Specification of History Based Constraints for Access Control in Conceptual Level*

Fathiyeh Faghih, Morteza Amini, and Rasool Jalili

Dept. of Comp. Eng., Sharif University of Technology
Tehran, Iran
{faghih@ce,m_amini@ce,jalili}@sharif.edu

Abstract. An access control model for Semantic Web should take the semantic relationships among the entities, defined in the abstract conceptual level (i.e., ontology level), into account. Authorization and policy specification based on a logical model let us infer implicit security policies from the explicit ones based on the defined semantic relationships in the domains of subjects, objects, and actions. In this paper, we propose a logic based access control model for specification and inference of history-constrained access policies in conceptual level of Semantic Web. The proposed model (named TDLBAC-2) enables authorities to state policy rules based on the history of users' accesses using a temporal description logic called $\mathcal{DLR}_{US}$. The expressive power of the model is shown through seven different patterns for stating history-constrained access policies. The designed access decision algorithm of the model leverages the inference services of $\mathcal{DLR}_{US}$, which facilitates the implementation of an enforcement system working based on the proposed model. Sound inference, history-awareness, ability to define access policies in conceptual level, and preciseness are the main advantages of the proposed model.

1 Introduction

A proper access control model for a semantic-aware environment like Semantic Web should consider the semantic relationships among entities in the subject, object, and action domains. The relationships are defined as ontologies in Semantic Web, and should be utilized in inferring the authorized accesses from the explicit policy rules. On the other hand, considering the history of accesses may be important in access control in many modern applications such as e-banking systems. To consider these two aspects, we propose an access control model based on a temporal description logic ($\mathcal{DLR}_{US}$).

An access control model named as SBAC (Semantic Based Access Control) is introduced in [1] for Semantic Web, with the aim of taking semantic relationships into account . The SBAC model is extended in [2] to express policies with constraints on history of users' accesses in the past. TDLBAC in [3] tries to remove

* Thanks to ITRC (Iran Telecommunication Research Center) for partial support of this work.

S. Jha and A. Maturia (Eds.): ICISS 2010, LNCS 6503, pp. 186–200, 2010.

two limitations of this extension. The first one is its restriction to the policies at the level of individuals. Therefore, security authorities are unable to state policies at the concept level. The second limitation is that the extension utilizes a formal language without proof theory that is only used for stating policy constraints. TDLBAC improves this model through a logical framework. Expressing security policy rules is possible using a logic-based language in TDLBAC.

Logic-based access control has many advantages including abstraction from implementation, expressiveness, and verifiability [4,5]. For Semantic Web, using logic can help to infer implicit authorized accesses from defined policies based on defined relationships among entities. On the other hand, description logic (DL) has always been considered as the foundation of Semantic Web for definition, integration, and maintenance of ontologies [6]. Accordingly, DL is chosen as a basis for proposing an access control model for Semantic Web in TDLBAC.

The other aspect in designing TDLBAC is access control with constraints on history of users' accesses. DLs in their standard forms are unable to state dynamic aspects. In order to state the history constrained policies, a temporal extension of a DL, named $\mathcal{TL}$-$\mathcal{ALCF}$ is used in TDLBAC. The underlying logic in TDLBAC ($\mathcal{TL}$-$\mathcal{ALCF}$) limits it to expression of a special pattern of policy rules with history based constraints. In this paper, we improve TDLBAC by changing the underlying logic to $\mathcal{DLR}_{US}$, which enables us to express various patterns of policy rules with constraints on history of accesses.

The proposed model in this paper is named TDLBAC-2. The model is introduced with its components and the procedure for storing the history of users' accesses. The access control algorithm is designed based on the inference services of $\mathcal{DLR}_{US}$. It enables us to discuss the complexity of the algorithm based on the inference services. It will be also helpful in implementation of the model using an available theorem prover for the $\mathcal{DLR}_{US}$ logic.

The rest of the paper is organized as follows: In section 2, TDLBAC-2 is introduced with its components, and the patterns for definition of security rules. Section 3 explains the access control procedure. Section 4 is a brief discussion on the complexity of the access control algorithm. Section 5 evaluates TDLBAC-2 focusing on its comparison with TDLBAC. Section 6 reviews the related work to the paper, and section 7 concludes the paper.

2 TDLBAC-2

2.1 Preliminaries

In this section, we briefly introduce the employed logic, $\mathcal{DLR}_{US}$. It is a temporal description logic proposed by Artale et al. in [7], [8]. The logic is based on the decidable DL, $\mathcal{DLR}$. There are efficient and complete algorithms for this logic which are used in the real applications. Considering the desirable characteristics of $\mathcal{DLR}$, Artale et al. tried to add the temporal extension to this logic.

$\mathcal{DLR}_{US}$ is designed as a combination of $\mathcal{DLR}$ and propositional linear temporal logic with the operators *Since* and *Until*. In this logic, temporal operators are applied to all syntactical expressions including concepts, relations and schemas.

The previous attempts for adding temporal aspect include weaker languages in the family of DLs which contain only binary relations. In $\mathcal{DLR}_{US}$, the flow of time, $\mathcal{T} = \langle \mathcal{T}_p, < \rangle$, in which $\mathcal{T}_p$ is a set of temporal points, and $<$ is a binary relation over $\mathcal{T}_p$, is considered to be equivalent to $\langle \mathbb{Z}, < \rangle$. Each time point is representative of a time interval (between this point and the next one.)

$\mathcal{DLR}$ is the non-temporal fragment of $\mathcal{DLR}_{US}$. For entity and relation expressions, it includes all the Boolean expressions, as well as the selection and projection expressions. The selection expression $U_i/n : E$ states an n-ary relation in which the argument U_i is of type E. The parameter name can be used instead of stating its position. The projection expression $(\exists^{\leqslant k}[j]R)^{\mathcal{I}(t)}$ is a projection of the relation R over its jth argument with respect to the cardinality relation ($\leqslant k$). The basic syntactical types in $\mathcal{DLR}_{US}$ include concepts and n-ary relations for $n \geq 2$. The semantics of $\mathcal{DLR}_{US}$ includes the temporal models, which are the triples $\mathcal{I} \doteq \langle \mathcal{T}, \Delta^{\mathcal{I}}, .^{\mathcal{I}(t)} \rangle$. $\Delta^{\mathcal{I}}$ is a nonempty set of objects (the domain of $\mathcal{I}$) and $.^{\mathcal{I}}$ is an interpretation function such that for every $t \in \mathcal{T}$, every concept E, and every n-ary relation R, $E^{\mathcal{I}(t)} \subseteq \Delta^{\mathcal{I}}$ and $R^{\mathcal{I}(t)} \subseteq (\Delta^{\mathcal{I}})^n$. In the following you can see a part of the syntax and semantic of $\mathcal{DLR}_{US}$ as used in the design of TDLBAC-2. The intuitive meanings of $\Diamond^+$, $\Diamond^-$, and $\oplus$ are "sometime in the future", "sometime in the past", and "at the next moment". Note that $\top_n$ is the top n-ary relation.

$$(\top_n)^{\mathcal{I}(t)} \subseteq (\Delta^{\mathcal{I}})^n, R^{\mathcal{I}(t)} \subseteq (\top_n)^{\mathcal{I}(t)}, (\neg R)^{\mathcal{I}(t)} \subseteq (\top_n)^{\mathcal{I}(t)} \backslash R^{\mathcal{I}(t)}$$

$$(U_i/n : E)^{\mathcal{I}(t)} = \left\{ \langle d_1, ..., d_n \rangle \in (\top_n)^{\mathcal{I}(t)} | d_i \in E^{\mathcal{I}(t)} \right\}$$

$$(\Diamond^+ R)^{\mathcal{I}(t)} = \{ \langle d_1, ..., d_n \rangle \in (\top_n)^{\mathcal{I}(t)} | \exists v > t. (\langle d_1, ..., d_n \rangle \in R^{\mathcal{I}(v)}) \}$$

$$(\oplus R)^{\mathcal{I}(t)} = \{ \langle d_1, ..., d_n \rangle \in (\top_n)^{\mathcal{I}(t)} | \langle d_1, ..., d_n \rangle \in R^{\mathcal{I}(t+1)} \}$$

$$(\Diamond^- R)^{\mathcal{I}(t)} = \{ \langle d_1, ..., d_n \rangle \in (\top_n)^{\mathcal{I}(t)} | \exists v < t. (\langle d_1, ..., d_n \rangle \in R^{\mathcal{I}(v)}) \}$$

$$(\exists^{\leqslant k}[j]R)^{\mathcal{I}(t)} = \left\{ d \in \top^{\mathcal{I}(t)} | \#\{ \langle d_1, ..., d_n \rangle \in R^{\mathcal{I}(t)} | d_j = d \} \leqslant k \right\}$$

2.2 Overall TDLBAC-2 Model Description

TDLBAC-2 is an access control model based on the temporal description logic $\mathcal{DLR}_{US}$. The model includes two main DL components; TBox and ABox.

TBox contains statements expressing properties and relationships among entities. Considering the nature of subsumption relationship which is the essential operator in TBox, this component has a lattice structure [9]. ABox includes assertions about the real world instances.

Since TDLBAC-2 is a history based access control model, it contains two essential components from the "history based access control" point of view:

1. History Base (HB): contains the history of previous accesses. HB is stored as a part of ABox.
2. History-based Policy-Base (HPB): is a set of security policy rules with constraints on history of accesses. The rules of HPB are stored in TBox.

2.3 Model Terminology

In TDLBAC-2, the axioms related to the domain are stored in TBox. The axioms include the definition of different *concepts*, in addition to their semantic relationships. TBox contains three hierarchies of concepts:

1. Subjects: Instances of these concepts are potential active users/agents/ webservices that require to access the objects.
2. Objects: Instances of these concepts can be accessed by subjects.
3. Actions: Instances of these concepts identify different access types.

Each security rule is defined as a logical *relation* and is stored in TBox. Security rules are defined by specifying the sorts of the subjects that have permission of a special type of access on the specific objects. In TDLBAC-2, security rules are defined as ternary relations among subjects, objects and access types. To specify the type of each access parameter, the selection operator $(U_i/n : E)$ in $\mathcal{DLCR}_{US}$ is utilized. The selection operator states an n-ary relation that one of its parameters, called $U_i (i \leq n)$ is of type E. Using this operator, we can specify the types of the three parameters of a security rule. The pattern for defining security rules in TDLBAC-2 is as follows:

$$(\text{policy-rule})_{\text{i}} \equiv (\text{PS}_1/3 : \text{SType}) \sqcap (\text{PO}_2/3 : \text{OType}) \sqcap (\text{PA}_3/3 : \text{AType})$$

In this pattern, $(\text{policy-rule})_{\text{i}}$ is a name given to the defined security rule. SType is a concept in the hierarchy of subjects, OType is a concept in the objects hierarchy, and AType belongs to the actions hierarchy.

TDLBAC-2 is an access control model which allows stating the history constrained policy rules. Therefore, in TDLBAC-2, there is a history of accesses similar to the policy rules. Each access logged in the history is a ternary relation between specific types of subjects, objects, and actions. To define the types of an access parameters, the selection operator $(U_i/n : E)$ is utilized in the access definition. The parameters of accesses are named as AS, AO, and AA. The pattern for access definition in TDLBAC-2 is as follows:

$$\text{access}_{\text{i}} \equiv (\text{AS}_1/3 : \text{SType}) \sqcap (\text{AO}_2/3 : \text{OType}) \sqcap (\text{AA}_3/3 : \text{AType})$$

access_{i} is a name given to the logged access. In this paper, we simply only use the name of the argument and omit the indexes (for example, AS instead of AS_1).

2.4 Ground Assertions

In a DL based system, ground assertions are stored in ABox. These facts are grouped into three categories in TDLBAC-2:

- The assertions which state correspondence of instances to the concepts defined in subjects, objects, and actions hierarchies.
- The assertions which state the users' previous accesses and are stored in HB as a part of ABox. These assertions are utilized for checking the history based constraints of security policy rules in the access control procedure. These assertions are added to ABox after granting each request.

– The assertions which are related to the type of requester in each request. They are added to ABox based on the certificates that the requester attaches to her request. This is discussed in Section 3 in details.

2.5 Security Policy Rules Definition

TDLBAC-2 allows security authorities to define history constrained policy rules. These rules specify the authorized types of subjects, objects, and actions, as well as the constraints defined over the previous accesses. In most of the access control systems based on the history of accesses (like e-government systems), the constraints are enforced on the history of subjects' accesses. For instance, consider a security rule which gives the right of attending in the second round of the election to those who have voted in the first round. The idea of expressing history constrained policy rules in TDLBAC-2 is stating the history based constraint of the rule besides the permitted type of the subject. The history based constraints are stated using the operators defined in the syntax of $\mathcal{DLR}_{US}$ logic. The operator which is utilized the most in stating the history constrained policy rules is $\Diamond^-$. The informal meaning of this operator is "sometime in the past".

To limit the subjects of a security rule to those who had a special access in past, the projection expression is used. As seen in the semantics, the interpretation of the projection expression is a set of instances for which the number of relations of type R having these instances as their ith argument, satisfies the specified relation with k (equals, less than or greater than). This operator can be used to specify the number and type of users' accesses.

2.6 Patterns of History Constrained Policy Rules

The main purpose of this paper is investigation into the specification of history constrained policies in conceptual level using $\mathcal{DLR}_{US}$. Therefore, in this section, different patterns to define possible security policy rules in TDLBAC-2 (based on $\mathcal{DLR}_{US}$) are introduced:

Pattern 1. Pattern 1 is used to state the rules which authorize those users who had special accesses in the past. In the other words, special accesses have been registered by these users in the history base (HB). The pattern of stating such policy rules is as follows:

$$(\text{policy-rule})_{\text{i}} \equiv (\text{PS/3:}(\text{SType} \sqcap \Diamond^- \exists^{\geqslant 1}[\text{AS}](\text{access}_{\text{i}}))) \sqcap (\text{PO/3:OType}) \sqcap (\text{PA/3:AType})$$

$(\text{policy-rule})_{\text{i}}$ is a policy rule that gives the SType users, the permission of AType accesses to objects of type OType with the constraint of having an access of type access_{i} in the past. The constraint on the existing an access of type access_{i} is defined by $(\Diamond^- \exists^{\geqslant 1}[\text{AS}](\text{access}_{\text{i}}))$ in this rule. This condition is conjuncted to the other condition of the subject on its type (i.e., SType). To clarify the meaning, let's take a look at the semantics of the history based constraint in this pattern:

$$(\Diamond^{-}\exists^{\geqslant 1}[AS]access_i)^{\mathcal{I}(t)} = \left\{d \in (\top)^{\mathcal{I}(t)} | \exists v < t. d \in (\exists^{\geqslant 1}[AS]access_i)^{\mathcal{I}(v)}\right\} =$$
$$\left\{d \in (\top)^{\mathcal{I}(t)} | \exists v < t. \# \left\{ \langle d_{AS}, d_{AO}, d_{AA}\rangle \in (access_i)^{\mathcal{I}(v)} | d_{AS} = d\right\} \geqslant 1\right\}$$

Interpretation of the constraint at time t contains instances (in the domain of $\mathcal{I}$) which are the AS argument of one or more relations of type access_i in a time point less than t.

As an example of this pattern, suppose the security rule in a second round election which gives the permission of voting to the residents of the country (instances of type resident). Moreover, the right of voting is limited to those who have voted in the first round. The security policy rule for this election can be stated as follows:

$$\text{vote-policy-2nd} \equiv \text{PS/3:(Resident} \sqcap \Diamond^{-}\exists^{\geqslant 1}[\text{AS}](\text{vote-1st}))$$
$$\sqcap \text{PO/3:election-system} \sqcap \text{PA/3:vote}$$

vote-1st is a relation which specifies the type of an access. Regarding this policy rule, a requester is authorized, if he/she is the AS argument (subject) of an access of type *vote-1st*, which can be defined in TBox as follows:

$$\text{vote-1st} \equiv \text{AS/3:resident} \sqcap \text{AO/3:election-system} \sqcap \text{AA/3:vote}$$

Pattern 2. Pattern 2 is used to state the policy rules which authorize the subjects who had a given number of accesses of special type.

$$(\text{policy-rule})_\text{i} \equiv \text{PS/3:(SType} \sqcap \exists^{\leqslant n}[\text{AS}](\Diamond^{-}\text{access}_\text{i})) \sqcap \text{PO/3:OType} \sqcap \text{PA/3:AType}$$

The cardinality in the projection expression is utilized to express the limitation on the number of accesses (equals, less than, or greater than a given number). To clarify, let us take a look at the semantics of the constraint in this pattern:

$$(\exists^{\leqslant n}[AS](\Diamond^{-}access_i))^{\mathcal{I}(t)} =$$
$$\left\{d \in (\top)^{\mathcal{I}(t)} | \# \left\{ \langle d_{AS}, d_{AO}, d_{AA}\rangle \in (\Diamond^{-}access_i)^{\mathcal{I}(t)} | d_{AS} = d\right\} \leqslant n\right\} =$$
$$\left\{d \in (\top)^{\mathcal{I}(t)} | \# \left\{ \begin{array}{c} \langle d_{AS}, d_{AO}, d_{AA}\rangle \in (\top_3)^{\mathcal{I}(t)} | \exists v < t. \\ \langle d_{AS}, d_{AO}, d_{AA}\rangle \in (access_i)^{\mathcal{I}(v)} \wedge d_{AS} = d \end{array} \right\} \leqslant n\right\}$$

For example, the permission of taking the entrance exam can be given to those who took it less than "y" times.

Pattern 3. Pattern 3 is used to state the policy rules which give the permission to the subjects who had accesses in some time interval in the past for specified number of times. As mentioned in "Preliminaries" section, time in $\mathcal{DLCR}_{US}$ is considered as a partial order, $\mathcal{T} = \langle \mathcal{T}_p, < \rangle$, in which $\mathcal{T}_p$ is a set of time points. Moreover, each time point is representative of a time interval defined in the logic semantics. The pattern for stating such policy rules is as follows:

$$(\text{policy-rule})_\text{i} \equiv \text{PS/3:(SType} \sqcap \Diamond^{-}\exists^{\leqslant n}[\text{AS}](\text{access}_\text{i})) \sqcap \text{PO/3:OType} \sqcap \text{PA/3:AType}$$

To clarify, let us take a look at the semantics of the constraint defined in the above rules:

$$(\Diamond^{-}\exists^{\leqslant n}[AS]access_i)^{\mathcal{I}(t)} = \left\{d \in (\top)^{\mathcal{I}(t)} | \exists v < t.d \in (\exists^{\leqslant n}[AS]access_i)^{\mathcal{I}(v)}\right\} =$$
$$\left\{d \in (\top)^{\mathcal{I}(t)} | \exists v < t.\#\left\{\langle d_{AS}, d_{AO}, d_{AA}\rangle \in (access_i)^{\mathcal{I}(v)} | d_{AS} = d\right\} \leqslant n\right\}$$

So an authorized subject has been the subject of a special kind of relations (actions) for more or less than a specified number of times in a specific time interval (the time point v is representative of this interval) . As an example, we can state a policy rule which gives the permission of undertaking a special kind of projects to those, who have the record of doing more than two projects in a year (in a model that each time point is representative of a year).

Pattern 4. Pattern 4 is used to state the policy rules which give a permission to ones who had accesses in some consecutive intervals. The pattern is as follows:

$$(\text{policy-rule})_{\text{i}} \equiv \text{PS/3:(SType} \sqcap \Diamond^{-}(\exists^{\geqslant 1}[\text{AS}]\text{access}_{\text{i}} \sqcap \exists^{\geqslant 1}[\text{AS}] \oplus \text{access}_{\text{j}}$$
$$\sqcap \exists^{\geqslant 1}[\text{AS}] \oplus \oplus \text{access}_{\text{k}} \sqcap ...)) \sqcap \text{PO/3:OType} \sqcap \text{PA/3:AType}$$

access_{i}, access_{j}, and access_{k} are representatives of three accesses in three consecutive intervals in the past. The semantics of the constraint on history of accesses in this pattern is as follows:

$$(\Diamond^{-}(\exists^{\geqslant 1}[\text{AS}]\text{access}_{\text{i}} \sqcap \exists^{\geqslant 1}[\text{AS}] \oplus \text{access}_{\text{j}} \sqcap \exists^{\geqslant 1}[\text{AS}] \oplus \oplus \text{access}_{\text{k}}))^{\mathcal{I}(t)} =$$
$$\left\{ d \in (\top)^{\mathcal{I}(t)} | \exists v < t.d \in (\exists^{\geqslant 1}[\text{AS}]\text{access}_{\text{i}} \sqcap \exists^{\geqslant 1}[\text{AS}] \oplus \text{access}_{\text{j}} \sqcap \exists^{\geqslant 1}[\text{AS}] \oplus \oplus \text{access}_{\text{k}})^{\mathcal{I}(v)} \right\}$$
$$= \left\{ d \in (\top)^{\mathcal{I}(t)} | \exists v < t. \begin{cases} \#\{\langle d_{AS}, d_{AO}, d_{AA}\rangle \in (\text{access}_{\text{i}})^{\mathcal{I}(v)} | d_{AS} = d\} \geqslant 1 \\ \wedge \#\{\langle d_{AS}, d_{AO}, d_{AA}\rangle \in (\text{access}_{\text{j}})^{\mathcal{I}(v+1)} | d_{AS} = d\} \geqslant 1 \\ \wedge \#\{\langle d_{AS}, d_{AO}, d_{AA}\rangle \in (\text{access}_{\text{k}})^{\mathcal{I}(v+2)} | d_{AS} = d\} \geqslant 1 \end{cases} \right\}$$

The pattern is extensible for more consecutive intervals in a similar way. As an example, using this pattern, we can define a policy rule which gives the permission of using insurance services to those, who have paid the insurance fee for at least three consecutive months.

Pattern 5. Pattern 5 is used to state the policy rules which give permission to those who had special accesses with a certain order in the past.

$$(\text{policy-rule})_{\text{i}} \equiv \text{PS/3:(SType} \sqcap \Diamond^{-}(\exists^{\geqslant 1}[\text{AS}]\text{access}_{\text{i}} \sqcap \Diamond^{+}\exists^{\geqslant 1}[\text{AS}]\text{access}_{\text{j}}))$$
$$\sqcap \text{PO/3:OType} \sqcap \text{PA/3:AType}$$

In this pattern, the history constraint is defined over the subjects, in the way that it should be the instance of two access types, access_{i} and access_{j}. Moreover, the access_{i} instance should precede the access_{j} instance. By nesting the $\Diamond^{+}$ and $\Diamond^{-}$ operators in this way, we can provide the concept of precedence relation

between the accesses in the past, since $\Diamond^+$ shows "sometime in the future", relative to "sometime in the past", which is provided by $\Diamond^-$. Let us take a look at the semantics of the constraint on the history of accesses in this pattern:

$$(\Diamond^-(\exists^{\geqslant 1}[\mathrm{AS}]\mathrm{access_i} \sqcap \Diamond^+\exists^{\geqslant 1}[\mathrm{AS}]\mathrm{access_j}))^{\mathcal{I}(t)} =$$
$$\left\{ d \in (\top)^{\mathcal{I}(t)} | \exists v < t. d \in (\exists^{\geqslant 1}[AS]access_i \sqcap \Diamond^+\exists^{\geqslant 1}[\mathrm{AS}]\mathrm{access_j})^{\mathcal{I}(v)} \right\} =$$
$$\left\{ d \in (\top)^{\mathcal{I}(t)} | \exists v < t. \left\{ \begin{array}{l} \# \left\{ \begin{array}{l} \langle d_{AS}, d_{AO}, d_{AA} \rangle \in \\ (access_i)^{\mathcal{I}(v)} | d_{AS} = d \end{array} \right\} \geqslant 1 \wedge \exists v < w. \\ \# \left\{ \begin{array}{l} \langle d'_{AS}, d'_{AO}, d'_{AA} \rangle \in \\ (access_j)^{\mathcal{I}(w)} | d'_{AS} = d \end{array} \right\} \geqslant 1 \end{array} \right\} \right\}$$

As an example of this pattern, access to the electronic payment page of a banking system is allowable, only if the client has accessed an online store, and then inserted the information of her credit card.

Meta Pattern 6. Definition of different accesses separately may cause TBox to be extremely large in some systems. Meta pattern 6 is a method to alleviate this problem. We explain this pattern with its application to pattern 3:

$$(\text{policy-rule})_\mathrm{i} \equiv \mathrm{PS/3{:}(SType} \sqcap \exists^{\leqslant n}\Diamond^-[\mathrm{AS}](\mathrm{accesss_i} \sqcap \mathrm{AS/3{:}S''Type}$$
$$\sqcap \mathrm{AO/3{:}O''Type} \sqcap \mathrm{AA/3{:}A''Type})) \sqcap \mathrm{PO/3{:}OType} \sqcap \mathrm{PA/3{:}AType}$$

$$\mathrm{access_i} \equiv \mathrm{AS/3{:}S'Type} \sqcap \mathrm{AO/3{:}O'Type} \sqcap \mathrm{AA/3{:}A'Type}$$

In this pattern, $\mathrm{access_i}$ is defined as an access with general types of access parameters. In the definition of $(\text{policy-rule})_\mathrm{i}$, the types of subject, object, and action are defined more specifically. In this pattern, the relation between S″Type, O″Type, and A″paType with S′Type, O′Type, and A′Type is as follows:

$$(\mathrm{S''Type} \sqcap \mathrm{S'Type} \neq \varnothing) \wedge (\mathrm{O''Type} \sqcap \mathrm{O'Type} \neq \varnothing) \wedge (\mathrm{A''Type} \sqcap \mathrm{A'Type} \neq \varnothing)$$

This way, we can save the volume of the ontology related to the definition of accesses. This meta pattern can be similarly applied to other patterns too. For example, consider the action of repaying a loan in a banking system as follows:

$$\text{repay-loan} \equiv \mathrm{AS/3{:}account\text{-}holder} \sqcap \mathrm{AO/3{:}loan} \sqcap \mathrm{AA/3{:}repay}$$

Now, consider a security rule, which gives the permission of getting the secured loan to those who already repaid at least three unsecured loans. The security loan policy rule can be defined as follows:

$$\text{secured-loan-policy} \equiv \mathrm{PS/3} : (\text{account-holder} \sqcap \Diamond^-\exists^{\geqslant 3}[\mathrm{AS}](\text{repay-loan}$$
$$\sqcap \mathrm{AO/3} : \text{unsecured-loan})) \sqcap \mathrm{PO/3{:}secured\text{-}loan} \sqcap \mathrm{PA/3{:}get}$$

"AO/3 : unsecured-loan" expresses the relations in which the AO parameter is of type unsecured-loan. This condition is conjuncted to the "repay-loan" access type, which is itself a relation.

Meta Pattern 7. Meta Pattern 7 is used to state the negation of history based constraints. The pattern might be applied to patterns 1 to 4 as follows:

- Applying pattern 7 to pattern 1 is as follows:

$$(\text{policy-rule})_{\text{i}} \equiv \text{PS/3:(SType} \sqcap \neg \Diamond^{-} \exists^{\geqslant 1} [\text{AS}](\text{access}_{\text{i}})) \sqcap \text{PO/3:OType} \sqcap \text{PA/3:AType}$$

  For example, using this pattern, we can state the policy rule which gives the permission of getting the loan to those who have not already got any loan.
- Applying pattern 7 to pattern 2 is as follows:

$$(\text{policy-rule})_{\text{i}} \equiv \text{PS/3:(SType} \sqcap \neg \exists^{\leqslant n} [\text{AS}](\Diamond^{-} \text{access}_{\text{i}})) \sqcap \text{PO/3:OType} \sqcap \text{PA/3:AType}$$

  As an example of this pattern, we can mention a policy rule, which gives the permission of registration in an education system to those who have not already registered more than 11 times in the system.
- Applying pattern 7 to pattern 3 is as follows:

$$(\text{policy-rule})_{\text{i}} \equiv \text{PS/3:(SType} \sqcap \neg \Diamond^{-} \exists^{\leqslant n} [\text{AS}](\text{access}_{\text{i}})) \sqcap \text{PO/3:OType} \sqcap \text{PA/3:AType}$$

  As an example, consider a rule in a banking system, based on which only those who have not withdrawn from their accounts more than five times in any of the previous years can fill loan application forms.
- Applying pattern 7 to pattern 4 is as follows:

$$(\text{policy-rule})_{\text{i}} \equiv \text{PS/3:(SType} \sqcap \neg \Diamond^{-} \exists^{\geqslant 1} [\text{AS}](\text{access}_{\text{i}} \sqcap \oplus \text{access}_{\text{j}} \sqcap \oplus ...)) \sqcap \text{PO/3:OType} \sqcap \text{PA/3:AType}$$

  For example, consider a rule in an education system for students who want to register as teaching assistantships. This rule might restrict the permission to those who did not pass two consecutive terms with GPA lower than 16.

To clarify the meaning of pattern 7, the semantics of the history constraint $(\neg \exists^{\geqslant n} [AS](\Diamond^{-} \text{access}_{\text{i}}))$ in application of this pattern to pattern 2 is formalized:

$$(\neg \exists^{\leqslant n} [\text{AS}](\Diamond^{-} \text{access}_{\text{i}}))^{\mathcal{I}(t)} = \top^{\mathcal{I}(t)} \backslash (\exists^{\leqslant n} [\text{AS}](\Diamond^{-} \text{accesss}_{\text{i}}))^{\mathcal{I}(t)} =$$

$$\top^{\mathcal{I}(t)} \backslash \left\{ \text{d} \in \top^{\mathcal{I}(t)} | \#\{\langle \text{d}_1, ..., \text{d}_{\text{n}} \rangle \in (\Diamond^{-} \text{accesss}_{\text{i}})^{\mathcal{I}(t)} | \text{d}_{\text{AS}} = \text{d}\} \leqslant \text{n} \right\} =$$

$$\top^{\mathcal{I}(t)} \backslash \left\{ \begin{array}{l} \text{d} \in \top^{\mathcal{I}(t)} | \#\{\langle \text{d}_1, ..., \text{d}_{\text{n}} \rangle \in (\top_n)^{\mathcal{I}(t)} | \exists v < t.\ \langle \text{d}_1, ..., \text{d}_{\text{n}} \rangle \in (\text{accesss}_{\text{i}})^{\mathcal{I}(v)} \\ \wedge \text{d}_{\text{AS}} = \text{d}\} \leqslant \text{n} \end{array} \right\}$$

The permission is given to those who did not have $\leqslant n$ access(es) of type access_{i}.

3 Access Control Procedure

A formal procedure for handling an access request in TDLBAC-2 is as follows.

An access request is interpreted as a triple $(\text{s}_{\text{r}}, \text{o}_{\text{r}}, \text{a}_{\text{r}})$, where s_{r} is the subject of the request, o_{r} is the object , and a_{r} is the requested access type. Two conditions should be satisfied for applicability of a security rule in HPB to a request:

- **Type Compatibility Condition:** The types of the access request parameters should be compatible with the types of the parameters specified in the the policy rule. Suppose, $\mathrm{sc_r}$, $\mathrm{oc_r}$, and $\mathrm{ac_r}$ are the types of the subject, object, and action parameters in the request, and $\mathrm{sc_p}$, $\mathrm{oc_p}$, and $\mathrm{ac_p}$ are the types of the corresponding parameters in the policy rule. A policy rule is applicable to an access request, if: $sc_r \sqsubseteq sc_p \wedge oc_r \sqsubseteq oc_p \wedge ac_r \sqsubseteq ac_p$
  In other words, the types of the parameters in the request should be subsumed by the types of the corresponding parameters in the security rule.
- **History Constraint:** The history constraint in the policy rule definition should be satisfied w.r.t. the facts inserted to the history base.

In the following, we first discuss how to store the history of accesses. Then, the proposed algorithm for access control is described. To clarify the whole procedure, an example is used.

Example: Consider an election system in a country. The election-system is a concept in the objects hierarchy in TBox. This concept has an instance in each city which can be accessed via the web. Suppose that the system is used in a presidential election in the second round. The security rule is to limit the voters to a group of the residents who voted in the first round. *Resident* is a concept in the subjects hierarchy. The security rule can be stated in HPB as follows:

$$\text{vote-policy-2nd} \equiv \text{PS/3:(resident} \sqcap \Diamond^{-} \exists^{\geqslant 1} [\text{AS}](\text{vote-1st})) \sqcap \text{PO/3:election-system} \sqcap \text{PA/3:vote}$$

The concept of vote-1st is defined in TBox as follows:

$$\text{vote-1st} \equiv \text{AS/3:resident} \sqcap \text{AO/3:election-system} \sqcap \text{AA/3:vote}$$

3.1 Storing the History of Accesses in HB

To describe the process of storing the history of accesses in HB, we use the election system example. The requests for voting in the first round are processed with respect to the security rules in HPB. For every request granted by the system, the facts related to the corresponding access will be inserted to ABox.

Suppose that "John Day" has a request for voting in the first round of presidential election in an instance of the election-system concept, named as election-sub20. The request is granted after being processed by the access control system. The next step is to store the access in HB. To store the access with its parameters, this statement is inserted to ABox: vote-1st(12345,election-sub20,$\mathrm{v_1}$,$\mathrm{t_1}$). $\mathrm{v_1}$ is an instance of the vote action which is delivered to the system as a parameter of the request. The point is that we have *UNA* (Unique Name Assumption) for subjects in TDLBAC-2, similar to the case in TDLBAC. Therefore, the uniqueness of the subjects in storing and retrieving is guaranteed. For instance, in the example of the election system, the national ID of people is used for this purpose. In the stored statement, 12345 is considered as John's national ID. The parameter $\mathrm{t_1}$ in the stored statement is the time of granting the access request. Using this parameter in storing an instance in ABox is necessary in $\mathcal{DLR}_{US}$.

Generally speaking, suppose that acc is the name of the relation related to the performed access, and s_r, o_r, and a_r are the instances forming the subject, object, and action parameters of the access request respectively. To store the access occurred in t_1, this statement is inserted to ABox: acc(s_r,o_r,a_r,t_1). Using this technique, each access could be distinguished by its four parameters in ABox.

3.2 Access Control Algorithm

To clarify the proposed algorithm, we continue with the election system example. Suppose that "John Day" wants to vote in the second round of the presidential election. Instances of the object and action are known in the server side. However, all the potential users are not known by the access control system. Subject identification is based on the certificates that the user attaches to her request. The system is able to map the requester to the concepts in the hierarchy of subjects using the delivered certificates. For instance, "John Day" might be mapped to the *resident* concept by one of his certificates, and to the *graduate* concept by the other one. It is important that a unique certificate be included in the attached certificates, so that *UNA* can be satisfied. Corresponding statements are inserted to ABox to be used in the access control procedure. For example, using 12345 as John's national ID, the statements resident(12345,t_2) and graduate(1234,t_2) are inserted to ABox before starting the access control procedure (in t_2).

Checking the applicability of each security rule with respect to the access request is performed using *instance checking* inference service. In our proposed algorithm, this service checks whether the triple related to the requested access is a member of the relation defined with respect to a security rule or not. Suppose, in the election system example, the parameters of the access request (voting in the second round) are 12345, election-sub30, and v_2 respectively. In this case, to check the applicability of the security rule "vote-policy-2nd" to the access request, the *instance checking* inference service will be used to check the satisfiability of vote-policy-2nd(12345,election-sub30,v_2,t_3).

Generally speaking, if s_r, o_r, and a_r are the parameters of the access request, and $(\text{policy-rule})_i$ is the security rule, the access control algorithm will be performed using the $(\text{policy-rule})_i(s_r,o_r,a_r,t)$ inference service. If this inference service returns true, the two following conditions of compatibility between the security rule and the access request are satisfied:

1. **Type Compatibility Condition:** The *instance checking* inference service returns true, if the three type conditions of $(\text{policy-rule})_i$ i.e., (PS/3:SType$\sqcap$ PO/3:OType$\sqcap$ PA/3:AType) are satisfied with respect to the three parameters of the access request, (s_r,o_r,a_r). The facts related to the types of the object and access parameters are already in ABox. The facts related to the type of the subject parameter have been inserted to ABox with respect to the attached certificates.
2. **History Constraint:** The *instance checking* inference service returns true, if the history constraint in the definition of $(\text{policy-rule})_i$ is satisfied w.r.t. the facts in ABox and parameters of the request.

Using the above procedure, each access request can be checked against each policy rule. The algorithm is terminated in one of the two following scenarios:

- An applicable security policy rule to the access request is found; the request is granted by the access control system.
- None of the security policy rules has the required conditions for granting the request; the request is rejected by the system.

4 Complexity of the Access Control Algorithm

Full $\mathcal{DLR}_{US}$, even restricted to atomic formulas, turns out to be undecidable [7]. $\mathcal{DLR}_{US}{}^{-}$ is a decidable fragment of $\mathcal{DLR}_{US}$, in which the temporal operators are applied only to the concepts and formulas. In other words, in this decidable fragment, the temporal operators can not be applied to n-ary relations (n$\geqslant$2). The computational behavior of $\mathcal{DLR}_{US}{}^{-}$ can be summarized as follows [7]:

- The logical implication problem in $\mathcal{DLR}_{US}{}^{-}$, when restricted to atomic formulas, is EXPTIME-complete.
- The formula satisfiability problem (and the logical implication) in $\mathcal{DLR}_{US}{}^{-}$ is 2EXSPACE-complete.
- The problem of query containment for non-recursive Datalog queries in $\mathcal{DLR}_{US}{}^{-}$ is decidable in 2EXPTIME and is EXSPACE-hard.

As mentioned before, policy rules are expressed as logical relations in TDLBAC-2. The only pattern that can not be stated under $\mathcal{DLR}_{US}{}^{-}$ constraints is pattern 2, since the temporal operator $\Diamond^{-}$ is applied to the ternary relation in this pattern. Therefore, we study the complexity of the access control algorithm without considering pattern 2. Note that undecidability of the underlying logic should not be considered as a great weakness for pattern 2. With further research in this area, decidable algorithms might be found for a larger fragment of $\mathcal{DLR}_{US}$ that includes such formulae.

The algorithm of access control in TDLBAC-2 in the worst case will include an *instance checking* inference service for each security rule. The *instance checking* inference service can be expressed as the problem of answering (boolean) conjunctive queries [10]. It is proved in [11] that query answering under DL constraints can be reduced to query containment. On the other hand, The problem of query containment for non-recursive Datalog queries in $\mathcal{DLR}_{US}{}^{-}$ is decidable in 2EXP-TIME and is EXSPACE-hard [7]. Since the problem of query answering is reducible to the problem of query containment, 2EXP-TIME can be considered as the upper bound for this problem. In the access control procedure, the service is performed for all the security rules in the worst case. Since there is a limited number of security rules, the upper bound of complexity of the algorithm will remain 2EXP-TIME in the worst case.

Note that we investigated the upper bound of the algorithm based on the known complexity of inference services. The point of this section is proof of the algorithm decidability. In future, efficient algorithms and reasoners might be developed for the logic that can lead to an efficient implementation of TDLBAC-2.

5 Evaluation and Comparison

TDLBAC [3] and TDLBAC-2 are two access control models based on a temporal extension of DLs for expressing the history constrained security rules in Semantic Web. In this section, we compare these two models from different points of view.

5.1 The Applied Logic

The logic used for stating the security policy rules in TDLBAC is $\mathcal{TL}$-$\mathcal{ALCF}$, while TDLBAC-2 uses $\mathcal{DLR}_{US}$. $\mathcal{TL}$-$\mathcal{ALCF}$ is an interval-based temporal extension of the DL $\mathcal{ALCF}$, while $\mathcal{DLR}_{US}$ is a point-based temporal extension of the DL $\mathcal{DLR}$. One of the advantages of $\mathcal{DLR}_{US}$ over $\mathcal{TL}$-$\mathcal{ALCF}$ is its ability to state n-ary relations (n$\geqslant$2). n-ary relations (n$\geqslant$2) in $\mathcal{DLR}_{US}$ allows us to define a 3-ary relation for each access and policy rule, instead of defining a concept and three binary relations as in TDLBAC. Similarly, to store the history of accesses in HB, one statement is enough for each access. However, in TDLBAC four statements are registered for each access. It is obvious that stating security rules and accesses is easier in TDLBAC-2. Moreover, the size of HB in TDLBAC-2 is smaller in comparison to TDLBAC in similar conditions. This can reduce the problem of size for large history bases. Moreover, it can help the access control procedure to be more efficient.

5.2 Expressiveness

To compare TDLBAC and TDLBAC-2 from the expressiveness point of view, we take a look at the ability of TDLBAC to state the patterns used in TDLBAC-2.

TDLBAC is able to state policy rules in pattern 1. However, it is not able to state policy rules of pattern 2, since there is no operator for stating the cardinality in the syntax of $\mathcal{TL}$-$\mathcal{ALCF}$. In TDLBAC, it is not also possible to state policy rules of patterns 3,4, and 5. The reason is the interval based nature of the temporal part of $\mathcal{TL}$-$\mathcal{ALCF}$. In TDLBAC, for each access, a separate concept should be defined, and it does not propose a method for decreasing the ontology size like meta pattern 6. Application of meta pattern 7 is not also possible using the $\neg$ operator in TDLBAC.

6 Related Work

Logical ideas and tools have been used since late 90's by many researchers to state different kinds of security rules. Kolaczek in [4] employs Deontic Logic to propose a new formal model for role-based access control. Deontic Logic is used as a language for specification of security policies, which helps to automate the implementation of policies. Chae in [5] uses a a modal logic to propose a a kind of role-based access control model. The novelty of this model is to perform authorization at the level of classes, instead of individual objects, which leads to more flexibility in security management.

The other related works include the works for expressing the security rules based on the past events. Among those, Chinese Wall Security Policy in [12] and Deeds system in [13] can be mentioned. These works focus on collecting a selective history of sensitive access requests and using this information to constrain future access requests. Abadi in [14] uses a different approach which considers the history of control transfers, rather than a history of sensitive requests. None of these works considers access control in a semantic-based environment, as well as conceptual-level policy specification and inference. Bertino in [15] presents a language to express both static and dynamic authorization constraints as clauses in a logic program. This work considers the execution history of the workflow for workflow management systems. Our schema differs from the above approach in employing a temporal description logic to propose a model suitable for Semantic Web. We also provide seven patterns for expressing different kinds of history-based policy rules.

With the growth of Semantic Web, access control for these environments have been extensively investigated. An access control model for a semantic-aware environment should consider the defined semantic relationships among entities to infer the authorized accesses from the defined policy rules. Several efforts as in [1] have been put into designing access control models for semantic-aware environments such as Semantic Web. The important novelty of the proposed model in [1], named SBAC, is reducing semantic relationships (defined as ontologies in three domains of access control) to subsumption. This approach reduces the temporal and spatial complexities, and simplifies the authorization propagation. TSBAC [2] is an extension of the SBAC model [1], which allows expressing the policies with the constraints on history of users' accesses. TDLBAC is proposed in [3] to remove two weaknesses of TSBAC; Restriction to the security policies in the level of individuals, and Lack of proof theory in the formal language.

TDLBAC tries to improve TSBAC with a logical foundation to take advantage of a logic-based model. In this model, it is possible to state the history-constrained policy rules in conceptual (ontology) level. TDLBAC is restricted to a special pattern for stating the policy rules. In this paper, we tried to improve this work using another temporal extension of DLs for our proposed model.

7 Conclusions

In this paper, we proposed an access control model based on a temporal description logic, $\mathcal{DLR}_{US}$, for Semantic Web. The main characteristic of the proposed model is its ability to express historical constraints in policy specification in conceptual or ontology level. To show how the model can be used in practice, seven usage patterns are introduced in the paper. The expressive power of the proposed model enables it to be used in different applications such as banking and insurance systems, elections systems, and education systems. The examples of such usages are presented in this paper after introducing each usage pattern. The inference ability, which is embedded in the access control procedure of the proposed model (using the inference services of the underlying logic), enables it to infer implicit security policies from the explicit ones based on the defined semantic relationships in the conceptual abstract layer of Semantic Web.

References

1. Javanmardi, S., Amini, M., Jalili, R., GanjiSaffar, Y.: SBAC: A Semantic–Based Access Control Model. In: Proceedings of the 11th Nordic Workshop on Secure IT-Systems, NordSec2006, Linkping, Sweden:[sn], pp. 157–168 (2006)
2. Ravari, A.N., Amini, M., Jalili, R.: A Semantic Aware Access Control Model with Real Time Constraints on History of Accesses. In: International Multiconference on Computer Science and Information Technology, pp. 827–836 (2008)
3. Faghih, F., Amini, M., Jalili, R.: A Temporal Description Logic Based Access Control Model for Expressing History Constrained Policies in Semantic Web. In: Proceedings of the 2009 IEEE International Symposium on Policies for Distributed Systems and Networks, pp. 142–149. IEEE Computer Society, Los Alamitos (2009)
4. Kołaczek, G.: Application of Deontic Logic in Role–Based Access Control. Int. J. Appl. Math. Comput. Sci. 12(2), 269–275 (2002)
5. Chae, J.: Towards Modal Logic Formalization of Role-Based Access Control with Object Classes. In: Derrick, J., Vain, J. (eds.) FORTE 2007. LNCS, vol. 4574, p. 97. Springer, Heidelberg (2007)
6. Baader, F., Horrocks, I., Sattler, U.: Description logics as ontology languages for the semantic web. LNCS (LNAI), pp. 228–248. Springer, Heidelberg (2005)
7. Artale, A., Franconi, E., Wolter, F., Zakharyaschev, M.: A temporal description logic for reasoning over conceptual schemas and queries. LNCS, pp. 98–110. Springer, Heidelberg (2002)
8. Artale, A., Franconi, E., Mosurovic, M., Wolter, F., Zakharyaschev, M.: The DLRUS temporal description logic. In: Proceedings of the 2001 Description Logic Workshop (DL 2001), Citeseer, pp. 96–105 (2001)
9. Baader, F., Calvanese, D., McGuinness, D.L., Patel-Schneider, P., Nardi, D.: The description logic handbook: theory, implementation, and applications. Cambridge Univ. Pr., Cambridge (2003)
10. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. In: Proc. of the 10th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR 2006), pp. 260–270 (2006)
11. Calvanese, D., De Giacomo, G., Lenzerini, M.: Conjunctive query containment and answering under description logic constraints. ACM Transactions on Computational Logic (TOCL) 9(3), 22 (2008)
12. Brewer, D.F.C., Nash, M.J.: The Chinese wall security policy. In: Proceedings of the 1989 IEEE Symposium on Security and Privacy, Citeseer, pp. 206–214 (1989)
13. Edjlali, G., Acharya, A., Chaudhary, V.: History-based access control for mobile code. In: Proceedings of the 5th ACM Conference on Computer and Communications Security, pp. 38–48. ACM, New York (1998)
14. Abadi, M., Fournet, C.: Access control based on execution history. In: Proceedings of the 10th Annual Network and Distributed System Security Symposium, Citeseer, pp. 107–121 (2003)
15. Bertino, E., Ferrari, E., Atluri, V.: The specification and enforcement of authorization constraints in workflow management systems. ACM Transactions on Information and System Security (TISSEC) 2(1), 104 (1999)