Attribute Delegation Based on Ontologies and Context Information

Isaac Agudo, Javier Lopez, and Jose A. Montenegro

Computer Science Department, E.T.S. Ingenieria Informatica
University of Malaga, Spain
{isaac, jlm, monte@lcc.uma.es}

Abstract. This paper presents a model for delegation based on partial orders, proposing the subclass relation in OWL as a way to represent the partial orders. Delegation and authorization decisions are made based on the context. In order to interact with the context, we define the Type of a credential as a way to introduce extra information regarding context constraints. When reasoning about delegation and authorization relationships, our model benefits from partial orders, defining them over entities, attributes and the credential type. Using these partial orders, the number of credentials required is reduced. It also classifies the possible criteria for making authorization decisions based on the context, in relation to the necessary information.

1 Introduction

This work presents a delegation model that defines general credentials. A credential is defined as a link between two entities, in relation with some attribute. Credentials also have a type, which defines their special characteristics and in particular, information regarding the context. Two of these characteristics are whether it is a delegation or an authorization credential and the validity time interval.

We use the word Delegation to describe the transfer of management rights over certain attributes. The sentence "A delegates attribute r to B" is used as a shortcut for "A authorizes B to issue credentials regarding the attribute r to any other entity C". We use delegation as a Meta concept in an authorization framework, because delegation statements are authorization statements over the act of authorizing. This meta information is used to facilitate the distribution of authorization, providing a powerful instrument to build distributed authorization frameworks.

In real organizations, there are several variables that need to be considered when taking authorization decisions. It is important to define the different kinds of entities in the system and the relationships between them. We need and Organization Chart to be able to classify entities. This chart establishes a hierarchy of entities and all the decisions made within the organization have to comply with this hierarchy. There are cases in which this organization chart only has a few classes of entities and others in which there are a considerable number of

them, but in any case the chart is quite static, so it is not worth using certificates or credentials to define it. One possible solution is to define an ontology of entities, represented as classes, using the subclass relation. This can be de done very easily using OWL [6] and any of the tools that support it. There is both an OWL plug in for Protege [5] and a stand alone application called SWOOP [2] that allows us to use OWL graphically.

Classes that are higher up in the hierarchy refer to more general classes and, on the other hand, those classes that are lower down in the hierarchy refer to more specific classes. So, if class A is a specialization of class B, then all privileges linked with B have to be inherited by A. Membership of entities is modeled using unitary classes, so we only talk about classes and subclass relations. This simple ontology simplifies the process of issuing credentials as we can issue privileges to several entities using only one credential and the OWL ontology. This ontology can be mapped to a partial order set in which the elements correspond to the OWL classes (unitary or not) and the subclass relation defines the partial order. We take the same approach with privileges. Attributes are used as a bridge to cover the gap between entities and privileges. We define a partial order over attributes as a hierarchy in which we go from general attributes (those more related to the concept of Role) to specific attributes (those more related to privileges and resources). With these ontologies, we simplify the delegation and authorization chart of the Organization, as we split it into three sections:

- Organizational relation between entities
- Organizational relation between attributes
- Specific delegation and authorization credentials.

The point here is to combine this information to make correct delegation and authorization decisions. Another interesting point of our proposal is the concept of Type, which is closely related to the context. In the Type of a credential we encode extra information that might affect delegation and authorization decisions depending on the context in which the decisions have to be made. The validity interval is an example of information that has to be included in the specification of the Type of a credential. We follow the same principle of using ontologies, in the definition of Types. In the simple example of time intervals, it is clear that we may establish a subclass relation over them, using the subset relationship. In general, to be coherent, all the information included in the Type is interpreted as a concept in a subclass only ontology. This ontology is used to automatically derive virtual credentials from real ones. Going back to the time intervals, if we chain two delegation credentials C_1 and C_2, which are valid in intervals t_1 and t_2 respectively, the validity interval of the path will be $t_1 \cap t_2$ as it is the maximal element that is both lower than t_1 and t_2. What we do here is to derive new credentials C_1 and C_2, both valid in $t_1 \cap t_2$, using the ontology (or partial order) defined for the type of credentials and in particular for validity intervals.

Therefore, in this paper we present a general framework to model both delegation and authorization relationships. This model is based on the definition of

typed credentials, which is one containing extra information called the Type of the credential. This information is used to link the credential with its context.

We propose the definition of subclass relations over all the concepts used in the framework: Entities, Attributes and Types. These relations are encoded using ontologies, and in particular the OWL `subclassOf` relation. These subclass relations can be interpreted as a partial order relation[1] and in this paper we use only the mathematical notation, because it is easier to understand and work with. These ontologies provide helpful information for defining access control polices, simplifying the process of defining access credentials, control policies and making authorization and delegation decisions more efficient.

The definition of a credential type is particularly useful as it allows us to restrict delegation according to context parameters, such as location and time, and in general according to the state of the system. When working with delegation credentials, it very important to define the requirements that must hold for a credential to be valid, because revoking delegation credentials is very expensive task in terms of computation.

The structure of the paper is as follows. In section 2, we present the basic elements or building blocks of the framework: Entities, Attributes and Types, and explain the meaning of the ontologies (partial orders) defined for each of them. In section 3, credentials are defined as a construct that describe the basic elements of the framework. Credentials are defined as tuples of the form: $(Issuer, Holder, Attribute, Type)$. In this section we introduce the validity evaluation as a function that distinguishes between valid (active) and invalid (inactive) credentials in a given state and classify them according to the information used to make this decision. Section 4 defines paths or chains of delegation credentials, and explains how validity evaluation can be extended to define an evaluation function over credential paths. In section 5 some conclusions are presented.

2 Elements of the Framework

We mentioned before that a Delegation is a Meta-Authorization. So, before the definition of delegation credentials, we have to review authorization credentials.

The standard X.509 defines an authorization credential having the following fields[2]:

- *Issuer*: The entity who authorizes the Holder.
- *Holder*: The entity who receives the attributes.
- *Attributes*: The attributes issued to the Holder.
- *Validity Period*: The validity period of the authorization.

For a credential to be valid, the Issuer should have management rights over the attributes issued in the credential. In this case, the Holder is able to use the new attributes but it is not able to issue new credentials regarding these

[1] $A \leq B$ if and only if A is a subclass of B

[2] we focus on the most important ones

attributes. In the case where it is a Delegation credential, the Holder should be able to issue new (Authorization) credentials regarding these attributes, instead of being able to use the attributes directly. This is the main difference between an Authorization Credential and a Delegation Credential. Based on this definition, we define credentials as 4-tuples, in which we have an *Issuer* field, a *Holder* field, an *Attribute* field and a *Type* field.

2.1 Entities

Delegation is established between two entities. Thus, we have to define what an entity is in our framework. We distinguish between individual entities, those that are indivisible, and general entities (or just entities) that may be composed of more than one individual entity or refer to more complex concepts. We use this distinction inspired by the concept of Role [3]. In this sense, Roles are related to general entities. We define both general and individual entities as OWL classes. For example, in our framework Bob will be represented as a class, and the members of this class will be the instances of Bob, each time he is authenticated in the system.

In any working environment, there are some fixed rules that define entity relationships. If we think of a University, there is an inclusion relation between professors and employees, as professor is a specification of the concept (class) employee. So the classes *Professor* and *Employee* are related by the `subClassOf` relation. All privileges granted to employees should also be granted to professor. A shortcut to avoid this double issuing of credentials could be to define a partial order on entities. In this case, employee is a general entity but professor is an individual entity as there are no other entities more specific than professor. If we include real people in this simple example, the following chain of implications is obtained:

$$Alice \Rightarrow Profesor \Rightarrow Employee$$

Therefore neither Employee, nor Professor are individual entities, and only Alice has this category of individual entity.

Instead of using the symbol $\Rightarrow$ we use the symbol $\leq$ to emphasize that we have a partial order defined in the set of entities, we also use $<$ as a shortcut for $\leq$ and $\neq$. In this way, the previous chain of implications can be translated to:

$$Alice \leq Profesor \leq Employee$$

Definition 1 (Entities). *The set of entities is defined as a partial order* $(\mathcal{E}, \leq_e)$ *and the subset of individual entities by* $\mathcal{E}^* := \{e \in \mathcal{E} : \nexists e' \in \mathcal{E}, e' < e\}$

Now, let us do it the other way round, and think about the following scenario. Suppose Bob needs a credential issued by an employee of the University to be able to enter the Library. If Alice issues this credential, the verifier of the credential, i.e. the library, should be able to compare *Alice* and *Employee* to determine if this credential is valid for this purpose. In this case, as $Alice \leq Employee$

a credential issued by *Alice* could also be interpreted as a credential issued by *Employee*.

So, if we have two entities E_1 and E_2 where $E_1 \leq E_2$ we obtain the following inference rules:

1. Any attribute issued to E_2 should also be issued to or used by E_1.
2. Any credential issued by E_1 could also be interpreted as a credential 'issued' by E_2.

Although only individual entities (minimal elements of $\mathcal{E}$) are allowed to issue credentials, we can give meaning to a credential 'issued' by non-individual entities. So a credential 'issued' by non-individual entities could be a requirement constraint for the authorization of some operations. This constraint means that an individual entity, which is lower in the hierarchy than the required non-individual entity, has to issue a credential in order to authorize the requested operation.

Depending on how many entities are ordered we get two extreme cases: a non ordered set and a complete lattice.

- Trivial poset. In this case, the set of entities in not really ordered. Thus, any entity is minimal and therefore all entities are individual entities, i.e $\mathcal{E}^* = \mathcal{E}$. This is the simplest approach and is the one used for example in SPKI [4].
- Power Set with the inclusion order. In this case $\mathcal{E} \simeq \mathcal{P}(\mathcal{E}^*)$. This structure allows us to use groups of entities as holders of credentials. The partial order is induced by the inclusion operator. In fact, the defined structure is a complete lattice, using the Union and Intersection operators as the join and meet operators. This case is not realistic due the huge number of classes that have to be considered.

Depending on the application context, the number of classes will vary. We have to reach a balance between the two extreme cases previously presented.

2.2 Attributes

In many authorization management systems, credentials grant access directly to resources, so there are no attributes. An attribute is a more general concept than resource. Resources can be easily identified in the system, but they do not provide us with a fine grain way of defining authorization. A Unix file, f, is clearly a resource, so an entity A could issue an authorization credential to B, regarding the resource f. In this case, shall B be able to modify it? In general it depends on the file. So, in many cases, resources are associated with different access modes. In order to describe an operation that could be authorized or not, we need a tuple $(resource, right)$. But resources are normally composed of lower level resources, e.g. a Computer is made up of memory, hard disk (which is also composed of files) and many other components. There is an implicit order relationship over the resources of a given system, as there is a partial order over entities. Because of the complexity of defining what is a resource and what not,

it is better to use *privileges* or more accurately, *attributes* to define the nature of the authorization.

The use of attributes is an initiative to generalize the use of a tuple of the form $(resource, operation)$. Using attributes we give the authorization policy more relevance, as it is in the policy where we have to match attributes with the traditional tuple $(resource, operation)$. So attributes allow us to divide the authorization into two phases:

- Defining Attributes. It consists of giving a meaning to the attributes used in the system and therefore, it assigns a tuple $(resource, right)$ or a set of privileges to each attribute.
- Issuing Attributes. It consists of issuing credentials in relation with the previously defined attributes.

We identify two kinds of attributes widely used in the literature:

- Membership Attribute. It encodes the concept of *x is a member of role A*. Here the attribute is totally equivalent to the concept of Role.
- Specific Attribute. It encodes the concept of *x has access privilege A*. Here the attribute is totally equivalent to the privilege A which directly translates to a tuple of the form $(resource, right)$.

These two examples show that by using attributes we could model any of the existing traditional approaches. In our framework we can encode roles and role membership using entities and the partial order over entities, so we think of attributes as an intermediate concept between roles and privileges.

There are many situations in which there is a hierarchy over attributes. For example, write access to a file may imply read access. Another clear example is the case of attributes with parameters, if $AGE(x)$ represents that the owner entities are at least x years old, then $AGE(x)$ also implies $AGE(x-1)$ and in general $AGE(y)$ where $y \leq x$.

For this reason, we translate this hierarchy into a partial order over attributes using an OWL ontology similar to the one defined over entities. This ontology helps us in the decision making process.

If we think of a drug store, where alcohol is sold, prior to buying an alcoholic drink, entities should present a credential stating that they are at least 21 years old, i.e. $AGE(21)$. Now, suppose that an older person, with the attribute $AGE(60)$ tries to buy whisky. In our case, as his attribute is 'greater' than the required one,i.e. $AGE(21) \leq AGE(60)$, he should be allowed to buy whisky. Then, the requirement is the attribute $AGE(21)$ instead of the classical approach in which we require an attribute $AGE(x)$ where $x \geq 21$.

The privilege $BuyAlcohol$ is defined as a specific attribute in the set of attributes $\{AGE(x) : x \in \mathbb{N}\} \bigcup \{BuyAlcohol\}$. The partial order is defined as follows: $BuyAlcohol \leq AGE(21)$ and $AGE(x) \leq AGE(y)$ if and only if $x \leq y$. This ontology helps us to understand the authorization policy in which only the attribute $BuyAlcohol$ is required to buy alcohol.

2.3 Type

There are other parameters that have to be included in the specification of the credentials, besides the attribute, the issuer and the holder. We define the type of credential to include all the properties that are not essential, but which are helpful, to define Authorization and Delegation. This separation of concepts was previously proposed in [1], but we extend it here to include more general information, in particular, information regarding the context.

A credential type is mainly used to determine if it is valid or not under certain circumstances. Time is the basic type and it will be used as a model to define new types that help us to restrict delegation and authorization according to context information.

Consider the situation in which the only relevant characteristic to determine whether a credential is valid at a particular point in time is the validity interval. In this case, the type of the credential consists of a time interval. The set of time intervals has a natural partial order induced by the inclusion relation, i.e. one interval I_1 is lower than another interval I_2 if $I_1 \subset I_2$. Formally, $\mathcal{T}_I := \{[n, m] : n, m \in Time\_Instants \bigcup \{\infty\}, n \leq m\}$.

As with Entities and Attributes, this partial order can be used to derive new information from the current credentials of the system. Suppose we have a credential with type $[0, 5]$, then we could derive new credentials with a lower type, e.g. $[1, 4]$.

Another important type is the one that defines whether a credential is delegable or not. We define the type 0 for non delegable credentials and the type 1 for delegable credentials. Therefore, the delegation type is defined by $\mathcal{T}_D := \{0, 1\}$. If we prefer that delegation credentials explicitly imply authorization credentials, i.e. non delegable credentials, then we should define the partial order $0 \leq 1$, but in general we consider $\mathcal{T}_D$ not to be ordered.

We can now combine these two types and define a new type, $\mathcal{T}_{I \times D} := \mathcal{T}_I \times \mathcal{T}_D$, which includes both types with the natural partial order. We will describe more types in the following sections.

3 Credentials

At first sight, the information needed for a Delegation credential is the same as the information used to define an Authorization credential plus the delegation type that states whether it is a delegation or an authorization credential.In this way, we include the two types of credentials into one single concept. So hereinafter, credential is used as a general concept that comprises both authorization and delegation credentials.

If we look at the differences between delegation and authorization credentials, we see that the revocation of delegation credentials is more problematic than the revocation of authorization credentials. If we think of revocation of authorization credentials, we know this is a bottleneck for authorization frameworks. Using delegation credentials, a new problem arises however because when they are

revoked, all the authorization and delegation credentials which are linked with it have to be revoked too. This is the chain effect.

In this situation, we need some mechanisms in order to minimize the number of revocations. To do this, we introduce restrictions in the credential to be valid. These restrictions involve the validity period and in general any parameter included in the credential Type.

We are now ready to define delegation and give an appropriate notation for this concept.

Definition 2 (Delegation Credential).
A delegation credential is a tuple (I, H, A, T) *in* $\mathcal{E}^* \times \mathcal{E} \times \mathcal{A} \times \mathcal{T}$ *where,*

- $(\mathcal{E}, \leq)$ *is a partial order set representing the possible issuers and holders of delegation credentials.*
- $(\mathcal{A}, \leq)$ *is a partial order set representing the possible attributes we consider in our systems.*
- $(\mathcal{T}, \leq)$ *is a partial order set representing the additional properties of the credentials that have to be considered when deciding if a chain of credentials is valid or not.*

The set of all possible credentials is denoted by $\mathcal{C} := \mathcal{E}^* \times \mathcal{E} \times \mathcal{A} \times \mathcal{T}$

The meaning of $(Alice, Bob, Attribute, Type)$ is that *Alice* issues a credential regarding the attribute *Attribute* to *Bob*, and that this credential has type $Type$.

3.1 Validity evaluations

When defining the credentials in the system, not all the possible credentials of $\mathcal{C}$ are going to be considered as valid. Instead of defining a subset of valid credentials, we define a map from $\mathcal{C}$ to the set $\{true, false\}$ in such a way that all the valid credentials will be mapped to $true$ and the non-valid to $false$. But this map should also take into account the state or context of the system: instant of time, context of entities, location of entities, etc., as this information could interfere in the credential validation process.

Let $States$ be the set of possible states of our system. Each estate $s \in States$ encodes all the relevant contextual information for making delegation and authorization decisions. In the simplest example it is reduced to the system time, so the only relevant information for a credential to be valid is the validity interval, but in a functional system, each state should include all the relevant information for determining the validity of any credential. We define then a function

$$f : \mathcal{C} \times States \rightarrow \textbf{Boolean}$$

which decides if a given instant of time (or state) is included in a certain time interval. Using this function, a credential (I, H, A, T) is valid in state s if and only if $f(\_, \_, \_, T, s) = true$. A function f like this is a *validity evaluation.*

Definition 3 (Validity Evaluation).
Let $\mathcal{S}$ be the set of all possible states of the system. A function

$$f : \mathcal{C} \times \mathcal{S} \rightarrow \{true, false\}$$

is a validity evaluation if and only if

$$f(I, H, A, T) = true \implies f(I', H', A', T') = true$$

for all (I', H', A', T') where $I = I'$, $H' \leq H$, $A' \leq A$ and $T' \leq T$

We distinguish between two sorts of evaluations, those that depend on the subjects involved in the credential, i.e. the issuer and the holder, and those that do not depend on them. We also distinguish between those functions that depend on the attribute and those that do not depend on it.

Definition 4 (Classes of validity evaluations).
*An **objective** validity evaluation is a function which depends only on the attributes and the type of the credential,*

$$f(I, A, H, T, s) = f(I', H', T, A, s)\, \forall I', H'$$

*An **universal** validity evaluation is a function which does not depend on the attributes,*

$$f(I, A, H, T, s) = f(I, H, T, A', s)\, \forall A'$$

*A **subjective** validity evaluation is a function that is not objective, i.e. depends on the subject or the issuer of the credential.*

Objective validity evaluations do not care about the issuers or holders of the delegation credentials but are concerned with the attributes issued in each credential. On the other hand, subjective validity evaluations are affected by the entities (holders and issuers) involved in the credentials.

As an example, let us think of a reputation system and suppose that the reputation of each entity is stored in the state of the system. If we take the reputation of the entities involved in a delegation credential into account in order to decide if the chain is valid or not, then we are using a subjective validity evaluation. If, on the other hand, the entities are not considered then we are using an objective validity evaluation.

Another example of validity evaluation is to use the instant of time. In this way

$$f(I, A, H, T, s) = true \text{ iff } time(s) \leq time(T)$$

where $time(\cdot)$ gives us both the information of the state regarding time and the time component of the Type. The symbol $\leq$ represents the subclass relation (partial order) of the time component of the type. This easy schema can be used with other Type components, like location. Location can be encoded using IP addresses with and ontology encoding the subnetting relation or using geographical information [9, 10]

We encode the validity evaluation using RuleML [7] or SWRL [8] which is also supported by a Protege plugin. The definition of the last kind of validity evaluation in SWRL is trivial, as it only involves one subclass relation. With some complex examples we have to use RuleML.

4 Chaining Delegation Credentials

We mentioned before that unlike authorization credentials, delegation credentials can be chained to form a delegation chain. This consists of a sequence of delegation credentials concerning the same attribute and in which the issuer of a credential is the holder of the previous credential. Furthermore, in any given path, the issuer of a credential has to be lower, in the subclass relation, than the holder of the previous credential, formally:

Definition 5 (Delegation Path). *A sequence of delegation credentials* $\{C_i\}_{i=1}^{n}$*, where* $C_i = (I_i, H_i, A_i, T_i)$*, is a delegation path or chain for the attribute* A *if,*

1. $I_{i+1} \leq H_i$ *for all* $i \in \{1, \ldots, n\}$.
2. $A \leq A_i$ *for all* $i \in \{1, \ldots, n\}$.
3. $D \leq T_i$ *for all* $i \in \{1, \ldots, n\}$.

Where D *represents the minimal type for delegation credentials. A sequence of delegation credentials* $C := \{D_i\}_{i=1}^{n}$ *is a chain or path if there exists an attribute* $A \neq \emptyset$ *such as* C *is a delegation chain for* A*. The set of all delegation paths is denoted by* $\mathcal{P}$*.*

Condition *1*, in Definition 5, makes use of the partial order given on the set of entities. When an entity y is more specific, lower in the hierarchy, than another entity x, then y *inherits* the attributes issued to x. In the extreme situation in which the partial order is trivial, this condition is reduced to $I_{i+1} = H_i$ for all $i \in \{1, \ldots, n\}$.

Condition *2*, makes use of the partial order given on the set of attributes. When an entity x issues a credential over any attribute a, it is implicit that any other attribute a' which is more specific than a ($a' \leq a$) is also issued. Thus, we use this implicit rule to chain credentials that have some attributes in common.

Condition *3*, only establishes that all credentials in a delegation path must be delegation credentials. We use the type element D to represent the type *delegable*.

Given a path of credentials, we can map it to a single credential using a sequential operator. This operator is well defined only when the partial order sets $\mathcal{A}$ and $\mathcal{T}$ are not only partial orders but semi-lattices for the *meet* operator. In this case we take advantage of the *meet* operator for lattices to define a sequential operator for credentials.

Definition 6 (Sequential Operator). *Let* $\wedge$ *denote the meet operator of the lattices* $\mathcal{A}$ *and* $\mathcal{T}$*. Then, given two credentials* (X, Y, A, T) *and* (Y', Z, A', T') *with* $Y' \leq Y$ *we define the sequential operator as*

$$(X, Y, A, T) \wedge (Y', Z, A', T') = (X, Z, A \wedge A', T \wedge T')$$

Using this operator we give an alternative definition for credential paths

Definition 7 (Delegation Path for lattices). *Let $\{C_i\}_{i=1}^n$ be a sequence of delegation credentials, where $C_i = (I_i, H_i, A_i, T_i)$ and let $(I_P, H_P, A_P, T_P) = C_1 \wedge C_2 \wedge \ldots \wedge C_n$. The sequence is a delegation path or chain for the attribute A if,*

1. $I_{i+1} \leq H_i$ *for all* $i \in \{1, \ldots, n\}$.
2. $A \leq A_P$.
3. $D \leq T_P$.

Making use of the sequential operator we can map each credential path with a single credential. These credentials encode the meaning of the path and will be used when taking authorization and delegation decisions.

If we have a poset we may complete it with the special element $\emptyset$, that is defined as the minimal element in the poset, in such a way that the resulting set is indeed a semi-lattice for the meet operator. Then, we can use the previous definition with the extended posets.

Now we define the concept of valid credential path. We decide if a chain of credentials is valid or not, in a given state, using the same idea as with simple credentials. To do so, we define a *validity function* to decide whether a chain of credentials is valid or not.

Definition 8 (Validity Function). *Let $\mathcal{S}$ be the set of all possible states of the system. A function*

$$f : \mathcal{P} \times \mathcal{S} \rightarrow \{true, false\}$$

is a validity function if restricted to the domain $\mathcal{C} \times \mathcal{S}$ is a validity evaluation.

The first and simplest approach to determine if a path of credentials is valid is to check whether all the credentials of the path are valid in the state. Indeed, this is the least restrictive approach. So, we call it *LR validity function.*

Definition 9 (LR validity function). *Let $P := C_1 C_2 \ldots C_n$ be a chain of credentials. The Least Restrictive (LR) validity function is defined by,*

$$\begin{aligned} \hat{f} \equiv f_{LR} : \mathcal{P} \times \mathcal{S} &\longrightarrow \{true, false\} \\ (P, s) &\longmapsto \bigwedge_{i=1}^{n} f(C_i, s) \end{aligned}$$

In the simple case in which, $f(C_i, s) \equiv f(\_, \_, \_, T_i, s)$, $\hat{f}$ depends only on the types of the credentials that composed the path P. As with validity evaluations, we distinguish between Objective and Subjective validity functions.

4.1 Examples of types

We introduce here two incremental examples. We focus on the definition of the type of the credentials and on the validity evaluations and functions associated to the credentials. First of all, we define the set of States $\mathcal{S}_0$ consisting of points in time. Let define

$$\mathcal{T}_0 := \mathcal{T}_I \times \mathcal{T}_D$$

where $\mathcal{T}_I$ and $\mathcal{T}_D$ are the types defined in Section 2.3.

We define a Universal Objective validity evaluation as, $f(I, H, A, T, s) = true$ if and only if $s \in T$.

The validity function defined above is clearly universal and objective as it only depends on the type of the credentials and of course on the given state. Let us try to reduce the condition $s \in T$ to a more general condition using only the partial order. If we represent the states of $\mathcal{S}_0$ as unitary intervals:

$$\mathcal{S}_0 := \{[s, s] : s \in \mathbb{N}\}$$

then the validity function f_0 is defined as the following:

$$\begin{array}{rl} f_0 : \mathcal{P} \times \mathcal{S}_0 & \longrightarrow \{true, false\} \\ (P, s) & \longmapsto (s \leq T) \end{array} \tag{1}$$

Suppose we want to use a Multilevel security policy in which we define two security levels: *weak* and *strong*. Suppose that the *strong* is more restrictive than the *weak* level, so there could be credentials that are valid for the *weak* but not for the *strong* one. In this case, we should include the label *weak* in those credentials that are only valid in the *weak* level and the *strong* label in those which are valid in any level. This situation can be easily encoded using partial order. We define a new set of states, $\mathcal{S}_1$, that contains the level of security of the state and a point in time.

$$\mathcal{S}_1 := \mathcal{S}_0 \times \{weak, strong\}$$

Analogously, we define a new type,

$$\mathcal{T}_1 := \mathcal{T}_0 \times \{weak, strong\}$$

that is a product of partial orders, where the partial order of $\{weak, strong\}$ is defined with the inequality $weak \leq strong$. With those definitions, we define the validity evaluation, f_1, as in Equation 1.

In those cases in which we could give a meaning to $s \leq T$ we refer to f_0 as the *canonical* validity evaluation and to $\hat{f}_0$ as the *canonical* validity function.

The last example is a subjective validity function that requires a reputation system. Suppose $r(E)$ gives us the reputation of entity E as a real number in the interval $[0, 1]$. We can define a lower bound of 0.5 for the reputation of the issuer of the first credential in the path. In this way $f(P, s) = true$ if and only if $f_{LR}(P, s) = true$, $I_1 \in \mathcal{E}^*$ and $r(I_1) \geq 0.5$.

5 Conclusions

We have defined a general mathematical framework for model delegation. Although we have used a mathematical notation, the ideas presented in this paper could have been formulated using a more common language. The use of partial orders is clearly supported by ontologies, and in particular OWL offers a subclass mechanism that is well suited to the concept of partial order. So, in practice, when we talk about partial orders, we are thinking about a simple subclass ontology. More work has to be done in order to support more complex ontologies. The other interesting concept presented in this paper is the context, which is encoded in the variable *state*. All information relevant to the system is encoded using ontologies which allows us to use rule languages such as RuleML and SWRL to reason on the delegation and authorization relationships in the system.

References

1. Isaac Agudo, Javier Lopez and Jose A. Montenegro. "A Representation Model of Trust Relationships With Delegation Extension". In 3rd International Conference on Trust Management, iTrust 2005, volume 3477 of Lecture Notes in Computer Science, pages 116 - 130. Springer, 2005.
2. Aditya Kalyanpur, Bijan Parsia, Evren Sirin, Bernardo Cuenca-Grau and James Hendler, "Swoop: A Web Ontology Editing Browser", Journal of Web Semantics Vol 4(2), 2005
3. D.F. Ferraiolo, D.R. Kuhn and R. Chandramouli, "Role Based Access Control", Artech House, 2003.
4. C. Ellison, B. Frantz, B. Lampson, R. Rivest, B. Thomas and T. Ylonen "SPKI Certificate Theory", *RFC 2693*, 1999.
5. Holger Knublauch, Ray W. Fergerson, Natalya F. Noy and Mark A. Musen "The Protege OWL Plugin: An Open Development Environment for Semantic Web Applications" Third International Semantic Web Conference - ISWC 2004, Hiroshima, Japan, 2004.
6. S. Bechhofer et al., "OWL Web Ontology Language Reference". 2004.
7. Boley, H., "The Rule Markup Language: RDF-XML Data Model, XML Schema Hierachy, and XSL Transformations", Invited Talk, INAP2001, Tokyo, Springer-Verlag, LNCS 2543, 5-22, 2003.
8. "SWRL: A Semantic Web Rule Language Combining OWL and RuleML". W3C Member Submission. 21-May-2004.
9. B. Purevjii, T. Amagasa, S. Imai, and Y. Kanamori. "An Access Control Model for Geographic Data in an XML-based Framework". In Proc. of the 2nd International Workshop on Information Systems Security (WOSIS), 2004, pages 251–260.
10. V. Atluri and P. Mazzoleni. "A Uniform Indexing Scheme for Geo-spatial Data and Authorizations". In Proc. of the Sixteenth Conf. on Data and Application Security, IFIP TC11/WG11.3, Cambridge, UK, 2002, pages 207–218.