Bring Efficient Connotation Expressible Policies to Trust Management*

Yan Zhang[1,2], Zhengde Zhai[1,2], and Dengguo Feng[1,2]

[1] State Key Laboratory of Information Security,
Institute of Software Chinese Academy of Sciences
[2] National Engineering Research Center of Information Security

Abstract. Trust Management(TM) aims to provide effective access control in open systems. It enables the resource owners to reason and determine the access permissions on the basis of a collection of distributed authorization knowledge about the requester. However, to be efficient, most current TM approaches are based on DATALOG which can't directly express the connotation of TM authorization policies. Thus these policies are hard to be understood and maintained by human beings. In this paper, we propose a new approach called OT based on the ontology language OWL 2 EL. OT supports the connotation expressible policies and remains efficient since its procedure of compliance checking is provable to be tractable.

1 Introduction

The access control in the open distributed system remains to be a challenging problem, and Trust Management(TM)[5] is one of the possible solutions. TM approaches support the distributed authorization and enable the local administrators to make decisions both on local policies and external authorization statements, which are generally in format of signed certifications.

Multitudes of TM approaches with different design features have been studied, such as PolicyMaker[5], KeyNote[4], and RT[13][14] etc. Existing TM approaches are often composed of a language for describing policies/credentials that state what attribute, role, clearance etc.(referred to as authorization terms from now on) a principal has when he/she has another issued authorization terms, and a compliance checking procedure for deciding whether an access request complies with the submitted credentials and local policies(we use p/c to denote policy/credential from now on).

Every authorization item in a p/c consists of a subject term and several parameter constraints. For example, if an authorization item's subject term is "graduated Student", it may have constraints over three parameters: major, attending university, and GPA. Under human language, these parameter constraints can be free expressed and as complicated as needed. For example, the

* this paper is supported by grants from 863 High-tech Research and Development Program of China (2007AA1204040 and 2007AA1204050).

S. Qing, C.J. Mitchell, and G. Wang (Eds.): ICICS 2009, LNCS 5927, pp. 396–410, 2009.

administrators may define such credentials with "graduated Student" authorization item in the real world:

Example 1.1. The Central Academy of Sciences(CAS) says Alice is "a graduated Student, studying in information security, attending the Institute A, gotten a cumulative GPA of 3.8"

Example 1.2. The Ministry of Education (MOE) says "a CAS graduated student, who is studying in any one of the following fields: information security, software engineering, network,..., and attends any one of the following CAS Institutes: Institute A, Institute B ,...., and has a GPA higher than 3.6" can get an 'application for becoming an exchange student to VirtualCountry with 5,000 dollars scholarship per year"

Example 1.3. MOE says "a CAS graduated student, who is studying in some fields about computer science, and attends some CAS institute which has cooperated projects with VirtualCountry's Institutes, and has a GPA which is higher than average" can get an "application for becoming an exchange student to VirtualCountry with 5,000 dollars scholarship per year"

The first case shows a credential of Alice issued by CAS. In this credential, every parameter is constricted to a specific value. In the second credential, the value of every parameter is constricted to be in a predetermined static set. If we call this way in case 1.2 as a denotation constraint description way, then in the last credential, administrator uses a connotation description way to determine a set for every parameter which describes the common property of elements in the set.

Compared with the denotation description way, the connotation constraint description way of authorization items has several advantages in the real world because:

(1) It can reflect the real authorization intention of local administrators accurately. According to some potential security consideration, administrators often can decide which type of parameter values may premise safety and qualification. Writing the "type" instead of plenty of distinct values into the authorization items will make p/c more comprehensible, more concise, and easier to be examined.

(2) It helps the local administrators to create p/c more quickly. It is a time-consuming task to predetermine the parameters' extension values of authorization items before creating a p/c. What's worse, when external authorization items are to be included in the new p/c, local administrators have to ask for external entities' help to figure out the external parameter values pertaining to their thoughts. That premises plenty of time spent for many rounds of communication and negotiation. So, if administrators can write some words standing for the connotation of parameter constraints into p/c directly, the time of predetermining the extension values of those parameter will be greatly reduced.

(3) It makes the created p/c more stable. Under the extension description way, after the extension value sets are determined and the p/c is created, the p/c can't adapt to any change of the value sets. Actually the sets' changes may be

frequent in the real world, e.g. the average of GPA in a university will be different when new semester arrives, the number of Institutes which have co-projects with VirtualCountry may increase or decrease daily by daily. Therefore, if there is a method for administrator to express the connotation of parameter constraints, the p/c will maintain stable no matter how frequently the extension value sets are changed.

Motivation. In order to support the connotation description way in TM, a schema to define, publish, share and maintain some machine-readable lexical conceptual structure is needed(so appropriate concept words can be picked to describe the connotation of authorization items in p/c), while a mechanism to match local policies with external credentials not syntactically but semantically (the semantic relation between "computer science" in policy and "information security,network..." in credential can be recognized) is necessary too.

To our best knowledge, there are no existing TM approaches which can be both connotation expressible and decidable. The earliest TM approach—Policymaker is able to express any kind of p/c and check specific semantic relation between policy and credential in some custom-built program since everything in it is free programmable. However, this super capability leads to its undecidability in almost all cases. Most of the subsequent TM approaches, such as DL[11], SD3[9], Binder[7], and ${RT_1}^C$ [11], are based on tractable logic programming language DATALOG or its tractable variant $DATALOG^C$. Though tractable in compliance checking, these TM approaches neither include any lexical conceptual structure construction schema, nor support any semantic matching mechanism, so none of them can support the connotation description way. What's worse, the pure DATALOG based ones can't even support the denotation description way because they can't express parameters constraint under the functionless deficiency of DATALOG.

Contribution. In this paper we propose a novel TM approach called OT based on the ontology language OWL 2 EL, which has the following advantages:

(1) OT supports the connotation description way, and partially supports the denotation description way. Local administrators can freely assemble simple abstract concepts from public ontologies into compound concepts to describe parameter constraints, i.e., they can use sharable concept blocks to construct the connotation of parameter constraints like the way they take under human language. As for the capability to describe the denotation of parameter values in p/c, though restricted in some respects, OT shows some merits over RT_1^C—the existing best TM language supporting the denotation description way. (2)The compliance checking procedure of OT is provable to be tractable. In OT, every access request can be converted into a concept subsumption question under an $\mathcal{EL}++$ (EL's logic foundation) knowledge base. It is provable in this paper that, the compliance checking procedure of OT is tractable when the concrete data domains used in OT's supporting ontologies are all so called p_admissible domains.

The remainder of this paper is organized as follows. Section 2 presents related work of Trust Management. Section 3 provides the overview of OWL 2 EL and its logic basis. In section 4, we describe the entire OT approach. In section 5 we discuss the efficiency and expressivity of OT. Lastly, section 6 summarizes this paper and presents our future work.

2 Related Work

PolicyMaker is the first TM approach proposed in 1996 by Blaze etc[5]. In PolicyMaker, security policies and credentials are freely programmable. Programmable expression guarantees the expressivity, however the compliance checking procedure of PolicyMaker is polynomial-time solvable in a strictly limited case but undecidable in general[6]. Keynote [4], the second-generator of PolicyMaker, stipulates main parts of p/c to be written as mappings from condition attributes sets to compliance values. Such stipulation makes Keynote more easily to be integrated into application, but meanwhile it lowers the flexibility of expression. Keynote's undecidability has been proved in many cases[14].

Most of the subsequent TM approaches are based on DATALOG which contributes to the tractability of compliance checking. RT_0, Binder, and SD3 etc are typical of them. Binder[7] uses the horn clauses to define authorization statements, and arbitrary predicates to represent authorization contents. RT_0 is one sublanguage of RT—a family of Role-based Trust Management languages[13]. RT_0 uses roles to denote the sets of subjects who are granted some specific authorization items and gives the fundamental roles define rules to specify the role membership relation. Except Binder and RT_0, there are a lot of other DATALOG-based TM approaches, such as Delegation Logic[11], and SD3 etc. The expressiveness of these approaches is constrained as DATALOG is a quite restrictive logic programming language. In fact, they even can't describe parameter constraints of the authorization terms (such as "$GPA > 3.6$", "$institute \in \{A, B, C\}$" and so on).

Li et al made use of DATALOG's extension version—$DATALOG^C$ to design a ${RT_1}^C$ TM language[11]. By introducing three kinds of parameter constraints: $f = c$, $f \in S$, $f = ref$ (Here, every f is a parameter with specific data type) to modify role types, ${RT_1}^C$ shows better expression power than former DATALOG-based TM approaches, and still keeps tractable in compliance checking. Actually, ${RT_1}^C$ is quite adapt to express the extension constraints. For example, it can describe the first authorization item in example 1.2 as "student(researchfield $\in \{infosec, soft\ engineering, network, ...\}$, institute $\in \{A, B, ...\}$, $GPA > 3.6$)" easily. However, even ${RT_1}^C$ still can't express the connotation of parameter constraint such as the ones in the case 1.3.

Cassandra[15] is another TM approach based on $DATALOG^C$. It is able to automatically retrieve missing credentials over the network and supports automated trust negotiation. Furthermore, by introducing the predicates "canActivate", "canDeactivate", "canRedCred" etc., Cassandra can make multiple authorisation decisions apart from the one of performing an action, such as activating

and deactivating a role, and requesting a credential. Polakow and Skalka proposed a TM approach based on a monadic linear logic programming language so called LolliMon[16]. Their approach unifies authorization checking and distributed credential retrieval, which solves the distributed certificate chain discovery problem in full RT framework . Though several novel functionalities are added in these TM approaches, their expressiveness about parameter constraints in authorization items is not superior to $RT_1{}^C$'s, i.e., they are still unable to express connotation of parameter values.

3 OWL 2 EL

3.1 Syntax and Semantics

An ontology is a formal, explicit specification of a shared conceptualization, including a taxonomy describing the structural concepts of a knowledge area, and axioms capturing the essential rules in that area. OWL 2 Web Ontology Language provides several sub-languages to describe ontologies. One of the profiles of OWL 2 is EL[1] whose logic correspondent is description logic $\mathcal{EL}++$. EL can describe definitions, roles and axioms about different concepts to capture the specific knowledge shared in specific domain. Its description contents form a knowledge ontology about that domain. The syntax of EL can be converted into the one of $\mathcal{EL}++$ to run knowledge reasoning. Table 1 illustrates the abstract syntax of EL and the syntax and semantics of $\mathcal{EL}++$.

In $\mathcal{EL}++$ Atomic concepts (denoted by C,D), atomic roles(denoted by R), features(denoted by $f_1, ..., f_k$), and individuals(denoted by $a_1, ..., a_m$) can be used to construct complex concepts with constructors. The one through five rows of table 1 illustrate the constructed forms of concepts in $\mathcal{EL}++$. Among them, $p(f_1, ..., f_k)$ denotes a concrete domain constructor with every f_i in it is a feature which maps every object to an element in that concrete domain. Formally, a concrete domain D is a pair $(\triangle^D, \mathcal{P}^D)$. $\triangle^D$ is a data set such as integer set $\mathbb{Z}$,string set $\mathbb{S}$ and so on, while $\mathcal{P}^D$ is a set of predicate names. Each $p \in \mathcal{P}$ can be interpreted as a set of n-ary tuple $p^D \subseteq (\triangle^D)^n$. The other rows represent the valid concepts axioms which constitute an $\mathcal{EL}++$ knowledge base. The knowledge base is divided into a TBox and an ABox. The TBox comprises four kinds of constraint axioms, which respectively are general concept inclusion axioms(GCIs)— a set of concept inclusion axioms such as $C \sqsubseteq D$, role inclusion axioms (RIs)— axioms for role hierarchy, domain restrictions(DRs) and range restrictions(RRs)—respectively contain axioms for domain and range of roles. ABox is a finite set of individual assertions of two kinds: C(a) or r(a,b). C(a) is an assertion denoting that a is an individual which belongs to the concept C, while r(a,b) denotes there exists a binary relation r between a and b.

The semantics of $\mathcal{EL}++$ is an interpretation $I = (\triangle^I, .^I)$, which consists of a non null abstract individual set $\triangle^I$ and an interpretation functions. I maps

[1] EL's current version only supports *ObjectOneof* containing a single individual, though $\mathcal{EL}$ supports nominal with more than one individuals.

Table 1. OWL 2 EL syntax and semantics

Name	OWL 2 EL Syntax	$\mathcal{EL}++$ syntax	Semantics
top	owl:Thing	$\top$	$\triangle^I$
bottom	owl:Nothing	$\bot$	$\emptyset$
nominal	ObjectOneof($a_1, ..., a_m$)[1]	$(a_1, ..., a_n)$	$\{x \in \triangle^I \mid x \in \{{a_1}^I, ..., {a_n}^I\}\}$
conjunction	ObjectIntersectionOf(C,D)	$C \sqcap D$	$C^I \cap D^I$
Existential restriction	ObjectSomeValuesFrom(R,C)	$\exists$R.C	$\{x \in \triangle^I \mid \exists y \in \triangle^I, (x,y) \in R^I \wedge y \in C^I\}$
Concrete domain	DataSomeValuesFrom($f_1, ...f_k$,p)	$p(f_1, ..., f_k)$ for $p \in P^{D_j}$	$\{x \in \triangle^I \mid \exists y_1, ..., y_k \in \triangle^{D_j}, {f_i}^I(x) = y_i$ for $1 \leq i \leq k, (y_1, ..., y_k) \in P^{D_j}\}$
GCI	SubClassOf(C,D)	$C \sqsubseteq D$	$C^I \sqsubseteq D^I$
Equivalent	EquivalentClasses(C,D)	C=D	$C^I = D^I$
RI	SubObjectPropertyOf(SubObjectPropertyChain($R_1, ..., R_k$),R)	$R_1 \circ ... \circ R_k \sqsubseteq R$	${R_1}^I \circ ... \circ {R_k}^I \subseteq R^I$
Domain	ObjectPropertyDomain(R, C)	Dom(R)=C	$\{x \in \triangle^I \mid \exists y \in \triangle^I, (x,y) \in R^I\} \subseteq C^I$
Range	ObjectPropertyRange(R, C)	Ran(R)=C	$\{y \in \triangle^I \mid \exists y \in \triangle^I, (x,y) \in R^I\} \subseteq C^I$
Concept assertion	ClassAssertion(a, C)	$C(a)$	$a^I \in C^I$
Role assertion	ObjectPropertyAssertion(R,a,b)	R(a,b)	$(a^I, b^I) \in R^I$

each concept A to a subset of $\triangle^I$, each role r to a binary relation r^I in $\triangle^I$, each individual a to a a^I in $\triangle^I$ and each feature name f to a partial function f^I: $\triangle^I \rightarrow \triangle^D$.

3.2 Knowledge Base Reasoning

$\mathcal{EL}++$ enables a lot of questions to be answered under the knowledge base reasoning. Subsumption questions are the most typical questions among them. Since all of other questions can be reduced to the subsumption questions in $\mathcal{EL}++$, the computational complexity of knowledge base reasoning in $\mathcal{EL}++$ depends on the performance of reasoning about the subsumption problems. Under the semantics of $\mathcal{EL}++$, given two concept C and D, C is subsumed by D under knowledge base K iff $C^I \sqsubseteq D^I$ for every model I of K.

In 2008, Badder et al have proved that subsumption in any knowledge base of $\mathcal{EL}++(D_1, ..., D_n)$ can be decided in polynomial time w.r.t the knowledge base's size when two conditions are satisfied [3]. The first one is each of $D_1, ..., D_n$ is p_admissible concrete domain whose definition can be find in [8]. The second condition is that each axiom in RIs of the $\mathcal{EL}++(D_1, ..., D_n)$ knowledge base should meet the restriction stated below. Before we quote the restriction, we must make this point clear: Under a knowledge base $\mathcal{T}$, and role names r, s, we write $\mathcal{T} \models r \sqsubseteq s$ iff r=s or T contains role inclusions:

$$r_1 \sqsubseteq r_2, ..., r_{n-1} \sqsubseteq r_n, where\ r_1 = r, r_n = s$$

And $\mathcal{T} \models ran(r) \sqsubseteq C$, iff $\mathcal{T} \models r \sqsubseteq s$, and $ran(s) \sqsubseteq C \in \mathcal{T}$.

Definition 3.1. *Role Range Inclusion Restriction(RRIR): for every* $r_1 \circ r_2 ... \circ r_k \sqsubseteq r \in \mathcal{T}$, $n \geq 1$, *if* $\mathcal{T} \models ran(r) \sqsubseteq C$, *then* $\mathcal{T} \models ran(r_k) \sqsubseteq C$.

4 The OT Approach

4.1 Design of Supporting Ontologies

For every authorization entity, fundamental vocabularies and background knowledge of authorization can be formally conceptualized in three ontologies. The first one is a built-in ontology about the most general taxonomy which is shared by all entities. The second one named Auth-Info ontology defines the taxonomy of parameterized authorization items granted by the entity, while the last domain ontology represents the domain knowledge of the application system which the entity resides in.

Specific Auth-Info ontology and Domain ontology need to be developed by the specific application system administrators. In this paper, we assume every application system can publish its Auth-Info ontology and Domain ontology, or at least their fragments relevant to the public credentials signed by it. Those top-secret application systems which want to hide their authorization architectures and domain vocabularies strictly, such as government or military information systems are not considered in this paper's scope. In the following, we discuss the atomic classes and properties in the three EL ontologies at length.

Built-In Ontology

1. **Principal class:** representing all involved subjects such as access requesters, resource owners, administrators and so on. Every individual of this class is a universal unique ID in the form of URI reference, related to a public Key (for example, based on PKI certification or mapped to ID-Based public key). *Principal* contains two overlapping subclasses: ***Grantor*** and ***Grantee***, which represent the authorization items' grantors and grantees respectively.
2. **Authorization class:** representing various authorization items in different applications.
3. **hasAuth objectProperty:** this object property relates every *Grantee* individual to his/her/its granted *Authorization* individuals.
4. **grantedBy objectProperty:** this connects every *Authorization* to its *Grantor*.

Auth-Info Ontology

1. **subAuthorization classes:** Applications can define and share their own *Authorization* subclasses. Every *subAuthorization* class denote a type of authorizations items and their names can be used as the subject term of those authorization items. For example, some common *subAuthorization* classes are "*Role*", "*Permission*", and "*Group*". The *subAuthorization* classes may form class hierarchies. The application system for CAS referred in example 1.1-1.3 can form a hierarchy of *subAuthorization* classes like the one in Fig.1.
2. **objectProperties:** Every *subAuthorization* class may possess some specific objectProperties to describe its object parameters. Every ObjectProperty *hasOP* has some class in the Domain ontology as it value range. It can be used to express a constraint in the form of $\exists hasOP.E$, where E is a

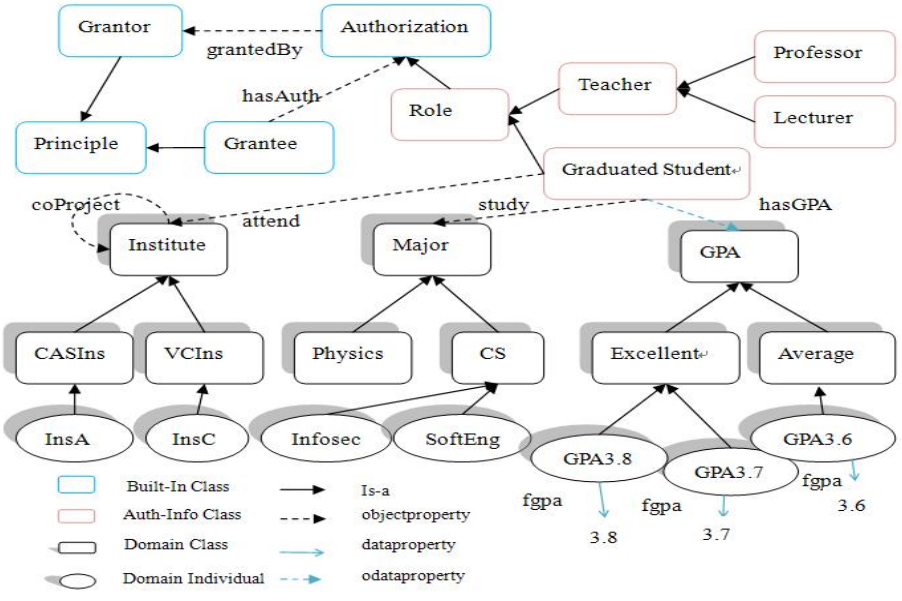

Fig. 1. An example of supporting ontologies in CAS's system

compound class freely constructed from a Domain ontology according to the administrators' needs.

3. **odataProperties:** In order to express constraints over data parameters, such as $age > 40$, $GPA > 3.6$, and so on, dataProperties are needed. However, in real world, even data value has diverse connotations in specific scenarios. For example, the GPA value 3.6 in GAS may stand for a minimum standard for student to be regarded as excellent. Hence all of the dataProperties of subAuthorization classes representing the data parameters are replaced by objectProperties in such way:
   For every data parameter dp, we define an objectProperty hasDP in Auth-Info ontology, and define a class DP and a dataProperty fdp for it in Domain ontology, where each value of dp corresponds to an individual of class DP whose feature fdp equals to that value, and hasDP's range is DP. Each constraint about dp can be expressed in the form of $\exists hasDP.(DP[\sqcap p(fdp)][\sqcap F])$, where p(fdp) is a data constraint about the parameter such as $< 3.6(fgpa)$, F is a compound class from Domain ontology to explain the connotation of the data parameter's value. In the expression, $\sqcap F$ and $\sqcap p(fdp)$ may be absent when administrator doesn't need them. In order to make such design behave well, we assume that every data parameter of authorization item has finite value range. In order to distinguish hasDP from other objectProperties in Auth-Info ontology, we call hasDPs as odataProperties.

Domain Ontology
Domain ontology models knowledge about specific application domain. Many applications have built or will build their own domain ontology no matter whether they use OT or not, such as the digital libraries which use the discipline ontology to classify their papers, and hospitals which use medical ontology to help their digital resource systems operate. The objectProperties of subAuthorization classes reside their ranges in Domain ontology, so administrators can use conceptualized knowledge from Domain ontology to describe the connotation of value constraints of objectProperties. Figure 1 also shows a simple ontology example from the application domain of CAS.

Applications can share the same Auth-Info and domain ontologies with others, or construct their own ones. All of these Auth-Info and domain ontologies should meet the RRIR restriction of definition 3.1 for all of their RI axioms. It can be easily achieved by making sure every R_k's range is subsumed by R's range when RI axioms are designed.

4.2 Basic Blocks of Credentials and Policies

Based on these ontologies, we can construct the OT credentials and policies and their basic building blocks with abundant vocabularies. There are many ways to format the policies/credentials, such as encapsulating every part of them into specific xml tag, expressing them as well-defined rules and so on. In order to simplify the following presentation, we developed an $\mathcal{EL}++$-EL mixed syntax to describe OT policies/credentials and their supporting blocks, which also makes the description easy to be converted into the Description Logic level in the process of compliance checking. At first, we use EBNF to define the syntax about the basic building blocks:

$$\textbf{CPR::} =' \exists hasAuth.(' \ CAI \ ' \sqcap \exists grantedBy.(' \ CPR|APR'))'$$
$$\textbf{CAI::} = subAuth\{'\sqcap' opconstraint\}\{'\sqcap' dpconstraint\}$$
$$opconstraint :: =' \exists' hasOP \ '.' \ E|('\{'s \ \{',' s\}'\}')$$
$$dpconstraint :: =' \exists' hasDP \ '.(' \ DP \ ['\sqcap' p'('f')'] \ ['\sqcap' F] \ ')'$$
$$\textbf{APR::} =' \{' anygrantor'\}'$$

Here, subAuth is a subAuthorization class in an Auth-Info ontology, while *hasOP* represents objectProperty and *hasDP* represents odataProperty of subAuth. *E*,*F* are compound classes whose components come from a Domain ontology. s is a Domain ontology's individual. *DP* and *f* are class and feature name of the domain ontology. anygrantor is an individual of Class *Grantor*. The former non-standard terms are all presented under the owl 2 EL syntax, while p is a predicate stated in $\mathcal{EL}++$ syntax.

A single **Compound Auth-Info expression(CAI)** can be regarded as an authorization item. It represents a group of *Authorization* individuals of the same subAuth class which match all *opconstraint* and *dpconstraint*. In the following, we demonstrate how the authorization items of example 1.3 can be directly interpreted by CAIs in the concept assembling way:

Example 4.1. The CAS authorization item in example 1.3:

$CAI_{exp1} = graduatedstudent \sqcap \exists study.CS \sqcap \exists attend.ObjectIntersectOf(CASIns, ObjectSomeValueFrom(coProject, VCIns)) \sqcap \exists hasGPA.(GPA \sqcap Excellent)$

Here, the relevant classes and properties are all from the ontologies in Fig.1.

Example 4.2. MOE authorization item in example 1.3

$CAI_{exp2} = Applicance \sqcap \exists Type.exStudentToVC \sqcap \exists hasSCH.(SCH \sqcap = 5000(fsch))$

Here the *Applicance* is a subAuhtorization class in Auth-Info ontology of MOE's application system. *Type* is its objectProperty. Together with *SCH* and *fsch* defined in Domain ontology, its odataProperty *hasSCH* is used for describe the data value constraint of "scholarship" parameter. *exStudentToVC* is a domain class representing a category of applications.

Building block **Atomic Principal expression(APR)** is a nominal class representing a single *Grantor* individual, while **Compound Principal expression(CPR)** describes a group of *Grantor* individuals who are granted the authorization item described by CAI and their grantor may be an APR individual or a member of another CPR class. One example of CPR is:

Example 4.3. the principals who are granted the network administrator Role by a super administrator

$\exists hasAuth.(NetAdmin \sqcap \exists grantedBy.\{superAdmin\})$

4.3 Complete Expression of Credentials and Policies

Credential: every credential is of the form:

$\langle Head : Reference \rangle \langle Body : Assertion \rangle \langle Signature \rangle$

The head part includes all the URI references of relevant signed ontologies and the public Key to verify them. The signature part contains the signature on this credential(the issuer's public Key certification can be included in this part). The authorization assertion in the body is one of the two forms:$AtoA^{\mathcal{C}}$ which provides an authorization to a group of grantees who have gotten other designated authorizations except identities, and $AtoI^{\mathcal{C}}$ which grants the authoritzation to a grantee with the designated ID.

$$AtoA^{\mathcal{C}} ::= CPR\{'\sqcap' CPR\}' \sqsubseteq \exists hasAuth.('CAI' \sqcap \exists grantedBy.\{'anygrantor'\})'$$
$$AtoI^{\mathcal{C}} ::= o' : \exists hasAuth.('CAI' \sqcap \exists grantedBy.\{'anygrantor'\})'$$

In $AtoA^{\mathcal{C}}$, the "anygrantor" should be the same principal as the issuer of the credential. Otherwise, the credential will be verified as invalid. In $AtoI^{\mathcal{C}}$, o is a grantee individual, and the "anygrantor" should be the same principal as the issuer of the credential.

Policies: Every policy is an authorization assertion stored in local system. In general, policies assert in what condition the local authorizations can be granted to the principals. They are not needed to be signed, connected to some local ontologies, and of the following two forms :

$AtoA^{\mathcal{P}} ::= CPR\ \{'\sqcap' CPR\}' \sqsubseteq \exists hasAuth.('CAI' \sqcap \exists grantedBy.\{'localAdmin'\})'$
$AtoI^{\mathcal{P}} ::= o' : \exists hasAuth.('CAI' \sqcap \exists grantedBy.\{'localAdmin'\})'$

Here *Grantor* individual localAdmin represents some local administrator.

Add standard term GROUP as macro: Every $\exists hasOP.E$ or $\exists hasDP.G$ ($G = DP[\sqcap p(f)][\sqcap F]$) in a CAI portrays a parameter constraint satisfied by the group of authorizations referred by the CAI. Since the quantifier used in the structures is $\exists$, the real meaning is "*there exists at least one individual pertaining to E/G that can be 'hasOPed'/ 'hasDPed' by every one of these Authorization individuals*". If creators want to express "*every individual pertaining to E/G can be 'hasOPed'/ 'hasDPed' by every one of these Authorization individuals*", they should look up every individual e_i in E/G from the Domain ontology, and write the feature of the CAI as:

$$EOP : \exists hasP.\{e_1\} \sqcap ... \exists hasP.\{e_m\}$$

Here, hasP represents hasOP or hasDP. To write the feature in this form is a heavy task. So we introduce a GROUP standard term to solve the problem. In this case, grantors only need to write the feature as the following SOP form in their credentials or policies.

$$SOP : \exists hasP.(E/G \sqcap GROUP)$$

When a request comes, credentials and policies are submitted to the decision module of the resource provider's OT subsystem. OT subsystem will recognize every GROUP standard term and automatically look up the corresponding Domain ontology to replace every SOP structure with the EOP structure.

4.4 Compliance Checking

In OT, every access request is made by one user, and decision is made under lots of relevant credentials and policies. Apparently, users are responsible for keeping, delegating or submitting their own $AtoI^{\mathcal{C}}$ credentials. However, who can be responsible for collecting the relevant $AtoA^{\mathcal{C}}$ credentials for every request? We suggest two reasonable and feasible strategies on this: (1) With a **free-style strategy**, resource providers don't control the delegation depth of authorization. So the middle level $AtoA^{\mathcal{C}}$ credentials are free issued out of their control and the requesters should be responsible for gathering them and submitting to the OT subsystem. (2) In the **strick-control strategy**, providers will designate explicitly all the $AtoA^{\mathcal{C}}$ credentials trusted by themselves, but permit the issuer to sign $AtoI^{\mathcal{C}}$ credentials freely. Henceforth all the $AtoA^{\mathcal{C}}$ credentials are stored in the OT subsystem of the resource provider. Then requesters only need to submit their own $AtoI^{\mathcal{C}}$ credentials along with the request.

Under any one of the two strategies, OT subsystem is supposed to get enough credential sets and can check the compliance efficiently. When an access request is submitted, the compliance checking procedure is processed in two phases.

Preparation phase: After receiving the request, server need to (1)collect relevant credentials according to free-style or strick-control strategy, (2) verify every

credential collected. If any credential is invalid drop it, (3) retrieve the relevant ontologies of every credential and verify their integrity. If any ontology fails in this step, drop the corresponding credential.

Execute phase: (1)Expand every GROUP macro appeared in collected credentials. (2)Merge and translate the relevant ontologies, authorization assertions in local policies and credentials into a synthesized $\mathcal{EL}++$ knowledge base TB, and translate the request into a subsumption problem "$\{req\} \sqsubseteq \exists hasAuth.(CAI \sqcap \exists grantedBy.\{localAdmin\})$?" where 'req' is the requester individual's ID, and CAI is the requested local authorization. (3)Run the reasoner and check if the subsumption holds under TB.

Next, we present a running example about the compliance checking. Supposed that the example 1.3 denotes a local policy of MOE, it will be expressed as such a OT policy:

$$\exists hasAuth.(CAI_{exp1} \sqcap \exists grantedBy.\{CAS\}) \\ \sqsubseteq \exists hasAuth.(CAI_{exp2} \sqcap \exists grantedBy.\{MOE\})$$

Meanwhile, Alice has a credential issued by Institute A of CAS like this:

$$Alice : \exists hasAuth.(graduatedStudent \sqcap \exists study.\{infoSec\} \sqcap \exists attend.\{InsA\} \sqcap \\ \exists hasGPA.\{GPA \sqcap = 3.8(fgpa)\} \sqcap \exists grantedBy.\{CAS\}).$$

If she wants to apply for the scholarship, she can make the request and submit her student credential MOE's OT subsystem. Then after relevant ontologies and credentials verified to be valid, MOE will generate a subsumption question: $\{Alice\} \sqsubseteq \exists hasAuth.(CAI_{exp2} \sqcap \exists grantedBy.\{MOE\})$. Under the relevant ontologies, MOE can infer that *InsA* is an individual of "$CASIns$" and is cooperating with the VCIns *InsC*, and *infoSec* is one research field in *CS*. No doubt that finally this subsumption question can be reasoned to be true. So Alice is a lucky girl to be qualified for this application.

5 Property Analysis

So far, we have presented the whole design of OT approach. As follows, we will present the efficiency result about OT's compliance checking procedure, and then give a detailed comparison between OT and former TM approaches in expressivity.

5.1 Compliance Checking Efficiency

As referenced at section 4.4, the compliance checking procedure is divided into two phases. Apparently, the preparation phase won't cost much time. Hence, the efficiency of compliance checking depends on the execute phase. Before introducing our conclusion about the efficiency of execute phase, we quote the p_admissible concrete domain's definition[8].

Definition 5.1. *A concrete domain D is p_admissible if:*

1. *the satisfiability of* $\bigwedge_i p_i(f_{i,1}, \ldots, f_{i,n_i})$ *and implication of* $\bigwedge_i p_i(f_{i,1}, \ldots, f_{i,n_i}) \rightarrow q(f_1, \ldots, f_n)$ *in D are decidable in polynomial time;*
2. *D is convex, i.e., if a conjunction of atoms of the form* $p(f_1, \ldots, f_k)$ *implies a disjunction of such atoms, then it also implies at least one of the disjuncts.*

Theorem 5.1. *If every concrete domain* D_i *used in the* $\mathcal{EL}++$ *knowledge base TB synthesized in execution phase is p_admissible, then the execution phase can be finished in polynomial time w.r.t the size of TB.*

Proof. Every RI axiom of every Domain ontology or Auth-info ontology meets RRIR when they are designed, so RRIR is kept for TB. Besides, every concrete domain D_i in TB is p_admissible. Thus, TB satisfies both the p_admissible condition and the RRIR condition. According to the discussion in section 3, the subsumption problem under TB is tractable w.r.t the size of TB. Because the Domain ontology of every relevant credential is part of TB's sources, the subsumption problem under the Domain ontology's corresponding $\mathcal{EL}++$ knowledge base is tractable w.r.t n if the size of that Domain ontology is n too.

Now we consider all steps in execution phase. In the first step, for every credential which has SOP expression to expand, the only time-consuming job is to find out all the individuals belonging to E_i of the SOP in its Domain ontology. This job can be done in polynomial time w.r.t the size of the Domain ontology n, because it needs no more than n subsumption decisions and every decision is solvable in polynomial time w.r.t n. Then the sum of all credentials' expansion time is polynomial w.r.t the size of the whole TB. The time spent in the second step can be neglected, while the third step is a polynomial time solvable subsumption decision in TB, so the whole execution phase can be finished in polynomial time w.r.t the size of TB. □

There exist a lot of p_admissible concrete domains, a typical one of which is $D = (Q, \mathcal{P})$, where Q is a domain of real numbers, and $\mathcal{P}$ consists of unary predicates $= q, > q$ for every q in Q. Though other non p_admissible may introduce intractability, they can still maintain the decidability if the satisfiability and implication in them are decidable. So the efficiency of OT can be kept at the decidability level at most cases.

5.2 Expressivity

In expressivity, with the unique concept assembling way to describe parameter constraints, OT can support not only the connotation description but also the denotation description.

(1) connotation description
As showed in the former sections, OT provides a desirable method for policy administrators to express the connotation of authorization. In OT, administrators can use concepts from external entity's Auth-Info ontology and Domain ontology

to express the connotation of authorization item they need. Since the ontologies are under the dynamic and lasting maintenance of external administrators, they can help the compliance checking procedure to automatically find out the current valid value sets for every property's connotation. But in the former efficient TM approaches, these can't be achieved.

(2) denotation description

To our best knowledge, ${RT_1}^C$ is the best solution when describing denotation of parameter constraints. Thus we just make a comparison between ${RT_1}^C$ and OT about their denotation description capabilities in the following:

Due to the absence of variable in OWL 2 EL, the third kind of parameter constraint "$f = ref$" of ${RT_1}^C$ is problematic in OT. However, OT can express ${RT_1}^C$'s "$f \in S$"("f=c" is just one kind of "$f \in S$") parameter constraints in its own way. Without loss of generality, we assume there is an authorization item expressed as $R(f \in S)$ in ${RT_1}^C$. In general, there are two translation versions about it in OT:

a. $R(f \in S)$ appears in the body part of any ${RT_1}^C$ rule(e.g, $A.R1(..) \leftarrow B.R(f \in S)$): since it's in the rule's body, f's hidden quantifier is $\exists$. If S belongs to a linearly ordered range set, we can use $R \sqcap \exists hasDP.(DP \sqcap \in S(f))$ to represent $R(f \in S)$. When the S belongs to some set of unordered enumeration elements, $R \sqcap \exists hasOP/hasDP.\{s_1, ..., s_n\}$ can be used where every s_i corresponds to an elements in S.

b. $R(f \in S)$ appears in the head part of any ${RT_1}^C$ rule(e.g, $A.R(f \in S) \leftarrow B.R_1$): Since it's in the rule's head, f's hidden quantifier is $\forall$. So we can use $R \sqcap \exists hasDP.(DP \sqcap \in S(f) \sqcap \text{GROUP})$, or $R \sqcap \exists hasOP/hasDP.(\{s_1, ..., s_n\} \sqcap \text{GROUP})$ to represent $R(f \in S)$.

In detail, these translation ways show both disadvantages and advantages:

(1) disadvantages: The introduction of predicate "$\in S$" will cause intractability since domain D=(C,{$\in S$}) where C is any linearly ordered set is non p_admissible (Despite this fact, the decidability can still be maintained). On the other hand, a few datatypes such as long, double are not supported in the current EL version, which causes those S sets of these restricted datatypes can't be expressed at present.

(2) advantages: As discussed before, the hidden quantifier for f in $R(f \in S)$ is decided by the position of $R(f \in S)$ in a rule. That is to say, $R(f \in S)$ can become neither $\exists f R(f \in S)$ in head nor $\forall f R(f \in S)$ in body. This inability hinders it to express some human thoughts about authorization. But in OT, GROUP macro can show up in head or body of rules, so this problem doesn't exist for OT.

6 Conclusions

In this paper, we proposed an OWL 2 EL based TM approach OT, illustrated OT's capability in expressing connotation of TM authorization policies and proved OT's tractability in compliance checking. In the future, we will study how to extend OT's expressivity further, such as introducing variables into OT's policy language.

References

1. OWL EL Introduction. See http://www.w3.org/TR/owl2-profiles/♯OWL_2_EL.
2. Baader,F.,Brandt,S., Lutz,C.: Pushing the $\mathcal{EL}$ envelope. In: Proceedings of IJCAI (2005) 364–369.
3. Baader,F., Brandt,S., C. Lutz.: Pushing the $\mathcal{EL}$ envelope further. In: Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions. (2008).
4. Blaze, M., Feigenbaum,J., Keromytis,A.D.: Keynote: Trust management for public-key infrastructures (position paper). In: Proceedings of the 6th International Workshop on Security Protocols. (1999) 59–63.
5. Blaze, M., Feigenbaum, J., Lacy, J.: Decentralized Trust Management. In: Proceedings of the 17th Symposium on Security and Privacy. (1996) 164C-173
6. Blaze, M., Feigenbaum, J., Strauss,M.: Compliance checking in the policymaker trust management system. In: Proceedings of the Second International Conference on Financial Cryptography. (1998) 254–274,.
7. DeTreville,J.: Binder, a logic-based security language. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy. (2002)page 105–113.
8. Baader,F., Lutz, C.: Pushing the $\mathcal{EL}$ envelope. Technical report, LTCS-Report ltcs-05-01, Inst.for Theoretical Computer Science, TU Dresden, See http://lat.inf.tudresden.de/research/reports.html (2005).
9. Jim,T.: SD3: A trust management system with certified evaluation. In: Proceedings of the 2001 IEEE Symposium on Security and Privacy. (2001) 106–115.
10. Li,N.: Local names in SPKI/SDSI. In: Proceedings of the 13th IEEE workshop on Computer Security Foundations. (2000) 2–15.
11. Li,N., Grosof, B.N., and Feigenbaum,J.: Delegation logic: A logic-based approach to distributed authorization. ACM Trans. Inf. Syst. Secur., 6(1):128–171 (2003).
12. Li,N., Mitchell,J.C.: Datalog with constraints: A foundation for trust management languages. In: Proceedings of the Fifth International Symposium on Practical Aspects of Declarative Languages. (2003).
13. Li,N., Mitchell,J.C., Winsborough,W.H.: Design of a role-based trust-management framework. In: Proceedings of the 2002 IEEE Symposium on Security and Privacy. (2002) 114-130.
14. Li,N., Mitchell,J.C., Winsborough,W.H.: Beyond proof-of-compliance: security analysis in trust management. J. ACM, 52(3):474–514, 2005.
15. Becker,M.Y., Sewell,P.: Cassandra: flexible trust management and its application to electronic health records.In IEEE Computer Security Foundations Workshop.(2004)139-154
16. Polakow,J., Skalka,C.:Specifying Distributed Trust Mnagement in LolliMon. In Proceedings of the 2006 workshop on Programming languages and analysis for security (2006)37–46.