A Graph-Based Formalism for Controlling Access to a Digital Library Ontology

Subhasis Dasgupta, Aditya Bagchi

A Graph-Based Formalism for Controlling Access to a Digital Library Ontology

Subhasis Dasgupta[1] and Aditya Bagchi[2]

[1] Indian Statistical Institute, 203 B T Road, Kolkata 700108, India.
<dasgupta.subhasis@gmail.com>
[2] Indian Statistical Institute, 203 B T Road, Kolkata 700108, India.
<aditya@isical.ac.in>

Abstract. This paper presents a graph-based formalism for an Ontology Based Access Control (OBAC) system applied to Digital Library (DL) ontology. It uses graph transformations, a graphical specification technique based on a generalization of classical string grammars to nonlinear structures. The proposed formalism provides an executable specification that exploits existing tools of graph grammar to verify the properties of a graph-based access control mechanism applicable to a digital library ontology description. It also provides a uniform specification for controlling access not only at the concept level but also at the level of the documents covered by the concepts including node obfuscation, if required.

Keywords: Ontology, Digital Library, Multiple Inheritance, OBAC

1 Introduction

Recent study on the modeling of digital library (DL) suggests an ontological structure for its representation [1], where documents may be classified and stored against appropriate concepts present in the ontology. Such DL ontology usually considers an underlying tree structure for its implementation, i.e. one concept can have only one parent concept [2][3]. However in real life situation, a particular concept may have more than one parent concepts. For example, a concept named Database may be reached from Computer Science & Engineering (CS), Geographic Information System (GIS) or Biology/Bio-informatics (BIO). This consideration changes the underlying structure of the ontology from a tree to a Directed Acyclic Graph (DAG).

Now, the three parent concepts of Database may have distinct or even overlapping user communities. As a result, any document under Database may be of interest to more than one of the above three user communities. Research work to control access to a digital library, done so far, ensures that a user must possess appropriate authorization to get access to a concept. However, if access to a concept is granted, all documents under it are available to the concerned user. Some work has already been done to control access even at the document level. In other words, a user getting access to a concept may not get access to all the

documents covered by that concept, particularly in a situation when a concept has multiple parent concepts. This gives rise to a flexible access control system for a DL environment hitherto unexplored [4]. This earlier work considered some implementation issues related to such access control environment. Present paper, however, offers a graph-based formalism for an Ontology Based Access Control (OBAC) system applicable to Digital Library (DL) ontology. It uses graph transformations, a graphical specification technique based on a generalization of classical string grammars to nonlinear structures [5][6]. The proposed formalism is useful for the following reasons:

- To specify the properties of a proposed Access Control specification for Digital Library Ontology
- To provide an executable specification that exploits existing tools to verify the properties of a graph-based access control mechanism applicable to a digital library ontology description.
- To provide a uniform specification for controlling access not only at the concept level but also at the level of the documents covered by the concepts including node obfuscation.

Similar attempt has already been taken to model Discretionary, Lattice based and Role Based Access Control system using similar graph-based formalism [7][8]. However, formalism proposed in this paper has considered a single user environment. Role/User Group and any possible conflict arising out of them will be studied later.

While Section 1 provides the introduction, Section 2 covers the technological preliminaries. Section 3 provides the graph model of concept hierarchy and Section 4 gives the fundamentals of graph transformation and security model. Section 5 covers the policy algebra. Section 6 draws the conclusion.

2 Technological Preliminaries

This paper proposes a flexible access control system for retrieving documents using a digital library ontology supporting an underlying DAG structure to handle multiple parent concept problem. So here a concept may have more than one parent concept in the hierarchy. As a result, the documents under a concept can be categorized against the concepts above it. A user can access a document only if he/she has appropriate authorization to access the category to which the document is placed. Figure.1 shows an environment where documents covered under the concept Database may be contributed by or of interest to any users of the parent concepts. So a document under a child concept can be a member of one or more than one of the parent concepts. Consequently, documents under a child concept having n parents, can be classified into $(2^n - 1)$ categories. So, the Database concept in Figure.1 can be classified into $(2^3 - 1)$ or 7 categories. Figure.2 shows the Venn diagram corresponding to the concept Database having three parent concepts Computer Science (CS), Geographic Information System (GIS) and Bio-Informatics (BIO) as explained earlier. So, a document under the

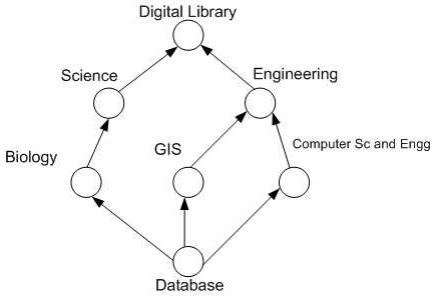

Fig. 1. An ontological structure with the concept Database having three parent concepts Biology, GIS and Computer Sc and Engg.

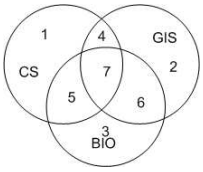

Fig. 2. Possible document categories under the common concept "DATABASE"

concept Database may be of interest to the users of CS/GIS/BIO or any combinations of them. Situation depicted in Figure.1 and Figure.2 is very common in case of a digital library. However, the present implementations avoid such document classification possibility and keep the child concept under the parent concept that contributes maximum number of documents. So, according to the above example, the concept Database would possibly be kept in the CS path with all documents under it. Any user of GIS or BIO community will directly access the Database concept and would be able to get all the documents under it. However, the proposed system provides $(2^3 - 1) = 7$ document classes as shown in Figure. 3. Depending on the parent concept where a user is authorized, corresponding document classes under Database concept and hence the documents covered by them can be accessed. An ontology based system has been discussed in many documents [9][10][11]. This section describes the related technologies.

- **Ontology:** The fundamental objective of Ontology is to form a semantic network based system for organizing concepts in a directed graph structure and to provide a mechanism to search a concept from such a structure by which a given schema element is referred. It also finds other related elements/concepts in the ontology. Ontology can be defined by a directed graph, where each node is a concept. If O is an ontology represented as $O = (C, L)$ then C is a concept, and L is the link between two concepts representing their semantic relationship.
- **Properties:** Each ontology has a set of properties, classified into either object property or data property. Data property describes about the data

and Object property deals with the concepts. All domain property of a concept c can be represented as $P(c) = DP(c) \bigcup OP(c)$ [10], where $DP(c)$ is the data property and $OP(c)$ is the concept property. The proposed system has used two types of links between concepts: $isSubClassOf$ and $hasContributedTo$ to represent the relations among concepts. These are object properties. In Figure. 1 , $Biology$ $(isSubClassOf)$ $Science$, so the $OP(C_{Biology}.(isSubClassOf)) \in \{Science\}$.

- **Concept:** Each ontology has a set of semantically related concepts, $C(o) = \{c_1, c_2, ..., c_n\}$. Fig. 1 is showing a Digital Library(DL) ontology, hence $C(DL) = \{c_{database}, c_{biology}, c_{computerSCandEngg}, ..., c_{DigitalLibrary}\}$.
- **Concept Hierarchy:** Ontology structure is a DAG, where a concept may have more than one parent concepts. Concepts in an ontology o can be represented as a partial order set. Given two concept $(Science, Biology) \in (DigitalLibrary)$ where $isSubClassOf(Biology) = Science$, i.e. Biology is more specialized than Science. It can also be denoted as $C_{Biology} \prec C_{Science}$.
- **Document Class:** Documents in a concept are classified into several classes depending upon the number of first degree parents. If a concept has n number of parents then the concept should have $(2^n - 1)$ number of document class. In Figure. 1 database has three parent concepts. So the documents covered by the concept database has 7 possible document classes.
- **Document Anotation:** Gonçalves et. al. has published some work on ontological representation of digital library. Concept of an ontology can be identified by it's URI. In the present system, a document is identified by its concept URI with a unique document-id suffixed.

3 Graph Model of Concept Hierarchy

A Graph model has been adopted in this paper to represent the DL Ontology. In an ontology, each node represents a concept and each link running from one concept to another represents the relationship between the concerned concepts. Present research effort has considered two types of relationship:

1. **isSubclassOf** : *isSubClassOf* relationship represents a partial ordered set. In the graph model, *isSubClassOf* represents the parent-child relationship. In Figure.1, Biology *(isSubClassOf)* Science i.e. in the graph model Biology will be a child concept of Science. *isSubClassOf* is a non-commutative property.
2. **hasContributedTo** : Multiple parent relationship has been represented through *hasContributedTo* relationship. In Figure.1 *Database* has been contributed by three parent concepts *Biology* , *GIS* and *Computer Sc and Engg.* This relationship has been represented as *hasContributedTo* in the Graph Model. Biology *hasContributedTo* Database. This relationship is also non-commutative.

3.1 Multiple Parent Representation

As explained earlier, if n number of parent concepts contribute to a child concept then there would be $(2^n - 1)$ number of document classes, i.e. if a node has

three parent concepts then there would be 7 possible classification of documents. This relationship is also represented by *hasContributedTo* relation. Figure.3 illustrates the relationship for Database Concept.

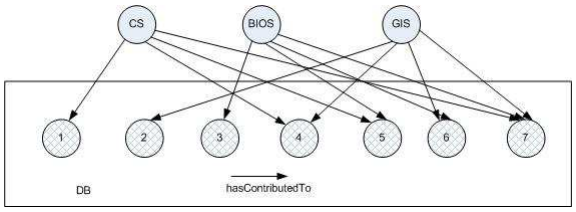

Fig. 3. *hasContributedTo* Relationship for all the Classification

Table 1. Document Class

Document Class	Access
1	CS
2	GIS
3	BIO
4	CS , GIS
5	CS , BIO
6	BIO, GIS
7	CS, BIO, GIS

3.2 Entities of OBAC Model

OBAC can be represented by a set of Subject (S), Object (O), Access Rights (A) and Sign (V). Depending upon the context and policy, subject can be users, groups, roles or applications. Object, on the other hand, can be a group of ontologies, an ontology, a concept within an ontology or a document inside a concept. Present effort has considered only read and browse operations for a digital library. Sign can either be positive or negative, signifying access permission or explicit denial respectively. OBAC model contains following entities :

1. User : User may be an individual user or a group of users. $U = \{u_i\}, i \geq 1$
2. Object : Object can be a document under a concept or a concept itself or an entire ontology or even a group of ontologies. $O = \{o_i\}, i \geq 1$
3. Subject : A Subject may refer to an individual or a user group or a role or may even be a web service.
4. Access rights : At present, the OBAC model has considered *read* and *browse* access rights only. $A \in \{read, browse\}$. A valid user can browse through all

the nodes by default but would need explicit positive authorization to read any document under a concept.

5. Sign : Sign are of two types, $ve+$ or $ve-$, positive for access permission and negative for explicit denial. $\vartheta = \{+, -\}$
6. Policy : Access control policy is a tuple (S, O, a, ϑ), where S is the subject, O is the object , a is the access rights and ϑ is the sign.

3.3 Relationship in OBAC

As mentioned earlier, ontology contains a set of related concepts and the relationship among the concepts with their data properties and object properties. Present paper considers *isSubClassOf* and *hasContributedTo* relations among the concepts.
From the inter-concept relations, following relationships can be derived in the OBAC model:

1. Inclusion($\odot$) : $(C_i \odot C_j)$ signifies that concept C_i is included in concept C_j. Inclusion relationship is non-commutative i.e. $C_i \odot C_j \neq C_j \odot C_i$. However, Inclusion relationship is transitive.
2. Inferable :($\Longrightarrow$) : If a concept C_i infers the existence of another concept C_j then C_j is inferable from C_i i.e $C_i \Longrightarrow C_j$. Inferable relationship is non-commutative , .i.e $C_i \Longrightarrow C_j \neq C_j \Longrightarrow C_i$, and transitive i.e. if $C_i \Longrightarrow C_j$ and $C_j \Longrightarrow C_k$ then $C_i \Longrightarrow C_k$.
   Since the present proposal considers only *isSubClassOf* relation between concepts and *hasContributedTo* relation between concepts and document classes, Inclusion and Inferable would virtually be similar.
3. Partially Inferable: ($\rightharpoonup$) : If C_i and C_j are two concepts, then $C_i \rightharpoonup C_j$ signifies that the concept C_i can partially infer the concept C_j. This relationship is also non-commutative and transitive. This relationship will be particularly important for concepts with multiple parents.
4. Non Inferable ($\not\Rightarrow$) : If a concept C_i cannot infer the existence of another concept C_j, the relationship is non-inferable.

3.4 Authorization

An authorization is defined by a four tuple $(\rho, \delta, \upsilon, \vartheta)$ where ρ is a member of the subject set S, δ is a member of the object set O, υ is a member of the set of access rights A available in the DL system and ϑ is either $+ve$ or $-ve$ signifying positive or negative authorization respectively. In Concept level authorization, an object O is identified by its ontology URI (with the path expression). Each concept C maintains an authorization list represented by S, A, V where a particular authorization of the form $(\rho, \upsilon, \vartheta)$ signifies that $\rho \in S, \upsilon \in A, \vartheta \in \{+, -\}$. Here, the subject ρ can access concept C with access right υ if $\vartheta = +ve$ or cannot access the same if $\vartheta = -ve$. Once again in the present paper, A is limited to read and browse only. $\vartheta = +ve$ signifies read is permitted and $\vartheta = -ve$ if read is not permitted. In the present proposal browse to any concept is permitted by default.

4 Fundamentals of Graph Morphism and Security Model

This section discusses the formal method of graph transformation, i.e. transformation steps and rules. A formal introduction to Graph based formalism can be found at [6] and a RBAC implementation of the model has been developed by Koch et. al. [7]. In very brief, a graph represents a state of a system and a rule is

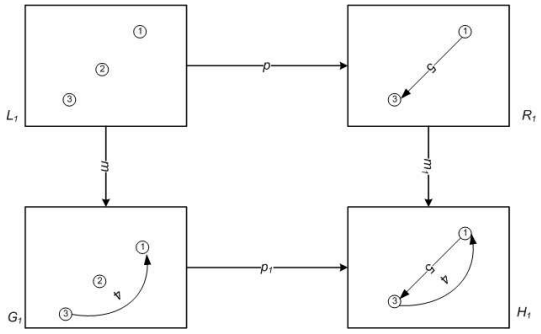

Fig. 4. Graph Morphism Basics

a transformation $\{r : L \to R\}$, where both L and R are graphs i.e. the left-hand side transforms to the right-hand side by the graph transformation rule. L is the original graph and R is the transformed graph after applying relevant access control policies. The rule, $\{r : L \to R\}$ consist an injective partial mapping form left-hand side to right-side, among the set of nodes r_n and the set of links/ relations r_e. Each mapping, should be compatible with graph structure and the type of the node. If the mapping is total the graph morphism is total. The L describe which objects a Graph G must contain for the rule $\{r : L \to R\}$ to be applicable to G. Nodes and edges of L, whose partial mapping are un-defined or restricted by the rules, will be either deleted or modified. However, nodes and edges of L, those are defined at rule will be stored at R after transformation.The basic idea of a graph transformation [6] considers a production rule $p : (L \rightsquigarrow R)$, where L and R are called the left-hand and right-hand side, respectively, as a finite schematic description of potentially infinite set of direct derivations. If the *match* m fix the occurrence of L in a given graph G, then $(G \overset{p,m}{\Longrightarrow} H)$ denotes a direct derivation where p is applied to G leading to directive graph H. The H is obtained by replacing the occurrence of L in G by R. From algebraic approaches, the graph has been considered as a collection of edges (E) and vertices (V). However, source $s : E \to V$ and target $t : E \to V$ are two unary operations. Figure 4 , shows the intermittent state of transformation, where each graph production $p : (L \rightsquigarrow R)$ defines the partial correspondence between the elements of left-hand side and the right-hand side on the basis of a rule, determining which edge should be present and which edge should be deleted. A match $m : L \to R$ for a production p is a graph homomorphism, mapping nodes and edges of L to

R in such a way that the graphical structure and the levels are preserved. Considering the scenario in Figure 4 consider the production $p_1 : (L_1 \rightsquigarrow G_1)$ which applied on the graph G_1. The number written in the vertices and edges consider the partial correspondence between $L_1 \rightarrow G_1$. Same number represent the same object before and after the transformation. The production $p_1 : (L_1 \rightsquigarrow G_1)$ gets three vertices at the left hand side L_1 and leaves behind two vertices, connected by an edge depicting a rule to permit information flow for security. The match $m_1 : (L_1 \rightarrow G_1)$ maps each element of L_1 to the element of G_1 carrying the same set of numbers. Following the other production/transformation rules in the same way, derived rule for H_1 will be $G_1 - (L_1 - R_1) \bigcup (R_1 - L_1)$.

4.1 Example of Graph Morphism

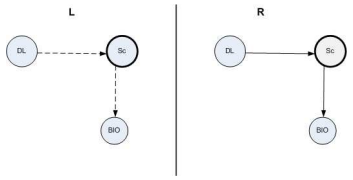

Fig. 5. Graph Transformation for Node Inference

Notion of rule and its application to a graph have been described by using three examples. Three example cases are **Node Traversal**, **Node Obfuscation** and **Partial Inferences**. Figure.5 describe the Node traversal. Let, a query wants to access a document under the concept *Bio*. However, the access of concept *Bio* can be done through *Science* node. Hence, the query will try to access the node through the concept *Science*. In the L rule hence a dashed line has been added. The dashed edge in the left side represents a *negative application condition* [7]. A *negative application condition* for the rule $\{r : L \rightarrow R\}$ is a set of (L, N) where L is the subgraph of N, and N is the structure as a whole and must occur in the original structure, i.e. the whole ontology G. Hence, in Figure. 5 query has inferred its access from the rule and right hand side has shown the access path.

Node obfuscation is a very common issue for this kind of structure both in XML and Ontological environments [12][13][14]. In the previous example, query can trace the exitance of the node *Science*. If the name of the node is sensitive for some user then system will block the identity of the node and it will be obfuscated for the concerned user. Now the query will pass through the node without accessing the node name or its structural details. Documents covered by an obfuscated node will not be available to the user as well. Figure.6 creates this scenario. In the left side the query intended its access for node *Science*, which has a *negative application condition*, and by the rule set the system has found that node *Science* should be obfuscated for this query. Hence, the rule $\{r : L \rightarrow R\}$

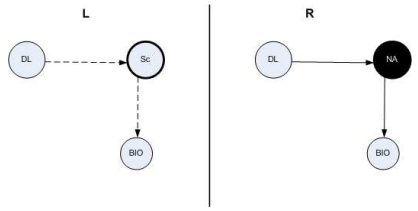

Fig. 6. Graph Transformation for Node Obfuscation

creates the new graph on the right hand side. **Partial Inferable** is another

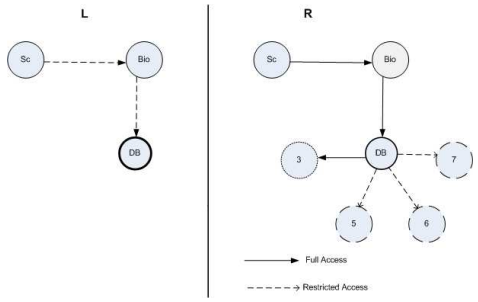

Fig. 7. Graph Transformation for Partial Inference

Table 2. Example Policy for Partial Inference

Name	Source	Joint	No Contribution
Policy A	Full	abstract	No access
Policy B	Full	No access	No access
Policy C	Full	Full	abstract

prolific problem discussed in many research papers. Mutiple parent condition has also been addressed here through graph morphism. As mentioned earlier, that documents under *database* concept have been contributed by three parent concepts and thus the documents under *database* concept can be classified into $(2^3 - 1)$ classes. Graph transformation ensures partial access to documents or access to relevant document classes only.

5 Policy Algebra

The production rules of the graph are generated by a rule engine depending upon the access control algebra. This section will define the formal algebra for

our model. Our model considers each node as a concept and the concept level authorization can be represented by S,O,A and V, where S is the concept or ontology or a group of concept from a set of ontology,concepts are related through semantic relationships, we are considering *isSubClassOf* and *hasContributedTo* relations here. Thus authorization for concept i can be represented as :

$$ca_i = (\rho_i, \delta_i, \upsilon_i, \vartheta) \tag{1}$$

Where , ρ_i δ_i, υ_i are the instances from S,O,A. We denote set of authorization explicitly specified for a concept. In addition to explicit authorizations, other authorizations can be generated through propagation. Hence, a propagation from concept i to j can generate the authorization according to the policy. Policy defines authorization of an object for a subject. Hence a simple policy can be defined as

$$E_i = (C_i, u_i, \{R\}, \{+\})$$

where, C_i is a constant of S or a variable over S, u_i is a constant of O or a variable over O, R is the access right and $+$ is the sign. Two or more policies can be joined to create combined policy. Hence, the combination of policy can be expressed as BNF grammar, i.e. a combination of policies can be written as

$$E ::= id|E + E|E\&E|E - E|E.C|o(E, E, E)|E * r|T(E)|(E)| \tag{2}$$

$$T ::= \tau id.T|\tau.E \tag{3}$$

Here , id is the token type of policy identifiers, E is the nonterminal describing policy *expression* , T is the policy template, C and r are the authorization content and authorization rules respectively. Above symbols have been disambiguated by proper precedence rules of the operators[15]. The policy can be stated explicitly or generated by some rule engine. Combination of more than one policy may be enforced over one *environment*, we will refer them as the composite policy. Using the rule , some policies can be obtained by the grammar through algebraic operators. We refer them as derived policies E_{der} which should satisfy the following rules :

1. Reflexivity : For all tuples of the E are inherited by E_{der}
2. Inheritance Rule : $(C_i, u_j, \{R\}, \{+\}) \in E) \wedge (\forall u_i \in U | u_j \xrightarrow{a} u_i) \longrightarrow (C_i, u_j, \{R\}, \{+\} \in E_{der})$
3. Override Rule : $(C_i, u_j, \{R\}, \{+\}) \in E) \wedge (C_i, u_j, \{R\}, \{-\}) \in E_{der}) \longrightarrow E_{der} \wedge (C_i, u_j, \{R\}, \{+\}) \notin E_{der}$
4. Commutative : $E_a + E_b = E_b + E_a$

5.1 Policy Construction

OBAC system contains two type of policies, **Simple Policy** and **Composite Policy**, Simple policy is the granularity of policy i.e. the mapping of rules with ground algebra, where as combinations of more than one simple policies can create a composite policy. Consider the previous example of CS, GIS, BIO

and Database(DB). We are considering that all the documents in DB has been contributed by three parent concepts. The policy of each concept has been represented by P_{CS}, P_{GIS} and P_{BIO} respectively, and those are the **Simple Policy**. On the contrary, the documents at **DB** has been annotated by either *cs.db*, *gis.db* or *bio.db*. Now the policy for accessing document (d) for **DB** will be a **Composite Policy** that can be represented as:

$$\Pi_{DB} = \begin{cases} P_{gis.db} + P_{cs.db} + P_{bio.db} + \delta_{db} & \text{if } (d \leq cs.db \bigwedge d \leq gis.db \bigwedge d \leq bio.db) \\ P_{cs.db} + P_{gis.db} + \delta_{db} & \text{if } (d \leq cs.db \bigwedge gis.db) \\ P_{bio.db} + P_{gis.db} + \delta_{db} & \text{if } (d \leq bio.db \bigwedge gis.db) \\ P_{cs.db} + P_{bio.db} + \delta_{db} & \text{if } (\ d \leq cs.db \bigwedge bio.db) \\ P_{cs.db} + \delta_{db} & \text{if } (\ d \leq cs.db\ ) \\ P_{gis.db} + \delta_{db} & \text{if } (\ d \leq gis.db\ ) \end{cases} \tag{4}$$

Where, Π_{DB} is the composite policy of **DB**. Here δ_{db} represents the concept specific constraints common for all document classes. Policy of a interrelated concepts/ontology can be represented by *Composite Policies*. If we have n number of related concept in an ontology, we can represent the policy for the total ontology as:

$$\Pi_C = \Pi_1 + \Pi_2 + ... + \Pi_k + \Pi_{k+1} + ... + \Pi_n \tag{5}$$

Where, Π_C is the policy of the ontology and Π_1, Π_2 are the policies of the respective concepts.

6 Conclusion

This paper provides a graph based formalism for controlling access to a digital library ontology. Using a graph specification technique based on classical string grammars to nonlinear structures, this paper proposes graph transformation rules to represent different access control situations. Corresponding Policy algebra and policy derivations have also been provided for clarification of the formalism. The entire study has not considered user groups/roles and any possible conflicts arising out of them. It would be part of future work.

References

1. Gonçalves, M.A., Fox, E.A., Watson, L.T.: Towards a digital library theory: a formal digital library ontology. Int. J. Digit. Libr. **8** (April 2008) 91–114

2. Adam, N.R., Atluri, V., Society, I.C., Bertino, E., Member, S., Ferrari, E.: A content-based authorization model for digital libraries. IEEE Transactions on Knowledge and Data Engineering **14** (2002) 296–315
3. Ray, I., Chakraborty, S.: A framework for flexible access control in digital library systems. In Damiani, E., Liu, P., eds.: DBSec. Volume 4127 of Lecture Notes in Computer Science., Springer (2006) 252–266
4. Dasgupta, S., Bagchi, A.: Controlled access over documents for concepts having multiple parents in a digital library ontology. In Chaki, N., Cortesi, A., eds.: Computer Information Systems Analysis and Technologies. Volume 245 of Communications in Computer and Information Science. Springer Berlin Heidelberg (2011) 277–285 10.1007/978-3-642-27245-5_33.
5. Ehrig, H., Engels, G., Kreowski, H.J., Rozenberg, G., eds.: Handbook of graph grammars and computing by graph transformation: vol. 2: applications, languages, and tools. World Scientific Publishing Co., Inc., River Edge, NJ, USA (1999)
6. Corradini, A., Montanari, U., Rossi, F., Ehrig, H., Heckel, R., Löwe, M. In: Algebraic approaches to graph transformation. Part I: basic concepts and double pushout approach. World Scientific Publishing Co., Inc., River Edge, NJ, USA (1997) 163–245
7. Koch, M., Mancini, L., Parisi-Presicce, F.: Graph-based specification of access control policies. Journal of Computer and System Sciences **71**(1) (2005) 1 – 33
8. Koch, M., Mancini, L.V., Parisi-Presicce, F.: On the specification and evolution of access control policies. In: SACMAT. (2001) 121–130
9. Kashyap, V., Sheth, A.: Semantic and schematic similarities between database objects: a context-based approach. The VLDB Journal **5** (December 1996) 276–304
10. Qin, L., Atluri, V.: Semantics aware security policy specification for the semantic web data. Int. J. Inf. Comput. Secur. **4** (February 2010) 52–75
11. Ouksel, A.M., Ahmed, I.: Ontologies are not the panacea in data integration: A flexible coordinator to mediate context construction. Distributed and Parallel Databases **7** (1999) 7–35 10.1023/A:1008626109650.
12. Damiani, E., di Vimercati, S.D.C., Fugazza, C., Samarati, P.: Modality conflicts in semantics aware access control. In Wolber, D., Calder, N., Brooks, C.H., Ginige, A., eds.: ICWE, ACM (2006) 249–256
13. Gabillon, A.: A formal access control model for xml databases. In Jonker, W., Petkovic, M., eds.: Secure Data Management. Volume 3674 of Lecture Notes in Computer Science. Springer Berlin / Heidelberg (2005) 86–103 10.1007/11552338_7.
14. Kaushik, S., Wijesekera, D., Ammann, P.: Policy-based dissemination of partial web-ontologies. In Damiani, E., Maruyama, H., eds.: Proceedings of the 2nd ACM Workshop On Secure Web Services, SWS 2005, Fairfax, VA, USA, November 11, 2005, ACM (2005) 43–52
15. Bonatti, P.A., di Vimercati, S.D.C., Samarati, P.: An algebra for composing access control policies. ACM Trans. Inf. Syst. Secur **5**(1) (2002) 1–35