The second is a syntactic comparison between normal forms.

Let C and D be two terms of $JClassic_{δ}$. To answer the question “Is C subsumed by D?” we apply the next procedure. The normal forms of C and “$C \cap D$” are calculated with the procedure of normalisation.

There are two steps in the comparison. We compare the strict parts of the two concepts. If these are equal, then we compare the default parts. If the two normal forms are equal, the algorithm returns “Yes”. Else it returns “No” otherwise.

In the next section we present our new access control model inspired by OrBAC model using $JClassic_{δ}$.

III. The Access Control Model

As in OrBAC model, the central entity is Organization. An Organization can be seen as an organized group of subjects, each playing a specific role. In our medical domain example, we consider an “hospital” as an Organization, “Jean” as a Subject, “Read” as an Action and “Medical record” as an Object which are respectively abstracted into Role, Activity and View [1].

A Role is a set of Subjects to which the same security rule applies. For example, the subject “John” plays the role of “Doctor” in the organization “Service of Pediatrics”. A View corresponds to a set of Objects that satisfy a common property. For example, in the medical domain, the view “Medical record” corresponds to the object “Medical record of patient”. An Activity groups together Actions that partake of the same principle. Actions will mainly contain computer actions such as “read”, “write”, etc, whereas Activities contain “consulting”, “writing”, etc. Privileges only apply in specific contexts. In OrBAC, Contexts are used to specify the concrete circumstances where organizations grant role permission to perform activities on views [7]. In our model, we take into account the context by considering the following postulate.

Postulate 1 (Normal context). By default, the context is normal.

Contrary to OrBAC model, our authorization model considers that:

Postulate 2. All actions that are not permitted are prohibited.

In this case, it suffice to defines permission relation.

The seven basic sets of entities of OrBAC are adapted here: OR (set of organizations), S (set of subjects), AC (set of actions), O (set of objects), R (set of roles), AV (set of activities) and V (set of views).

We now conceptualize access control model by a DL knowledge base capturing its characteristics, including the context with the use of defaults ($\delta$) and exceptions ($\epsilon$). We define then TBox and ABox axioms with examples to illustrate their content and use. We conclude in section 3.3 with a case study in which we demonstrate how an authorization can be deduced in each context variation by the use of our reasoner [24], [25].

A. The TBox

We define a DL knowledge base $K$. The alphabet of $K$ includes the following atomic concepts: Organization, Subject, Object, Role, View, Action and Activity. The TBox is as follow:
Definition of rules of security:

\[ \delta \text{Is} - \text{permitted} \subseteq \text{Employ} \sqcap \text{Use} \sqcap \text{Consider} \sqcap \delta \text{Permission} \]

- If a subject S is employed by organization Or in a role R (Employ), and if there is a relation between action Ac and activity Av (Consider), and if there is a relation between an object O and a view V (Use), and if we have by default a permission relation between role R, activity Av and a view V in an organization Or (δPermission), we deduce that a subject S is by default permitted to perform action Ac on object O (δIs – permitted), and because Is – permitted \( \subseteq \delta \text{Is} – \text{permitted} \) (a concrete permission can be deduced from a default permission), we can finally say that a subject S is permitted to perform an action Ac on object O.

\[ \text{Is} – \text{permitted}^c \subseteq \text{Employ} \sqcap \text{Use} \sqcap \text{Consider} \sqcap \text{Permission}^c \]

- By cons, if we have an exception on a permission concept wrote Permission\(^c\), we say that we have an exception on a concept Is-permitted wrote Is – permitted\(^c\), and because Is – permitted \( \not\subseteq \) Is – permitted\(^c\) (a concrete permission can not be deduced from an exceptional permission), we can deduce that a subject S is prohibited to perform an action Ac on object O.

We illustrate this in the next section, we show how a security policy can be modeled in our framework and how we can infer authorizations.

In the next section, we present an example of one ABox of our medical system information to show how authorization can be deduce.

C. Case Study

To illustrate how authorization can be deduce, we use the following instances of the ABox:

- Using the previous instances, the system infers that in organization X, each person who play the role of Doctor is by default permitted to modify Diagnosis and add this instance to the ABox: \( \delta \text{Permission}(P1) \).

Where:

\[ \delta \text{Permission}(P1) \sqsubseteq \text{PermissionAv.Activity}(\text{Modify}) \sqcap \text{PermissionR.Role}(\text{Doctor}) \sqcap \text{PermissionV.View}(\text{Diagnosis}) \sqcap \text{PermissionOr.Organization}(X) \]

Using the previous ABox, we show how deduction can be done in different contexts.

- Permission hierarchy: we show first how authorization can be deduced when we have an hierarchy of roles.

We know that a Surgeon is a sub-role of Doctor, so we can write:

\[ \text{Sub} – \text{role}(S1) \sqsubseteq \text{Sub} – \text{role}1.\text{Role}(\text{Surgeon}) \sqcap \text{Sub} – \text{role}2.\text{Role}(\text{Doctor}) \sqcap \text{Sub} – \text{roleOr.Organization}(X) \]

If we want to know if a Surgeon is permitted to write an ordinnance, we use the following rules:

\[ \delta \text{Permission}(PE1) \sqsubseteq \text{PermissionAv.Activity}(\text{Modify}) \sqcap \text{PermissionR.Role}(\text{Doctor}) \]

### Table I

<table>
<thead>
<tr>
<th>TBox</th>
</tr>
</thead>
<tbody>
<tr>
<td>Role attribution axiom:</td>
</tr>
<tr>
<td>Subject ( \subseteq \top );</td>
</tr>
<tr>
<td>Role ( \subseteq \top );</td>
</tr>
<tr>
<td>Organization ( \subseteq \top );</td>
</tr>
<tr>
<td>Employ ( \sqsubseteq \text{EmployS.Subject} \sqcap \text{EmployR.Role} \sqcap \text{EmployOr.Organization} );</td>
</tr>
</tbody>
</table>

| View definition axiom: |
| Object \( \subseteq \top \); |
| View \( \subseteq \top \); |
| Use \( \sqsubseteq \text{UseO.Object} \sqcap \text{UseV.View} \sqcap \text{UseOr.Organization} \); |

| Activity definition axiom: |
| Action \( \subseteq \top \); |
| Activity \( \subseteq \top \); |
| Consider \( \sqsubseteq \text{ConsiderAc.Action} \sqcap \text{ConsiderAv.Activity} \sqcap \text{ConsiderOr.Organization} \); |

To define a default permission, we use the following axiom:

\[ \delta \text{Permission} \sqsubseteq \text{PermissionAv.Activity} \sqcap \text{PermissionR.Role} \sqcap \text{PermissionV.View} \sqcap \text{PermissionOr.Organization} \];

To define a relation of hierarchy between roles, we introduce the following axiom:

\[ \text{Sub} – \text{role} \sqsubseteq \text{Sub} – \text{role}1.\text{Role} \sqcap \text{Sub} – \text{role}2.\text{Role} \sqcap \text{Sub} – \text{roleOr.Organization} \];

A concrete permission is expressed with the next axiom:

\[ \text{Is} – \text{permitted} \sqsubseteq \text{Is} – \text{permittedAc.Action} \sqcap \text{Is} – \text{permittedS.Subject} \sqcap \text{Is} – \text{permittedO.Object} \];

### ABox

- By cons, if we have an exception on a permission concept wrote Permission\(^c\), we say that we have an exception on a concept Is-permitted wrote Is – permitted\(^c\), and because Is – permitted \( \not\subseteq \) Is – permitted\(^c\) (a concrete permission can not be deduced from an exceptional permission), we can deduce that a subject S is prohibited to perform an action Ac on object O.

B. The ABox

The ABox of \( \mathcal{K} \) includes seven catalogs of axioms: Organization assertion’s axiom, Subject assertion’s axiom, Object assertion’s axiom, View assertion’s axiom, Role assertion’s axiom, Action assertion’s axiom and Activity assertion’s axiom.

It contains statement about individuals. We could have many ABox for one TBox depending on applications.
Access control in a default context: Suppose that user Jean wants to write diagnosis1, did he have this right?

We know that:
- Jean plays role of doctor in organization X: Employ(E1);
- and, Diagnosis1 is an object used in the view Diagnosis: Use(U1);
- and, Write is considered as a modification activity: Consider(C1);
- and finally, by default, in organization X, each person who plays the role of Doctor is permitted to modify Diagnosis, when Normal context is true: \( \delta Permission(P1) \).

Formally, we write:

\[
Employ(E1) \sqcap Use(U1) \sqcap Consider(C1) \sqcap \delta Permission(P1)
\]

Using security rules, we can deduce that the preceding proposition subsumes \( \delta Is \rightarrow permitted(I1) \).

Where:
- \( Is \rightarrow permitted(I1) \) subsumes \( Is \rightarrow permittedAc.Action(Write) \)
- \( Is \rightarrow permittedS.Subject(Jean) \) subsumes \( Is \rightarrow permittedO.Object(Diagnosis1) \)

And because \( Is \rightarrow permitted(I1) \sqsubseteq \delta Is \rightarrow permitted(I1) \), we can deduce that Jean is permitted to write diagnosis.

- Access control if context “Contamination-risk” is true: Suppose that there is a contamination risk and Jean wants to write diagnosis1, did he have this right?

In the context Contamination-risk, the system deduce a new instance P2 which is defined as follows:

\[
\delta Permission(P2) \sqsubseteq PermissionAc.Activity(Modify) \sqcap PermissionR.Role(Doctor) \sqcap PermissionV.View(Diagnosis) \sqcap PermissionOr.Organization(X)
\]

And we add to the ABox the next rule:

\( Permission(P1)^c \sqsubseteq \delta Permission(P2) \)

We know that:
- Jean plays role of doctor in organization X: Employ(E1);
- and, Diagnosis1 is an object used in the view Diagnosis: Use(U1);
- and, Write is considered as a modification activity: Consider(C1);
- and finally, by default, in organization X, each person who plays the role of Doctor is permitted to modify Diagnosis, when context Contamination-risk is true: \( \delta Permission(P2) \).

We obtain:

\[
Employ(E1) \sqcap Use(U1) \sqcap Consider(C1) \sqcap \delta Permission(P2) \equiv Employ(E1) \sqcap Use(U3) \sqcap Consider(C1) \sqcap \delta Permission(P1)^c
\]

We know that \( A^c \equiv \delta A^c \), we obtain:

\[
\equiv Employ(E1) \sqcap Use(U1) \sqcap Consider(C1) \sqcap Permission(P1)^c
\]

Using security rules, we can deduce that the precedent proposition subsumes \( Is \rightarrow permitted(I1)^c \).

And, because \( Is \rightarrow permitted(I1) \not\sqsubseteq Is \rightarrow \)
\(\text{permitted}(I_1)^\ast\), we can’t deduce Is-permitted(I1). Therefore Jean is not permitted to write diagnosis1 when there is a contamination risk. Our policy language allows us to have more than one exception in a context.

- Access control if context “Fatal-Desease” is true: a new context appears and Jean want to write diagnosis, can he get this right? We illustrate here an example of exception of exception case.

In the context Fatal-Desease, the system deduce a new instance P3 defined as follows:

\[
\delta\text{Permission}(P3) \sqsubseteq \text{PermissionAv}\cdot\text{Activity} (\text{Modify}) \sqcap \text{PermissionR}\cdot\text{Role} (\text{Doctor}) \sqcap \text{PermissionV}\cdot\text{View} (\text{Diagnosis}) \sqcap \text{PermissionOr}\cdot\text{Organization} (X)
\]

We know that permission in a Context Contamination-risk is an exception of permission in a context normal, and permission in a context Fatal-Desease is an exception of permission in a context Contamination-risk (we have an example of exception of an exception), then we augment the ABox by the rule:

\[
\text{Permission}(P2)^\ast \sqsubseteq \delta\text{Permission}(P3)
\]

We have:
- Jean plays role of doctor in organization X: Employ(E1);
- and, Diagnosis1 is an object used in the view Diagnosis: Use(U1);
- and, Write is considered as a modification activity: Consider(C1);
- and finally, by default, in organization X, each person who plays the role of Doctor is permitted to modify Diagnosis, when context Fatal-Desease is true: \(\delta\text{Permission}(P3)\).

We obtain:

\[
\text{Employ}(E1) \triangleleft \text{Use}(U1) \triangleleft \text{Consider}(C1) \triangleleft \delta\text{Permission}(P3)
\]

\[
\equiv \text{Employ}(E1) \triangleleft \text{Use}(U1) \triangleleft \text{Consider}(C1) \triangleleft \delta\text{Permission}(P2)^\ast
\]

\[
\equiv \text{Employ}(E1) \triangleleft \text{Use}(U1) \triangleleft \text{Consider}(C1) \triangleleft \delta(\text{Permission}(P1)^\ast)^\ast
\]

Using the rule \(\delta A \equiv \delta (A^\ast)^\ast\), we obtain

\[
\equiv \text{Employ}(E1) \triangleleft \text{Use}(U1) \triangleleft \text{Consider}(C1) \triangleleft \delta\text{Permission}(P1)
\]

Using security rules, we can deduce that the precedent proposition subsumes \(\delta I s \prec \text{permitted}(I1)\).

We know that \(\text{Is} \prec \text{permitted}(I1) \sqsubseteq \delta I s \prec \text{permitted}(I1)\), so we can deduce Is-permitted(I1).

Therefore Jean is permitted to write diagnosis when the disease is fatal.

We illustrate by this example, one of our main result, which is to consider the context dynamically, and infer the authorization to each context changing.

IV. CONCLUSION AND FUTURE WORKS

In this paper, we presented an approach for an access control authorization with concept and role by means of DL with default and exception. To summarize our result, we have:

- developed a model based on the main concepts of OrBAC: organization, subject, role, object, view, action, activity and context. However, instead of defining context as argument, we point out that it is more coherent to use the two operators of default and exception.
- provided an algorithm for the main computational tasks of classifying concepts under strict and default conditions without increasing the complexity of equivalent expressivity reasoner.

An interesting topic for future research is to extend our system to take into account other connectives for example the disjunction operator to make our system more expressive with keeping a reasonable complexity. We also want to consider time as contextual information.

REFERENCES