# UCAT: the Uniform Categorization for Access Control

Denis Obrezkov[1][0000−0001−8822−2932] and Karsten Sohr[2][0000−0001−6781−4226]

[1] TIB Leibniz Information Centre for Science and Technology, Hannover, Germany
denis.obrezkov@tib.eu
[2] University of Bremen, Bremen, Germany
sohr@tzi.de

**Abstract.** The basic primitives of access control models evolve together with the development of new technologies. The increased availability of computers in organizations brought the notion of roles in, the growing popularity of online social networks led to access control models based on relationships. The new era introduces new challenges. One of those is related to data management in collaborative platforms. Being involved in numerous collaborations, the users need convenient solutions to express their access control preferences. In this paper we address this problem by developing an access control model based on categorization. Relying on evidence from cognitive science, we employ categories as a natural primitive for expressing users' access control preferences. We develop our model using a fragment of hybrid logic and evaluate its performance in a simulated environment.

## 1 Introduction

The evolution of access control models follows the development of technology. Earlier access control models, like Bell-LaPadula [4] or lattice-based [11], operated primarily on subjects, objects, information flows, actions — keeping a high level of generalization. The increased availability of computer systems for organizations led to the appearance of such models as Role-Based Access Control (RBAC) [13]. The latter leveraged notions of roles and users, thereby closing a gap between the model and the target domain. Lastly, the widespread use of personal technology and the Internet gave a birth to user-centric models. For instance, Relationship-Based Access Control (ReBAC) systems allow a user to specify a local policy based on their relationships with others [6].

The modern era brings new challenges. Collaborative platforms require policies that would allow users to specify their access control preferences not only with regard to other users but also with regard to data [17]. The data itself is starting to play a more prominent role in new systems. For instance, in the scientific domain the importance of data-centricity is reflected in FAIR data management principles. FAIR is defined as a set of guidelines: data should be Findable, Accessible, Interoperable, and Reusable [19]. The aim of the guidelines is to empower researchers with more rigorous management of scientific digital objects. In the context of access control, it is interesting to consider the Accessibility principle. Among other points, it is stated that "the protocol allows for an authentication and authorization procedure, where necessary". The authorization procedure itself, however, is out of scope of the guidelines.

The problem of data- and user-centric permission management is a prominent challenge for collaborative platforms [18,17]. Classic access control policies might not fit well the new environment. In the context of scholarly publishing platform, one can consider a situation when a researcher shares their digital manuscript draft with their colleagues from other institutions, with a third-party proofreader, and with a reviewer. It would be very notorious to establish the appropriate roles in an RBAC-system: the reviewer and the colleagues might have similar roles in the system. Additionally, after the positive review, it might be desired to grant the access to a broader audience, changing the type of the document from "draft" to "accepted paper". In that case, ReBAC models are also out of consideration, since they do not account for object types.

In a modern collaborative system we should allow a user to express their preferences towards resources and other people. One of the approaches is to account for the main mechanics behind access control — categorization. In case of ReBAC, a user categorizes requesters, e.g. as friends or colleagues. In the Bell-LaPadula model, an authority assigns sensitivity labels to files (for instance, "secret", "top secret"), thereby categorizing them. Given that cognitive science considers categorization to be the basic mechanism behind the perception of both objects [12] and people [2], it seems desirable to develop a model that would combine a user-centered view of ReBAC models with a uniform and deliberate application of categorization.

In this paper, we propose a user-centric model that accounts for both types of relationships: with subject (people, applications) and objects (data resources, files). In the next section, we review some established ways of dealing with categories and relations in access control. In section 3 we introduce hybrid logic, the formalism that we base our model on. In section 4, we describe our model. Subsequently, in section 5 we show examples of how our model can be used from both user- and platform- centric perspectives, and in section 6 we evaluate the performance of our model in an artificial collaborative environment. Lastly, in section 7 we discuss the implications of our contribution and the interesting directions for future work.

## 2 Related work

In this section, we review some of the prominent approaches in access control. We selected these works since they represent different ways of accounting for relationships between subjects and objects.

The need to reflect an organizational structure in a company's data access policies led to the appearance of the Role-Based Access Control model, or RBAC [13]. RBAC allows an organization to establish a mapping between a role of an employee and a set of desired permissions. Thereby, for instance, an *accountant* might be allowed to access only financial reports, while a *developer* might be granted access to code repositories.

Social networks brought new challenges to the field. Fong proposed to consider users' relationships as a basis for access control decisions [14]. The author leverages modal logic as a basis for his policy language. Using modal operators of necessity (`[i]`) and possibility (`<i>`), he allows such formulas as: $\langle spouse \rangle \mathtt{a}$ ("grant access to the owner's spouse."), and $\langle -parent \rangle \mathtt{a} \wedge [-parent]\,\mathtt{a}$ ("grant access if accessor is the only child of the owner"). Although this formalism accounts for interpersonal relationships, it still lacks a personal aspect: it is hard to make a decision from a certain user's perspective.

To address the limitations of the previously introduced modal logic in access control, Bruns et al. proposed to leverage hybrid logic [6]. Hybrid logic can be viewed as an extension of modal logic with enumerated graph nodes (see section 3 for more details). It allows one to evaluate formulas from a certain point of view. To achieve this, two operators are introduced: $@_i$ and $\downarrow x$. The first operator jumps to a node, named by $i$. The second, binding operator, binds the current node to the variable $x$. In their approach, the authors propose to evaluate formulas of a type $@_{own}\phi \wedge @_{req}\psi$. Thereby, the policy is evaluated from two perspectives: one of the owner and one of the requester. The safety problem is considered in the view of local model checking. It should be noted that the proposed system has a strong flavor of user-centricity, even though it is a formal access control model.

The need to account for user-object relations is partially addressed in the work by Damen et al [10]. The authors introduce CollAC, the framework that extends the aforementioned work of Bruns et al. It is proposed to use a notion of archetype that accounts for the relations between users and files. The authors avoid the inclusion of files or objects into the access control model. Instead, they define additional roles for users, such as Data Host or Data Provider. These roles are introduced as free variables. In our view, this solution might lack flexibility in scenarios where a large amount of relationship types exist between users and objects.

The work of Crampton and Sellwood introduces a new framework based on path conditions [9]. The latter are defined as chains of relationships that make it possible to identify, whether a particular principal is associated with a request. The authors also propose to use a two-step approach: first, to determine relevant principals, second, to look for a corresponding authorization policy for the matched principals. Unlike classical ReBAC model, the proposed framework supports general-purpose computing systems. At the same time, the path conditions are similar to models, based on modal logic: the latter do not provide local operators, such as $@_n$, for expressing policies that account for a certain viewpoint.

The idea of considering object-to-object relations is realized in several frameworks. Carminati et al. propose to use semantic technologies for access control [7,8]. The authors utilize modeling languages to encode security rules, they also employ reasoners to inference new relationships between entities. Ahmed et al. introduces OOReBAC, a framework that accounts for object-to-object relationships [1]. It should be noted that OOReBAC misses some important features of ReBAC, for instance, it considers only symmetric relationships. Thereby, it is not possible to account for a specific user's perspective (e.g. "Alice thinks that Bob is her friend, Bob considers Alice to be an acquaintance").

The presented works reveal several desirable features. First, there is a need for a flexible model that would be able to account for the underlying structure of an organization or community. Second, it is possible to consider different relationships in the model: user-to-object, user-to-user, object-to-object. Last, it might be desirable to have an ability of reasoning from a certain user perspective, thereby making access control user-centric.

## 3 Hybrid Logic

The need to account for different types of relationships together with a desired ability to incorporate user-centric perspectives shapes our requirements for the underlying mathematical foundation. It is not sufficient for us to rely on the apparatus of modal or description logic as done in the earlier-mentioned works (i.e. [14,7]). These systems do not provide easy solutions for making policies, accounting for a certain viewpoint. For similar reasons, we cannot employ the framework of path conditions, as in [9]. Given this, we base our framework on hybrid logic [5]. The latter is shown to be useful for accounting for a certain user's preferences [6,10].

Informally speaking, a model based on hybrid logic can be viewed as a graph. The nodes of the graph represent entities, with the edges reflecting relations between them. Hybrid logic allows one to build formulas using modalities of necessity and possibility. The intuition behind these two terms is the following. Let us consider a node $w$, connected with zero or more nodes via relation $r$. We say that $\langle r\rangle\phi$ ('possibly $\phi$ via $r$') is true at $w$, if $\phi$ is true at some nodes accessible via relation $r$ from $w$. We say $[r]\phi$ ('necessarily $\phi$ via $r$'), if $\phi$ is true at all nodes accessible vie relation $r$ from $w$. Thereby, the meanings of necessity and possibility are similar to those in modal logic. In addition to the modalities, hybrid logic introduces the mechanism of nodes enumeration. In this case each node is assigned a unique name, thereby, one can refer to a specific node in a formula. To achieve this, two operators are introduced: @ and $\downarrow$. The first one, $@_s$ ('at s') shifts the evaluation point to the node, named $s$. The binding operator $\downarrow x$ allows us to bind variable $x$ to the current point of evaluation, thereby, enabling us to refer to the current point later in the formula.

**Definition 1.** *Given the set `NOM` of nominal symbols, the set `REL` of relational symbols, the set `VAR` of propositional symbols, and the set `VAR` of variables, we can define a well-formed formula of the hybrid language `HL(@, ↓)` using the following recursive definition:*

$$\phi ::= \top \mid p \mid s \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \langle r\rangle\phi \mid \langle -r\rangle\phi \mid @_s\phi \mid \downarrow x\phi$$

*where $p \in$ `PROP`, $s \in$ `NOM`$\cup$ `VAR`, $x \in$ `VAR`, $r \in$ `REL`, and $\phi_1, \phi_2$ are formulas.*

**Definition 2.** *A hybrid model $M$ is defined as a triple $M = (W, \Omega, V)$, where $W$ is a non-empty set of nodes, $\Omega(r)$ is a binary relation on $W$ for all $r \in R$, and $V$ is a total function, such that $V : PROP \cup NOM \to 2^W$, and $V(n)$ is a singleton for all $n \in NOM$. The satisfaction relation for $M, w, g \models \phi$ is defined in fig. 1.*

$$\begin{array}{rcl}
M, w, g \models \top & & \text{always} \\
M, w, g \models x & \text{iff} & w = g(x) \\
M, w, g \models n & \text{iff} & V(n) = w \\
M, w, g \models p & \text{iff} & w \in V(p) \\
M, w, g \models \neg\phi & \text{iff} & M, w, g \not\models \phi \\
M, w, g \models \phi_1 \wedge \phi_2 & \text{iff} & M, w, g \models \phi_1 \text{ and } M, w, g \models \phi_2 \\
M, w, g \models \langle r \rangle \phi & \text{iff} & M, w', g \models \phi \text{ for some } (w, w') \in \Omega(r) \\
M, w, g \models \langle -r \rangle \phi & \text{iff} & M, w', g \models \phi \text{ for some } (w', w) \in \Omega(r) \\
M, w, g \models @_n \phi & \text{iff} & M, w^*, g \models \phi \text{ where } V(n) = w^* \\
M, w, g \models @_x \phi & \text{iff} & M, g(x), g \models \phi \\
M, w, g \models \downarrow x \phi & \text{iff} & M, w, g[x \mapsto w] \models \phi
\end{array}$$

Fig. 1: Definition of satisfaction relation $M, w, g \models \phi$. The formula $\phi$ of hybrid logic HL is true in node $w$ of model $M$ under valuation $g$.

The intuition behind the model is the following. The function $V(p)$ maps a propositional symbol to the set of nodes where the latter is true. For a nominal symbol it returns a node, to which this nominal is assigned. As we discussed earlier, hybrid logic allows a user to enumerate nodes, thereby $V(n)$ maps a nominal to a node, 'named' by this nominal. Lastly, the valuation function $g$ is a mapping from variables to nodes.

## 4 UCAT: uniform categorization for access control

The traditional ReBAC model accounts only for the viewpoints of an object owner and a requester. In other words, a policy is based on how the owner 'treats' the requester. Given the evidence from cognitive science that people apply categorization to both persons and objects [12,2], it seems natural to include 'relationships' to objects into the decision making as well. In our model we enable users to specify how they categorize other users and their own objects and to create policies based on these categories. Thereby, the following statement becomes possible: "grant access to my *private* files to my *family members*." In line with our main principle, we call our access control model Uniform Categorization, or UCAT.

**Definition 3.** *Let* `HL(own,req,dobj)` *be the set of formulas of* `HL`*, where:*

- `own,req,dobj` *are the only free variables,*
- *formulas are built using Boolean combinations of formulas of the form* $@_{own}$*,* $@_{req}$*, and* $@_{dobj}$*.*

Similar to the work of Bruns et al. [6] we refer to formulas of `HL(own,req,dobj)` as policies. Given $\circ = \{\wedge, \vee\}$, general policy pattern has the following form:

$$@_{own}\phi_1 \circ @_{req}\phi_2 \circ @_{dobj}\phi_3$$

The leftmost part $@_{own}\phi_1$ specifies a viewpoint of a resource owner. The middle statement $@_{req}\phi_2$ represents a perspective of a requester. The rightmost part of the formula allows one to make policies that account for relationships between objects. That might be useful when objects' interconnections can be utilized for security decisions. For example, it is natural to grant access to a document and also to new versions of the document. In other words, the relationship 'new-version' might be used in an access control policy.

*Authorization decisions.* Each access requests is evaluated according to the following rule. Given a model $M$, a data object $d$ in $Obj$, an object owner $o$, an access requester $r$, and a corresponding formula $pol$, the following condition should hold:

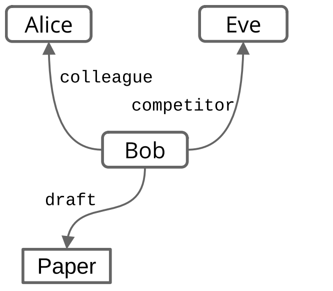
$$M, [own \mapsto o, req \mapsto r, dobj \mapsto d] \models pol \tag{1}$$

This statement binds free variables `own,req,dobj` to an object owner, a requester, a requested object and then evaluates the formula *pol*. Thereby, each authorization decision is done be evaluating a respective formula. For the evaluation one can use a local model-checking algorithm provided in the ReBAC paper [6]. The only change required for the algorithm is to include the variable `dobj` into the procedure of free variable binding.

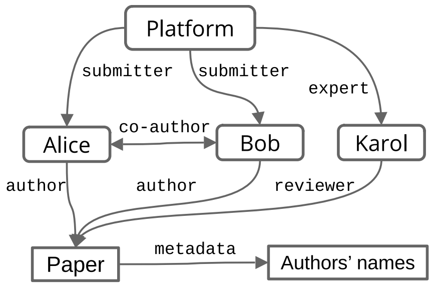
## 5 Use cases

In this section we demonstrate the applicability of our model from three different perspectives. First, we demonstrate how a user can specify their policies, considering their categorization of files and other people. Second, we provide examples, representing a traditional single authority use-case. Lastly, we demonstrate policies, adopting a data-centric perspective.

**Example 1. A user-centric perspective.** In the user-centric scenario a Bob wants to allow his colleagues to access his paper drafts. As it is shown in fig. 2a, Bob has already described Alice as a colleague, Eve as a competitor, and a certain paper as a draft. The possible policy to achieve Bob's goal is $@_{\mathtt{own}}\langle colleague\rangle\mathtt{req}$. It can be translated as "a requester should be accessible via 'colleague' relation". Similar to ReBAC, policies are associated with objects. UCAT, however, allows one to specify policies of higher granularity. Consider this formula: $@_{own}\langle colleague\rangle\mathtt{req} \wedge @_{\mathtt{own}}\langle draft\rangle\mathtt{dobj}$. This policy allows Bob to



(a) User-centric view. The graph represents how Bob categorizes Alice and Eve, and his paper



(b) Platform-centric view. Platform assigns roles to its users and to their relations with uploaded documents and other users.

Fig. 2: Two versions of an access control graph.

state that access is granted to all his colleagues to files that he considers as drafts. Together with attaching a policy to an object, UCAT leverages the hybrid model binding mechanism to map an object to the free variable `dobj`.

**Example 2. A single authority perspective.** Our platform-centric view represents a traditional model with a single authority (see fig. 2b). It is interesting to note that this use case is just a variation of the previous one: Platform can be treated as a common user. Several rules can be demonstrated. For instance, let us consider a use case of a scientific publishing system. It is desirable for a platform to allow authors to access metadata of their papers. There are at least two ways to achieve this goal. First, it is possible to make a policy from a requester's point of view: $@_{\texttt{req}}\langle author\rangle\langle metadata\rangle\texttt{dobj}$. With this formula we state that the requester should be an author of something that is related with the 'metadata' relation to the object. To allow experts of the platform to access any paper, we can specify the following policy: $@_{\texttt{own}}\langle expert\rangle\texttt{req}$. Since we employ a single-authority perspective, the `own` variable is bound to 'Platform' (e.g. to a platform's administrator). Thereby the last policy can be interpreted as "grant access to those experts, who are considered experts by the platform".

The user-centric and platform-centric perspectives can be effectively combined. A platform might have default relations to its users, e.g. categorizing them as submitters or experts. The system can automatically assign the 'co-author' label to the relation between authors of one paper. A default platform-wide policy might allow co-authors to see all the available works of each other. On top of that, a single user can assign their own labels to their co-authors for a more fine-grained access control. The latter might be required if the user involved in several projects with conflicts of interest among co-authors.

**Example 3. A data-centric perspective.** In addition to user- and platform-centric views, our model allows for shifting the decision-evaluation point to an object. For instance, we can reformulate a policy that allows authors to access their data: $@_{\texttt{dobj}}\langle -author\rangle\texttt{req}$. In that case access is granted, if the requested object is connected with a reverse *author* relation with the requester. We can also consider relations between objects. For example, to grant access to the paper's metadata to the paper's authors, we can construct the following policy: $@_{\texttt{dobj}}\langle -metadata\rangle\langle -author\rangle\texttt{req}$. It is also possible to grant access to papers with any metadata: $@_{\texttt{dobj}}\langle metadata\rangle\top$. Object-to-objects permissions are specifically useful when objects have some explicit relation, for instance, representing different versions of a single documents.

## 6 Performance evaluation

In order to assess the feasibility of our model, we performed an evaluation of several policies. We constructed an artificial environment that resembles a platform for a scientific publishing system. The system exhibits the following structure (see fig. 2b). Users can be registered as submitters or experts; they might be related with a 'co-author' relation. Some users are authors of papers, and some are reviewers. Each paper is related to its authors' names with the 'metadata' relation.

The construction of the graph, representing the aforementioned system, included the following steps. First, we leveraged the Arxiv General Relativity and Quantum Cosmology (GR-QC) collaboration network dataset from the SNAP project [16]. The dataset describes co-authorship relationships based on e-print arXiv papers in the GR-QC category. In our graph we included these relations from the dataset by specifying symmetric edge with a 'co-author' label. Second, we introduced an artificial node 'Platform' and connected it to all author nodes, randomly assigning the edges either 'submitter' or 'expert' labels. Third, to each submitter we attached 10 papers, connecting them with an 'author'-labeled edge. For each paper we randomly chose one additional author (among the co-authors of the original author) and two reviewers, randomly choosing them among experts. Lastly, each paper was associated with a node, representing its authors' names. The edges between these two types of nodes were assigned 'metadata' labels. The constructed graph aims to represent a use-case scenario of a scientific event with proceedings.

We evaluated the following policies:

$$@_{\texttt{own}}\langle\text{co-author}\rangle\texttt{req} \tag{Policy 1}$$

$$@_{\texttt{req}}\langle\text{author}\rangle\texttt{dobj} \vee @_{\texttt{own}}\langle\text{expert}\rangle\texttt{req} \tag{Policy 2}$$

$$@_{\texttt{dobj}}\langle\text{-metadata}\rangle\langle\text{-author}\rangle\langle\text{co-author}\rangle\texttt{req} \tag{Policy 3}$$

$$@_{\texttt{req}}\langle\text{co-author}\rangle\texttt{own} \vee @_{\texttt{own}}\langle\text{-submitter}\rangle\langle\text{expert}\rangle\texttt{req} \tag{Policy 4}$$

The first formula represents the policy that allows a requester to access an object, if they are related to the owner with the 'co-author' relation. In other words, to obtain access, a requester should be categorized as a co-author by the file owner. The second policy grants access to either authors of the object, or to the experts on the platform. The third formula aims to restrict access to authors' names. In that case access is granted only to those requesters, who are co-authors of the authors of the paper. The last policy states that a requester should be a co-author of an object's owner, or she should be an expert. To evaluate each policy we implemented an adopted version of the Local Model Checking algorithm for the ReBAC's Hybrid Model [6]. Our implementation does not significantly affect the complexity of policy evaluation, since it only adds one free variable (to account for relationships with objects). In our implementation we perform simple recursive descent parsing of the given policies. In *Policy 1* and *Policy 4* the free variable `own` is bound to a random user (a user-centric perspective), while in *Policy 2* it is bound to Platform node (a single authority perspective).

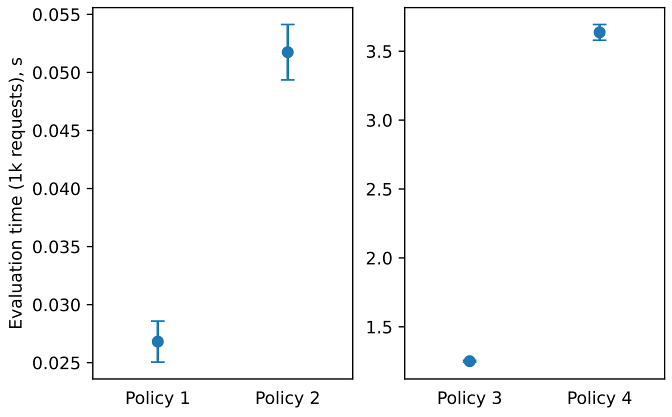


Fig. 3: Evaluation of four access control policies. Each policy was evaluated 1000 times. The experiment was repeated 10 times, the means and 95% confidence intervals are presented.

We performed the evaluation using commodity hardware with the following configuration: AMD Ryzen 7 5700G, 3.8 GHz processor, 32 GB RAM (DDR4, 3200 MHz), WD SN350 NVMe SSD. The computer was running Ubuntu 22.04 operating system with Linux 5.15 kernel. To create our simulated environment we leveraged Python 3.10.12 together with networkx 3.1 library. The latter allows one to create and process graph-based networks [15].

The evaluation results are shown in fig. fig. 3. The first and the second policies were evaluated in less than 60 ms, while for more complex policies it took less than 4 seconds to be evaluated (1000 requests were made). The observed difference can be explained by the structure of the policies. *Policy 1* and *Policy 2* involve only a search of depth of 1 relation. *Policy 3* and *Policy 4* required a search of depth of 2 relations (the 'metadata' relation in *Policy 3* is a one-to-one relation, not significantly affecting the results). Given that many real-life scenarios only involve relation paths of length 1 (a user tags objects and not between-object relations, a company assigns sensitivity labels to employees), we believe that our framework could be a good solution in many practical applications.

## 7 Discussion

Our work can be viewed as an extension of the existing access control models. To achieve user-centricity, we base our model on hybrid logic. The latter allows one to evaluate access requests from a certain user perspective. Additionally, we explicitly rely on categorization, since it is shown to be the basic cognitive mechanism of human reasoning about other people and objects. In the next paragraphs we provide an overview of how our proposition is related to other prominent research models.

The UCAT model can be used to represent simple RBAC policies. One of the core ideas behind RBAC is that an organization assigns roles to users. In UCAT terms, the organization is the owner of the files and it categorizes its employees. For instance, a role 'manager' in an RBAC system can be represented via relation $@_{\mathtt{own}}\langle manager\rangle$. In a similar fashion it is possible to account for the relationships between permissions and allowed actions (for instance, via binding a free variable `dobj` to the requested action). Additionally, UCAT allows one to specify local policies. For example, a global policy might assign Bob to the 'manager' role and authorize other managers to access his files. Bob can additionally label some of his colleagues as 'competitors' and deny them access to 'in-progress' files. Apparently, a conflict resolution scheme should be seen as a prominent future work direction for UCAT. Similarly, a further effort is required to represent hierarchical structures and mutually exclusive roles. In both cases it is required to account for relationships between roles. Thereby, special role entities should be introduced, so it becomes possible to form policies of a type $@_{\mathtt{req}}\langle has\text{-}role\rangle role\_name$.

Our framework can also represent many relationship-based models. Obviously, it can mimic classic ReBAC systems. Additionally, it can be used to substitute some of ReBAC's extensions. CollAC, a model for collaborative access control, introduces additional free variables to account for relations between objects and users. For instance, a Data Host variable is defined for a user, who hosts the data. In UCAT it gets easy to account for such relationships: one can define $\langle data\text{-}host\rangle\mathtt{dobj}$ relation from a user to an object. Similarly, UCAT can account for between-object relations, as it is done in OOReBAC. For instance, to grant access to a co-author to a new version of a document, one can use the following policy: $@_{\mathtt{dobj}}\langle -new\text{-}version\rangle\langle -author\rangle\langle co\text{-}author\rangle\mathtt{req}$. An interesting challenge arise: how can one account for multiple nested relationships? For instance, let's consider a situation when we have several versions of one document. We have a policy for the first document $@_{\mathtt{dobj}}\langle -author\rangle\mathtt{req}$, which grants access to the document to its authors. How can one specify a policy that would grant access to all subsequent versions of the document? One of the workarounds is to introduce an additional relation $\langle has\text{-}root\rangle$ for each descendant. We consider the development of a more general approach to be a prominent future work direction.

Our work goes in line with a spirit of Barker's approach [3]. Barker proposed to elicit a small number of access control primitives and to build a general access control model. Unsurprisingly for us, the proposed primitive was category. Similarly to our work, Barker notes that the notion of roles in RBAC is merely a type of a category. To achieve his goal the author introduces the rule language based on constraint logic programming. In our work we approached the problem differently. Similarly to Barker we noticed the possibility of a generalized model. We also came up with the notion of categories. However, we were guided by a different motivation. First, we wanted to make it possible for users to specify their local "user-perspective" policies, to allow for local policy evaluation. Thereby, we developed a model based on hybrid logic. Second, we aimed to make an easy-to-use solution. Being guided by the evidence from cognitive science on ubiquity of categorization in human reasoning, we chose category as an appropriate primitive for a usable access control model. Given the evident interconnection between Barker's and our works, it seems interesting to investigate the question of general access control models. In that case, one can specifically consider questions of local and global policies together with administrative models for such systems.

## 8 Conclusion

In this paper we introduced UCAT, an access control framework based on uniform categorization. Our work is a generalization of the ReBAC family models. In accordance with the evidence from cognitive science on ubiquity of categorization in human reasoning, we state that a user should be able to categorize not only subjects (e.g. friends and colleagues), but also objects. In our work we define such a model, provide use-case scenarios and evaluate it in the context of a scholarly publishing system. Our results suggest that the model can be an appropriate fit for many applications.

We envision that our work can have many beneficial applications. In many real-life scenarios categories can be a suitable access control primitive, replacing roles, groups, relationships. Thereby, as a prominent research direction we foresee the adoption of uniform categories as the basic primitive for the existing models such as RBAC and ReBAC. Additionally, it is interesting to investigate the question of the combination of global and local access control policies and to develop administrative models for the UCAT framework. We believe that these questions might lead to a more general user-centric access control model.

**Acknowledgements.** This work was co-funded by the European Research Council for the project ScienceGRAPH (Grant agreement ID: 819536) and the TIB Leibniz Information Centre for Science and Technology.

## References

1. Ahmed, T., Patwa, F., Sandhu, R.: Object-to-object relationship-based access control: Model and multi-cloud demonstration. In: 2016 IEEE 17th International Con-

ference on Information Reuse and Integration (IRI). pp. 297–304. IEEE (2016)
2. Augoustinos, M., Walker, I., Donaghue, N.: Social cognition: An integrated introduction. Sage (2014)
3. Barker, S.: The next 700 access control models or a unifying meta-model? In: Proceedings of the 14th ACM Symposium on Access Control Models and Technologies. p. 187–196. SACMAT '09, Association for Computing Machinery, New York, NY, USA (2009). https://doi.org/10.1145/1542207.1542238, https://doi.org/10.1145/1542207.1542238
4. Bell, D.E., Padula, L.J.L.: Secure computer system: Unified exposition and multics interpretation (1976)
5. Blackburn, P., Seligman, J.: Hybrid languages. Journal of Logic, Language and Information **4**, 251–272 (1995)
6. Bruns, G., Fong, P.W., Siahaan, I., Huth, M.: Relationship-based access control: its expression and enforcement through hybrid logic. In: Proceedings of the second ACM conference on Data and Application Security and Privacy. pp. 117–124 (2012)
7. Carminati, B., Ferrari, E., Heatherly, R., Kantarcioglu, M., Thuraisingham, B.: A semantic web based framework for social network access control. In: Proceedings of the 14th ACM symposium on Access control models and technologies. pp. 177–186 (2009)
8. Carminati, B., Ferrari, E., Heatherly, R., Kantarcioglu, M., Thuraisingham, B.: Semantic web-based social network access control. computers & security **30**(2-3), 108–115 (2011)
9. Crampton, J., Sellwood, J.: Relationships, paths and principal matching: a new approach to access control. arXiv preprint arXiv:1505.07945 (2015)
10. Damen, S., Hartog, J., Zannone, N.: Collac: Collaborative access control. pp. 142–149 (05 2014). https://doi.org/10.1109/CTS.2014.6867557
11. Denning, D.E.: A lattice model of secure information flow. Communications of the ACM **19**(5), 236–243 (1976)
12. Eysenck, M.W., Brysbaert, M.: Fundamentals of cognition. Routledge (2018)
13. Ferraiolo, D.F., Barkley, J.F., Kuhn, D.R.: A role-based access control model and reference implementation within a corporate intranet. ACM Transactions on Information and System Security (TISSEC) **2**(1), 34–64 (1999)
14. Fong, P.W.: Relationship-based access control: protection model and policy language. In: Proceedings of the first ACM conference on Data and application security and privacy. pp. 191–202 (2011)
15. Hagberg, A.A., Schult, D.A., Swart, P.J.: Exploring network structure, dynamics, and function using networkx. In: Varoquaux, G., Vaught, T., Millman, J. (eds.) Proceedings of the 7th Python in Science Conference. pp. 11 – 15. Pasadena, CA USA (2008)
16. Leskovec, J., Kleinberg, J., Faloutsos, C.: Graph evolution: Densification and shrinking diameters. ACM transactions on Knowledge Discovery from Data (TKDD) **1**(1), 2–es (2007)
17. Paci, F., Squicciarini, A., Zannone, N.: Survey on access control for community-centered collaborative systems. ACM Computing Surveys (CSUR) **51**(1), 1–38 (2018)
18. Tolone, W., Ahn, G.J., Pai, T., Hong, S.P.: Access control in collaborative systems. ACM Computing Surveys (CSUR) **37**(1), 29–41 (2005)
19. Wilkinson, M.D., Dumontier, M., Aalbersberg, I.J., Appleton, G., Axton, M., Baak, A., Blomberg, N., Boiten, J.W., da Silva Santos, L.B., Bourne, P.E., et al.: The fair guiding principles for scientific data management and stewardship. Scientific data **3**(1), 1–9 (2016)