Towards a Theory for Semantics and Expressiveness Analysis of Rule-Based Access Control Models

Amirreza Masoumzadeh
University at Albany – SUNY
Albany, New York, USA
amasoumzadeh@albany.edu

Paliath Narendran
University at Albany – SUNY
Albany, New York, USA
pnarendran@albany.edu

Padmavathi Iyer
University at Albany – SUNY
Albany, New York, USA
riyer2@albany.edu

ABSTRACT

Recent access control models such as attribute-based access control and relationship-based access control allow flexible expression of authorization policies using the concepts of rules and conditional expressions. The independent nature of policy rules from each other and the amount of flexibility that they enjoy (e.g., the type of conditional expressions they support and whether they can permit or deny matching requests) make those policies quite expressive. But how expressive are they? Do we need to enable all possible flexibilities in a rule-based model to achieve the maximum possible expressiveness? Answering such questions is essential in making informed decisions when designing new models or choosing existing models for implementation. In this paper, we propose an approach towards answering those questions by developing a novel theory for capturing the semantics of rule-based policies depending on their support of different constructs such as flexibility of conditional expressions, rule modalities, and conflict resolution. Our formal policy semantics model enjoys an intuitive design that can capture the semantics of various rule-based policies. We show the well-formedness properties of such semantics and how they can be used to analyze the expressive power of a number of rule-based models.

CCS CONCEPTS

• **Security and privacy → Formal security models**; **Authorization**; **Access control**.

KEYWORDS

conditions, conditional expressions, expressiveness, policy analysis, rule-based access control, semantics

ACM Reference Format:
Amirreza Masoumzadeh, Paliath Narendran, and Padmavathi Iyer. 2021. Towards a Theory for Semantics and Expressiveness Analysis of Rule-Based Access Control Models. In *Proceedings of the 26th ACM Symposium on Access Control Models and Technologies (SACMAT) (SACMAT '21), June 16–18, 2021, Virtual Event, Spain.*. ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3450569.3463569

SACMAT '21, June 16–18, 2021, Virtual Event, Spain.

ACM ISBN 978-1-4503-8365-3/21/06…$15.00
https://doi.org/10.1145/3450569.3463569

1 INTRODUCTION

Access control policies determine authorized operations in computing systems. An access control policy is defined in the context of an access control model which provides a blueprint of how the policy must be designed and how it must be interpreted (i.e., what its semantics is). A common theme for the design of modern access control models is specifying policies in terms of a collection of rules. Each rule typically has a conditional expression that can flexibly determine the applicability of the rule to access requests and a certain effect that directs the access control decision. We call such models *rule-based access control*. For example, an attribute-based access control (ABAC) model [10, 28, 27] is a rule-based model that uses conditional expressions that test subject and object attributes. A relationship-based access control model [12, 7] uses conditional expressions that test paths on a graph of system entities (that can be subjects or objects).

Regardless of the concrete conditional expression model used by a rule-based access control model, we are interested in understanding how factors such as rule composition affect the expressiveness of the model. For instance, a model can allow rules to test for the negation of a condition (i.e., the rule will be applicable when the condition does not hold). Also, the effect of applicable rules may be flexible to permit or deny matching access requests. In such cases, there needs to be a conflict resolution strategy. Does allowing both permit and deny rules add to the expressive power of a model? How does the choice of conflict resolution strategy affect the expressiveness? Assume that we intend to develop a new rule-based model or extend an existing one. How can we decide about including features in the model? Similarly, security developers who intend to implement rule-based access control models in their systems need to decide about the support of various features.

To the best of our knowledge, there is currently no formal approach to answer the above (and other similar) questions about the expressiveness of rule-based models. In order to address this problem, we propose a novel theory for capturing the semantics of rule-based access control policies. While formally capturing the semantics of policies, the proposed approach enjoys a simple presentation and allows for intuitive characterization of policies. We demonstrate how to characterize the expressive power of different rule-based policy models in the context of this semantics model.

We make the following specific contributions in this work:

- We present a framework of rule-based access control policy models that, depending on the choices of conditional expressions, rule modality, conflict resolution strategy, and default decision, leads to different rule-based models. We analyze the expressiveness properties of six of those models in this

paper. We have named those models as Negation, DDDO, DPPO, DDPO, DPDO, and DDFA.
- We develop a theory for representing the semantics of rule-based policies based on a set of conditional expression units, which we call *condition minterms*. Such a semantics model can be used to compare the semantics of policies (even when they are expressed using different models), and the expressive power of models. We show how the semantics of policies in our specific rule-based models can be derived.
- One of the interesting aspects of the proposed semantics model is that the semantics of rules and policies can be characterized based on their well-formedness properties. We show how such policy forms and *policy shapes* can be utilized to provide a formal and intuitive understanding of policy behavior and expressiveness.
- We formally prove and present our first set of expressiveness results obtained based on the proposed policy semantics model for the selected rule-based models in this paper. In particular, we show that Negation and DDFA are the most expressive models, while others can express proper subsets of the policy semantics space.

The rest of the paper is organized as follows. We present a framework of rule-based access control policy models in Section 2. We introduce and formalize the semantics of rule-based policies in terms of condition minterms in Section 3, and characterize their well-formedness properties in Section 4. In Section 5, we provide formal expressiveness analyses of a number of rule-based policy models. In Section 6, we present a concrete use case scenario of rule-based policies to demonstrate the implications of the expressive power of models. We further discuss about our choices of policy models and implications of our results in Section 7. Finally, we provide an overview of closely related work in Section 8, and discuss our contributions and future work in Section 9.

2 FRAMEWORK OF RULE-BASED POLICY MODELS

In this section, we present a general framework of rule-based policy models. We consider an abstract notion of *conditions* that rules in a rule-based policy evaluate in order to determine authorizations. Each condition has certain system-dependent semantics and is evaluated using some authorization information (usually about the subject and object of access). For instance, an attribute-based access control (ABAC) policy has a concrete condition model based on subject and object attributes. A certain attribute-based condition in a university system, for example, may test if the subject is a student. As another example, a relationship-based access control (ReBAC) policy has a concrete condition model that tests a relationship path between the subject and the object on a relationship graph of system entities. In this paper, we abstractly assume that the access control policy is capable of testing set of conditions C.

Rule: The unit of decision making in a rule-based policy is a rule denoted by $\langle \phi, d \rangle$ where ϕ is a conditional expression specifying to which access requests the rule is applicable, and d is the associated decision when the rule is applicable. The format of those elements is specified based on the specific conditional expression and rule modality choice of the model.

Conditional Expression: The conditional expression is specified based on a conjunction of supported conditions (C). A model can choose to support only *uncomplemented conditions* in its expressions. We denote such conditions by a concatenation of the involved conditions. For example, conditional expression c_1c_2 is satisfied when both conditions c_1 and c_2 are satisfied. We also consider a special uncomplemented conditional expression *true*, denoted by $\top$, that is always satisfied. The true conditional expression can be seen as *containing no uncomplemented conditions*, meaning that it does not require any of the conditions to satisfy. A model can also support *complemented (negated) conditions* in addition to uncomplemented conditions. We denote the complement of a condition using a bar sign. For example, conditional expression $c_1\overline{c_2}$ is satisfied when c_1 is satisfied and c_2 is unsatisfied.

Rule Modality: Rule decisions can be either of two modalities, `PERMIT` or `DENY`. We consider models that only support `PERMIT` decisions (as customary to many access control systems), as well as those that allow both `PERMIT` and `DENY` rules. `DENY` rules are also known as negative authorizations in the literature.

Ruleset: The policy ruleset is defined as an ordered set of individual rules. Semantically, a ruleset can be viewed as a disjunction of the individual rules. While the notion of a set is sufficient for most models, as we will discuss below, certain conflict resolution strategies may require a total ordering among rules. We denote a ruleset as a sequence $\langle r_1, r_2, \ldots, r_n \rangle$ (where $|R| = n$). Here, r_i has higher rank than r_j (where $i < j$).

Conflict Resolution: When a model supports both rule modalities (`PERMIT` and `DENY`), the conflict resolution strategy determines the decision when multiple rules are applicable to a given request. We consider three commonly-used strategies. The `DENY`-*overrides* and `PERMIT`-*overrides* prioritize `DENY` and `PERMIT` decisions, respectively. The *first-applicable* strategy prioritizes the decision of the first rule that is applicable to a given request.

Default Decision: The default decision determines the authorization decision when no rule is applicable to a given request. The choices are `DENY` (restrictive and more common) or `PERMIT` (permissive).

Depending on the choice of the abovementioned factors of conditional expressions, rule modality, conflict resolution (if needed), and default decision, different rule-based models can be produced. For example, perhaps the most basic model would allow only uncomplemented conditional expressions, `PERMIT` rules, and `DENY` as default. In this paper, we focus on the following set of interesting and commonly-used models based on this framework. Note that due to to the number of factors affecting a model, we do not use a universal naming convention for models produced by the framework.

Negation Model supports conditional expressions with both complemented and uncomplemented conditions, only `PERMIT` rules, and `DENY` as default decision.

Additionally, we define the following models that allow both rule modalities and conditional expressions containing only uncomplemented conditions. However, they use different default decision and conflict resolution strategies:

DDDO Model default `DENY` (abbrv. DD) and `DENY`-overrides (abbrv. DO).

DPPO Model default PERMIT (abbrv. DP) and PERMIT-overrides (abbrv. PO).
DDPO Model default DENY and PERMIT-overrides.
DPDO Model default PERMIT and DENY-overrides.
DDFA Model default DENY and first-applicable (abbrv. FA).

It is worth noting that the above list of models is not exhaustive. They are chosen either because they are commonly used and/or have certain interesting theoretical characteristics which we are concerned about in this paper.

We show two small sample policies based on the Negation and DDDO models below.

Example 2.1 (Negation Policy). The following Negation policy permits access only when c_1 is satisfied and c_3 is not satisfied, or when c_2 is satisfied and c_3 is not satisfied. Access will be denied in all other conditions:

$\langle c_1\overline{c_3}, \texttt{PERMIT}\rangle$
$\langle c_2\overline{c_3}, \texttt{PERMIT}\rangle$

Example 2.2 (DDDO Policy). The following DDDO policy permits access when c_1 or c_2 is satisfied. However, it will deny access when c_3 is satisfied *irrespective of whether* c_1 *or* c_2 *is satisfied*. Note the conflict between the two cases and that DENY overrides. All other accesses are denied as well:

$\langle c_1, \texttt{PERMIT}\rangle$
$\langle c_2, \texttt{PERMIT}\rangle$
$\langle c_3, \texttt{DENY}\rangle$

3 POLICY SEMANTICS

We develop a theory for representing the semantics of rule-based policies based on the semantics of conditional expressions in their rules. We show that conditional expressions themselves can be decomposed into conditional units, which we call *condition minterms*. As the name might suggest, we have borrowed the concept of *minterms* from digital logic design.

3.1 Condition Minterms

Definition 3.1 (Condition Minterm). Given the set of conditions C, $|C| = n$, a *condition minterm* (*minterm* for short) is a boolean function constructed as conjunction of n literals where each literal is either a distinct $c_i \in C$, or its complement $\overline{c_i}$. We denote the set of all minterms based on conditions C by $\mathcal{M}_C$.

Since all conditions in C are present in a minterm, in either their direct (uncomplemented) form or their complemented form, there are $2^{|C|}$ possible minterms. Let us consider condition set $C = \{c_1, c_2, c_3\}$. Then, $\mathcal{M}_C$ is

$\{\overline{c_1c_2c_3},\ \overline{c_1c_2}c_3,\ \overline{c_1}c_2\overline{c_3},\ \overline{c_1}c_2c_3,\ c_1\overline{c_2c_3},\ c_1\overline{c_2}c_3,\ c_1c_2\overline{c_3},\ c_1c_2c_3\}$

In the rest of this paper, unless explicitly mentioned, we assume the same set of conditions C in our examples. Each condition minterm captures one possible combination of truth values for all tested conditions. Therefore, the set of all minterms represents the complete space of possible outcomes of condition evaluations. Given an access request, at least one condition minterm (and possibly more) will be evaluated as true.

We define a partial order relation ($\leq$) on condition minterms based on their uncomplemented conditions as follows:

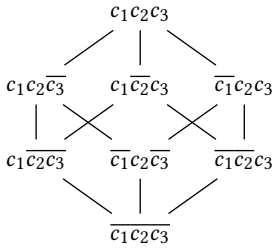


Figure 1: Lattice of Condition Minterms for $C = \{c_1, c_2, c_3\}$

Definition 3.2 (Partial Ordering of Minterms). For a given pair of minterms $m_1, m_2 \in \mathcal{M}_C$, let S_1 and S_2 be the set of uncomplemented conditions in them, respectively. $m_1 \leq m_2$ if and only if $S_1 \subseteq S_2$.

For example, $\overline{c_1}c_2\overline{c_3} \leq c_1c_2\overline{c_3}$ and $c_1\overline{c_2c_3} \leq c_1c_2c_3$. However, $\overline{c_1}c_2\overline{c_3} \not\leq c_1\overline{c_2c_3}$.

It is trivial to define a *strict order relation* ($<$) on minterms based on the above definition as $\leq \ \wedge \ \neq$. The notions of (immediate) predecessor/successor are defined based on the strict ordering.

Definition 3.3 (Predecessor and Successor). Given minterms $m_1 < m_2$, m_1 is called a *predecessor* of m_2, and m_2 is called a *successor* of m_1. Furthermore, given S_1 and S_2 as defined in Definition 3.2, m_1 is said to be an *immediate predecessor* of m_2 (and m_2 an *immediate successor* of m_1) if and only if $|S_2 \setminus S_1| = 1$ (i.e., they differ in only one uncomplemented condition). We denote this as $m_1 \lessdot m_2$.

The partial ordering among condition minterms forms a lattice: there is a supremum and an infimum for every pair of minterms. Figure 1 depicts the Hasse diagram of the condition minterms for conditions $C = \{c_1, c_2, c_3\}$.

3.2 Semantics of Policies as Sets of Minterms

A rule-based access control policy uses rules (and other metadata as discussed in Section 2) to establish the conditions under which an access request is granted or denied. Therefore, the policy can be viewed as partitioning the condition minterm space into authorized and unauthorized minterms. This makes the condition minterms a great choice to represent the semantics of policies.

Definition 3.4 (Policy-Minterms). Given a policy $p \in P$ (P represents the space of all possible policies), the *policy-minterms* function $\mu : P \rightarrow 2^{\mathcal{M}_C}$ represents the semantic of the policy in the form of its set of permitted condition minterms.

Based on the above definition, $\mu(p)$ characterizes the PERMIT space of policy p. Conversely, the DENY space of the policy can be characterized by $\mathcal{M}_C \setminus \mu(p)$.

We slightly abuse the above μ notation to also show the minterms corresponding to the conditional expression of a rule. This allows us to formally define the semantics of the policies based on the semantics of their rules.

Definition 3.5 (Expression-Minterms). Given conditional expression $\phi \in \Phi$ (Φ represents the space of all possible conditional expressions), the *expression-minterms* function $\mu : \Phi \rightarrow 2^{\mathcal{M}_C}$ is defined as the set of minterms that are covered by ϕ.

The coverage of a conditional expression can be calculated by enumerating both complemented and uncomplemented versions of conditions that are not present in the expression. For instance, $\mu(c_1\overline{c_3}) = \{c_1\overline{c_2}\overline{c_3}, c_1c_2\overline{c_3}\}$. As another example, $\mu(c_1) = \{c_1\overline{c_2}\overline{c_3}, c_1\overline{c_2}c_3, c_1c_2\overline{c_3}, c_1c_2c_3\}$.

The policy-minterms for a given policy can be derived from expression-minterms of its rules while considering the rule modalities and other decision factors in its policy model. In Table 1, we present how the semantics of the specific policy models presented in Section 2 can be formulated. In the Negation model, it is sufficient for an access request to satisfy one of the rules to be permitted. Therefore, each rule effectively contributes a set of permitted condition minterms to the policy. The union of those contributed sets constitutes the policy-minterms for the Negation policy. In the case of the DDDO model, the first term of the expression calculates the union of permitted condition minterms (associated with `PERMIT` rules). However, since a DDDO policy follows the `DENY`-overrides conflict resolution strategy, condition minterms corresponding to `DENY` rules need to be removed from the permitted minterms, and will not be in the final set of policy-minterms. In the case of DPPO, since everything is permitted by default, we start with the set of all possible minterms and adjust that based on the rules. Since `PERMIT` overrides, we first remove all denied minterms followed by adding back those explicitly permitted minterms. The case of DDPO is simpler. Since the model denies by default but lets the `PERMIT` rules to override, the `DENY` rules do not impact the outcome of the policy. Therefore, its semantics can be specified by the union of expression-minterms of its `PERMIT` rules. Similarly, in the case of DPDO, we start from the set of all minterms but we only need to remove those condition minterms that are explicitly denied by `DENY` rules. In other words, `PERMIT` rules have no effect on the semantics. The case of DDFA is slightly more complex since the order of rules in the policy needs to be considered. As shown in the pseudocode in the table, we need to process rules in reverse order and consider the impact of each `PERMIT`/`DENY` rule by adding/removing its corresponding minterms. The reverse processing order allows the semantics of rules with a lower index to take precedence over rules with a higher index.

The underlying pattern that can be seen in all of the derived formulas discussed above is to process components of a policy in increasing order of their semantic priority. In particular, the default policy has the lowest priority (compared to explicitly specified rules), and the conflict resolution policy determines the order in which the rules need to be processed.

3.3 Semantic Equivalence

The proposed representation of policy semantics in Section 3.2 can be used to accurately reason about policy equivalence based on policy-minterms even when policies are expressed using different models. We define the semantic equivalence of policies .

Definition 3.6 (Semantic Equivalence of Policies). Policies p_1 and p_2 are considered *semantically equivalent* if and only if $\mu(p_1) = \mu(p_2)$.

Example 3.7 (Policy Equivalence). We can use the formula in Table 1 to show that the Negation policy p_1 and the DDDO policy p_2, described in Examples 2.1 and 2.2, respectively, are semantically equivalent:

$$\begin{aligned}
\mu(p_1) &= \mu(c_1\overline{c_3}) \cup \mu(c_2\overline{c_3}) \\
&= \{c_1\overline{c_2}\overline{c_3}, c_1c_2\overline{c_3}\} \cup \{\overline{c_1}c_2\overline{c_3}, c_1c_2\overline{c_3}\} \\
&= \{\overline{c_1}c_2\overline{c_3},\ c_1\overline{c_2}\overline{c_3},\ c_1c_2\overline{c_3}\} \\
\mu(p_2) &= \mu(c_1) \cup \mu(c_2) \setminus \mu(c_3) \\
&= \{c_1\overline{c_2}\overline{c_3}, c_1\overline{c_2}c_3, c_1c_2\overline{c_3}, c_1c_2c_3\} \\
&\quad \cup \{\overline{c_1}c_2\overline{c_3}, c_1c_2\overline{c_3}, \overline{c_1}c_2c_3, , c_1c_2c_3\} \\
&\quad \setminus \{\overline{c_1}\overline{c_2}c_3, \overline{c_1}c_2c_3, c_1\overline{c_2}c_3, c_1c_2c_3\} \\
&= \{\overline{c_1}c_2\overline{c_3},\ c_1\overline{c_2}\overline{c_3},\ c_1c_2\overline{c_3}\}
\end{aligned}$$

4 POLICY SHAPES

The sets of condition minterms, as shown in Section 3.2, can capture the semantics of policies. In this section, in order to facilitate comparing semantics of policies and models, we further characterize such sets of condition minterms. In particular, we show that we can derive theoretical properties about the composition of sets of condition minterms that represent the semantics of conditional expressions and rule-based policies. Such compositions can be visualized to enable more intuitive analysis, and hence, we call them *shapes*.

4.1 Rules as Well-Formed Sets of Minterms

We recall that sets of minterms are partially ordered sets according to Definition 3.2. We define two well-formedness properties for sets of minterms: *convexity* and *upward-closedness*.

Definition 4.1 (Convex Set of Minterms). A set of minterms M is *convex* if and only if for every pair of minterms $m_1 \leq m_2$ in M: $\{m \mid m_1 \leq m \leq m_2\} \subseteq M$.

Definition 4.2 (Upward-Closed Set of Minterms). The upward closure of set of minterms M, denoted by $M^{\uparrow}$, is $\{u \mid \exists m \in M : m \leq u\}$. Set of minterms M is *upward-closed* if and only if $M^{\uparrow} = M$.

In other words, the above definitions state that a convex set of minterms must include every minterm in between any pair of its elements. Also, in an upward-closed set of minterms all successors of a member must also belong to the set.

Interestingly, the sets of minterms capturing the semantics of rules in a rule-based policy are well-formed. In the following two lemmas, we show that conditional expressions that only use uncomplemented conditions are upward-closed, while those that use both complemented and uncomplemented conditions are convex. Recall that we use the $\mu()$ notation to represent the semantics of both conditional expressions and policies.

Lemma 4.3. *For a conditional expression ϕ that only uses uncomplemented conditions, $\mu(\phi)$ is upward-closed.*

Proof. $\mu(\phi)$ is the upward closure of minterm $\phi\omega'$ where ω' is the product (*and*) of the complements of all the conditions that are not present in ϕ. □

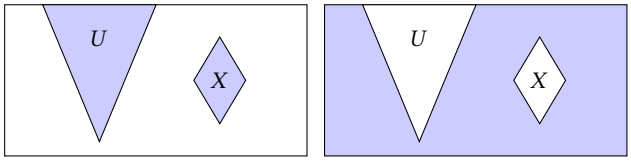
For instance, if $\phi = c_1c_2$, then $\mu(\phi)$ is the upward closure of the minterm $c_1c_2\overline{c_3}\ldots\overline{c_n}$.

Lemma 4.4. *For a conditional expression ϕ that uses both complemented and uncomplemented conditions, $\mu(\phi)$ is convex.*

Table 1: Policy Model Semantics Calculated Based on Rule Semantics

Model of Policy p	Semantics of p: $\mu(p)$
Negation	$\bigcup_{r \in p} \mu(r.\phi)$
DDDO	$\bigcup_{r \in p \wedge r.d=\texttt{PERMIT}} \mu(r.\phi) \setminus \bigcup_{r \in p \wedge r.d=\texttt{DENY}} \mu(r.\phi)$
DPPO	$\mathcal{M}_C \setminus \bigcup_{r \in p \wedge r.d=\texttt{DENY}} \mu(r.\phi) \cup \bigcup_{r \in p \wedge r.d=\texttt{PERMIT}} \mu(r.\phi)$
DDPO	$\bigcup_{r \in p \wedge r.d=\texttt{PERMIT}} \mu(r.\phi)$
DPDO	$\mathcal{M}_C \setminus \bigcup_{r \in p \wedge r.d=\texttt{DENY}} \mu(r.\phi)$

DDFA	$M \leftarrow \emptyset$
	for $i = n$ to 1 **do**
	  if $r_i.d =$ PERMIT **then**
	  $M \leftarrow M \cup \mu(r_i.\phi)$
	  else if $r_i.d =$ DENY **then**
	  $M \leftarrow M \setminus \mu(r_i.\phi)$
	$\mu(p) \leftarrow M$



(a) PERMIT Rules in a Default-DENY Space **(b) DENY Rules in a Default-PERMIT Space**

Figure 2: Illustrating Rules as Well-Formed Sets of Minterms. U is an Upward-Closed Set (Triangle-Shaped) and X is a Convex Set (Diamond-Shaped). Filled (Blue) Area Represents Permitted Condition Minterms. Unfilled (White) Area Represents Denied Condition Minterms

Proof. Let $\phi = c_1 \dots c_i \, \overline{c_{i+1}} \dots \overline{c_j}$. Every minterm in $\mu(\phi)$ has ϕ as a subterm, and conversely every minterm that contains ϕ as a subterm is in $\mu(\phi)$. Suppose $m_1 < m < m_2$ where both m_1 and m_2 belong to $\mu(\phi)$ and m does not. But since ϕ is a subterm of m_1 and m_2, it must be a subterm of m as well. □

For the specific policy models discussed in Section 2, conditional expressions in the Negation model can use both complemented and uncomplemented conditions. Therefore, each Negation rule is convex. In contrast, the rest of the models that we considered (i.e., DDDO, DPPO, DDPO, DPDO, and DDFA) only allow uncomplemented conditions. Therefore, each of their rules is upward-closed. Note that for policies that support both rule modalities, DENY rules are effectively specifying an upward-closed set of minterms with the intention of denying them.

In order to provide a more intuitive approach to think about the behavior of rules and policies, we use a pictorial approach for well-formed sets of minterms by borrowing ideas from Venn diagrams and Hasse diagrams. Note that, unlike Venn diagrams that are useful for analyzing unordered sets, here, we are dealing with partially ordered sets. Therefore, the y-axis in our diagrams intends to mimic the hierarchical relation captured in Hasse diagrams. Figure 2 shows two example minterm spaces. The rectangle represents the whole set of possible minterms, $\mathcal{M}_C$. An upward-closed subset of minterms is drawn as an upside down triangle (since it includes all higher elements). That corresponds to a rule that has a conditional expression with only uncomplemented conditions. A convex subset of minterms is drawn as a diamond. That corresponds to a rule that has a conditional expression with both complemented and uncomplemented conditions. The filled area indicates the permitted condition minterms (policy-minterms). In Figure 2a, upward-closed set U and convex X represent PERMIT rules, while the default decision is assumed to be DENY. In contrast, in Figure 2b, U and X represent DENY rules, while the default decision is PERMIT.

We will see in the next section how such semantics can be used to characterize the form/shape of policies as a whole.

4.2 Operations on Well-Formed Sets of Minterms

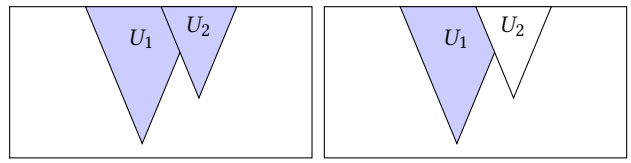
In order to characterize policy semantics based on the characteristics of their underlying rules, we establish a set of properties for combining well-formed sets of minterms.

Lemma 4.5. *Given upward-closed sets of minterms U_1 and U_2, and convex sets of minterms X_1 and X_2, the following properties hold:*

(1) U_1 is convex.
(2) $U_1 \cup U_2$ is upward-closed.
(3) $U_1 \setminus U_2$ is convex, but not necessarily upward-closed.
(4) $X_1 \cup X_2$ is not necessarily convex.
(5) $X_1 \setminus X_2$ is not necessarily convex.
(6) $X_1 \cup U_2$ is not necessarily convex.
(7) $U_1 \setminus C_1$ is not necessarily convex.
(8) $X_1 \setminus U_1$ is convex.

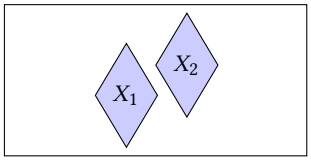
Figure 3 illustrates examples of the above operations on well-formed sets of minterms. The example diagrams help visual inspection of the properties. We also provide a proof for the property in Lemma 4.5(3) as an example. The rest of the properties can be similarly proven.

Proof of Lemma 4.5(3). First, we prove that $U_1 \setminus U_2$ is convex by contradiction. Assume that $U_1 \setminus U_2$ is not a convex. Therefore, $\exists m_1 \leq m_2 \leq m_3$ where $m_1, m_3 \in U_1 \setminus U_2$, but $m_2 \notin U_1 \setminus U_2$. Those, respectively, require that $m_3 \notin U_2$ and $m_2 \in U_2$. This results in a

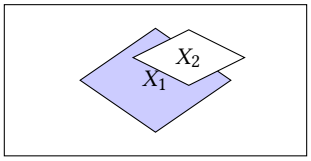


(a) $U_1 \cup U_2$ **is upward-closed**

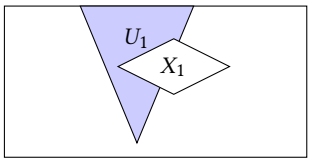
(b) $U_1 \setminus U_2$ **is convex, but not upward-closed**



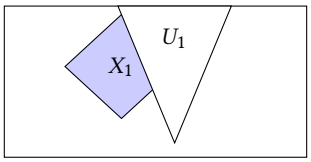
(c) $X_1 \cup X_2$ **is not convex**



(d) $X_1 \setminus X_2$ **is not convex**



(e) $U_1 \setminus X_1$ **is not convex**



(f) $X_1 \setminus U_1$ **is convex**

Figure 3: Examples of Operations on Well-Formed Sets of Minterms. U_i **is Upward-Closed and** X_j **is Convex**

contradiction because upward-closedness of U_2 requires $m_3 \in U_2$ (recall $m_2 \leq m_3$).

Next, we show that $U_1 \setminus U_2$ may not be upward-closed. Consider minterms $m_1, m_2 \in U_1$ where $m_1 \leq m_2$. Now consider the case that $m_1 \notin U_2$ while $m_2 \in U_2$. In such a case $U_1 \setminus U_2$ will clearly include m_1, but not m_2. Therefore, it will not be upward-closed. □

4.3 Shapes of Rule-Based Policies

We use the policy semantics discussed in Section 3.2 and the operation properties that were discussed in Section 4.2 to reason about the shape of policies in each model. It should be noted that the following is applicable in general to every policy that follows a particular policy model. We recall that rules in Negation policies are convex while rules in other models that we are interested are upward-closed.

Theorem 4.6 (Shapes of Policy Models). *Given the detailed semantics shown in Table 1 and operation properties discussed in Lemma 4.5, the following statements are correct about the shape of the policies categorized by their model:*

(1) DDDO policies are convex.

Proof Sketch. Each union of PERMIT *rules and* DENY *rules are separately upward-closed. The subtraction of the two upward-closed sets of minterms produces a convex set of minterms (according to Lemma 4.5(2)).*

(2) DDPO policies are upward-closed.

Proof Sketch. The union of PERMIT *rules will be upward-closed (according to Lemma 4.5(2)).*

(3) DPDO policies are convex.

Proof Sketch. The set of all minterms is upward-closed. The union of DENY *rules is upward-closed as well. Therefore, subtracting latter from the former will be convex (according to Lemma 4.5(3)).*

(4) DPPO policies are not necessarily convex, but their complement semantics (the set of denied minterms) are convex.

Proof Sketch. The first two parts in the semantics of DPPO policies are similar to DPDO policies (convex). The last part (i.e, the union of PERMIT *rules) is upward-closed. However, the subtraction of an upward-closed set from a convex set does not need to be convex (according to Lemma 4.5(8)). The complement semantics of DPPO policies can be derived by subtracting the union of* PERMIT *rules from the union of* DENY *rules. Similar to DDDO policies, such a set of minterms is convex.*

(5) DDFA policies are not necessarily convex.

Proof Sketch. In the pseudocode (in Table 1) describing the semantics of a DDFA policy, the set of minterms M is initially empty, which is upward-closed. In each iteration of the loop, if the current set M is upward-closed, it can remain upward-closed when visiting a PERMIT *rule (according to Lemma 4.5(2)) or can become non-convex when visiting a* DENY *rule (according to Lemma 4.5(7)). Once M is not convex anymore, future iterations will not necessarily make M convex either.*

(6) Negation policies are not necessarily convex.

Proof Sketch. Each rule in a Negation policy is convex, but not necessarily upward-closed. Therefore, their union is not necessarily convex (according to Lemma 4.5(4)).

5 ANALYZING EXPRESSIVENESS OF MODELS

5.1 Expressiveness of Models: Case of Negation

The expressive power of policy models refers to their ability to represent a diverse range of policies. The proposed model of semantics for policies in Section 3 provides the needed framework in which the expressive power of models can be compared. Our first expressiveness result is about the Negation model.

Theorem 5.1. *The Negation model can express any possible semantics.*

Proof. In the most granular form, each Negation policy rule can express a single permitted condition minterm. This is when the conditional expression includes all conditions in their either complemented or uncomplemented form. Since the semantics of a Negation policy is the union of the semantics of its rules (refer to Table 1), one can express any set of minterms by using one rule per minterm. □

Are the other models presented in this paper as expressive as Negation? How do they compare against each other? We may already have some guesses or partial answers to those questions, for example, based on Theorem 4.6. However, a deeper analysis is needed to address those questions. In the rest of this section, we present a few such interesting results.

5.2 Representing Convex Sets of Minterms as DDDO Policies

In Theorem 4.6(1), we showed that DDDO policies are convex sets of minterms. In order to fully characterize the expressiveness power of DDDO, we need to investigate whether an arbitrary convex set of minterms can be represented using a DDDO policy. We first show that a set of DDDO rules can express an upward-closed set of minterms.

Lemma 5.2. *Every upward-closed set of minterms can be represented using a set of rules with uncomplemented conditions.*

Proof. Let M be an upward-closed set of minterms. Let $M_{\min}$ be the minimal minterms in M (i.e., $M_{\min} = \{m_i \mid \nexists m_j \in M, m_j < m_i\}$). Then, M is equal to $M_{\min}^{\uparrow}$ (upward closure of $M_{\min}$). The upward-closure can be represented by the set of PERMIT rules each corresponding to a minimal minterm in M. The condition for such a rule is the product of all the positive literals of the corresponding minterm. □

Next, we prove that any convex set of minterms can be formulated as the set difference of two upward-closed sets of minterms.

Lemma 5.3. *For every convex set of minterms X, there are upward-closed sets U_1 and U_2 such that $X = U_1 \setminus U_2$.*

Proof. If X is upward-closed, the property holds by considering $U_1 = X$ and $U_2 = \emptyset$. Otherwise, let

$$D = \{y \mid \forall x \in X : y \not\leq x\}$$

We note that D is upward-closed since by definition there does not exist any minterm higher than D's elements. We now show that $X = X^{\uparrow} \setminus D$. Suppose $x \in X$. Then, clearly $x \in X^{\uparrow}$. But, $x \notin D$ based on D's definition. On the other hand, suppose $y \in X^{\uparrow} \setminus D$ but $y \notin X$. Then, $y \in X^{\uparrow} \setminus X$. This means that $\forall x \in X, y \not\leq x$, which subsequently means $y \in D$. This is a contradiction since if $y \in D$, then $y \notin X^{\uparrow} \setminus D$.

Therefore, we can choose $U_1 = X^{\uparrow}$ and $U_2 = D$. □

We are finally ready to present the main result of this section:

Theorem 5.4. *Every convex set of minterms can be represented using a DDDO policy.*

The above theorem is a direct result of the Lemmas 5.2 and 5.3. This becomes more clear if you refer to the DDDO semantics presented in Table 1. One can first capture a given convex set as the set difference of two upward-closed sets (Lemma 5.3). Then, according to Lemma 5.2, the first upward-closed set can be captured using a set of PERMIT rules, while the second upward-closed set can be captured using a set of DENY rules.

Theorem 5.4 along with Theorem 4.6(1) precisely capture the expressive power of DDDO policy model, which is equivalent to all convex sets of minterms in our semantics model.

5.3 Expressive Power of DDFA

We showed in Theorem 4.6 that both Negation and DDFA policies are not necessarily convex. As also shown earlier, Negation can express any authorization semantics. In this section, we explore whether DDFA enjoys similar expressiveness. This question is especially interesting since DDFA supports simpler rule expressions compared to Negation, but in contrast, supports both rule modalities and comes with an order-sensitive conflict resolution strategy.

Algorithm 1 Representing a Set of Minterms using DDFA

```
function MintermSetToDDFA(M)        ▷ M: Set of Minterms
    p ← ⟨⟩
    for all minterms m ∈ M_C in reverse topological order do
        φ ← conjunction of uncomplemented conditions in m
        if m ∈ M then
            Append rule ⟨φ, PERMIT⟩ to p
        else
            Append rule ⟨φ, DENY⟩ to p
    return p
```

Theorem 5.5. *The DDFA model can express any policy semantics.*

Proof. The theorem can be proven by showing that any set of minterms can be captured using a DDFA policy. Rather than proving the existence of such a DDFA policy, we propose a naive algorithm to produce the corresponding DDFA policy, and subsequently prove its correctness. Algorithm 1 loops through all possible minterms in their reverse topological order. This means if $m_i < m_j$, m_j will be ahead of m_i in the new order. For example, minterm $c_1c_2\overline{c_3}$ is processed before minterm $c_1\overline{c_2}\overline{c_3}$; but minterms $c_1c_2\overline{c_3}$ and $c_1\overline{c_2}c_3$ can be processed in arbitrary relative order. If the minterm at hand is part of the desired minterm set M a corresponding PERMIT rule is added. Otherwise, a corresponding DENY rule is added. The proposed algorithm is correct if and only if $\mu(p) = M$. We first show that given $m \in M$, $m \in \mu(p)$. In the loop iteration corresponding to m, clearly $m \in \mu(\phi)$, and a corresponding rule $r = \langle \phi, \texttt{PERMIT} \rangle$ is added to the policy. Now, observe the DDFA semantics shown in the pseudocode in Table 1. Rule r will clearly contribute m to $\mu(p)$. We will have $m \notin \mu(p)$ at the end of processing policy semantics only if a later processed rule $r' = \langle \phi', \texttt{DENY} \rangle$ removes m. In that case:

$$m \in \mu(\phi') \tag{1}$$

Let m' be the denied minterm for which r' was added. Since the loop in Table 1 processes rules in reverse order, Algorithm 1 must have appended r' earlier than r. Therefore, we must have $m' \not< m$ due to the reverse topological order. Let S and S' be the set of uncomplemented conditions in m and m', respectively. Since $m' \not< m$, we have:

$$S' \not\subset S \tag{2}$$

Equation 2 contradicts Equation 1 which itself requires $S' \subseteq m$ by definition. Therefore, DENY rule r' cannot exist, and hence, $m \in \mu(p)$. We can similarly show that if $m \in \mu(p)$ then $m \in M$. □

5.4 Expressiveness Hierarchy of Models

Figure 4 summarizes our key findings regarding the expressiveness of the rule-based models as a Venn diagram. The name of each model is shown inside its set area. Note that DDPO is drawn as a shaded area at the intersection of DDDO and DPPO. We have formally proven some of those findings earlier in this paper and

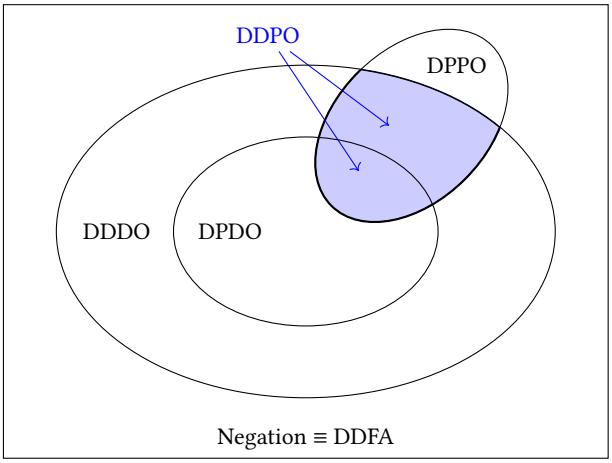


Figure 4: Expressiveness Hierarchy of Rule-Based Policy Models Presented in This Paper

informally argue about the others in this section. Further formal analysis will be of our future work.

As shown in Sections 5.1 and 5.3, Negation and DDFA can express any intended semantics. We also showed in Section 5.2 that DDDO is capable of expressing any semantics in the shape of a convex set of minterms, which will be the subset of possible semantics. DPDO policies were shown to be convex too (Theorem 4.6(3)). However, they are not as expressive as DDDO policies (i.e., they cannot express every possible convex set). Consider convex set of minterms $M = \{\overline{c_1}c_2\overline{c_3}, c_1\overline{c_2}\overline{c_3}\}$. It can be shown that no DPDO policy can represent M. In general, DPDO policies can only include the minimum minterm ($\overline{c_1}\overline{c_2}\overline{c_3}$) by using rule $\langle \top, \texttt{DENY} \rangle$. But including such a rule will result in the minimum minterm being the only permitted minterm (since DENY overrides).

In Theorem 4.6(4), we showed that the DPPO policies are not necessarily convex themselves. Still, DPPO intersects with DDDO (and DPDO) because particular DPPO policies can be convex. We also mentioned that the complement of a DPPO policy is convex. Therefore, the DPPO model is not general enough to represent an arbitrary policy. This makes DPPO a proper subset of Negation and DDFA.

DDPO policies are upward-closed (Theorem 4.6(2)), and any upward-closed set of minterms is convex (Lemma 4.5(1)). This makes DDPO a proper subset of DDDO. DDPO is also a proper subset of DPPO since adding rule $\langle \top, \texttt{DENY} \rangle$ to a DPPO policy can simulate the default-DENY behavior of DDPO. One can also show that there are policies expressible at the intersection of DDPO and DPDO.

6 USE CASE OF EXPRESSIVENESS RESULTS

In this section, we provide a use case scenario based on a realistic policy to demonstrate our concept of policy shape. Specifically, we illustrate how to utilize this concept to determine whether a given Negation policy can be represented in the DDDO model.

We consider a simple policy in the ABAC model for an online university application. In addition, suppose the subjects involve students and the objects involve the coursework (e.g. assignments, exams, etc.) for a course. The authorizations are specified in terms of attributes of subjects and objects, and determine if a given student can access particular coursework.

In the following, we initially represent a sample ABAC policy in the Negation model, and then demonstrate that the specified Negation policy is not representable in the DDDO model. Particularly, we show that the semantics of the policy is a set of minterms that is not convex (and therefore not representable in DDDO according to Theorem 4.6(1)).

Before discussing the policy, we first specify the attributes and the corresponding conditions on subjects and objects in the given university application. Suppose the university application consists of the following attributes about its entities (students and courseworks):

- Subject attributes:
  - `coursesCurrentlyEnrolled` (set value)
  - `coursesPreviouslyTaken` (set value)
- Object attributes:
  - `partOfCourse` (single value)
  - `restrictedToCurrentStudents` (Boolean value).

Based on the attributes enumerated above, suppose the implemented access control system consists of the following conditions:

- c_1: `res.partOfCourse` $\in$ `usr.coursesPreviouslyTaken`
  (Whether the student has already taken the course to which the coursework belongs.)
- c_2: `res.partOfCourse` $\in$ `usr.coursesCurrentlyEnrolled`
  (Whether the student is currently enrolled in the course ro which the coursework belongs.)
- c_3: `res.restrictedToCurrentStudents`
  (Whether the coursework is restricted to only the currently enrolled students.)

Finally, suppose the policy enforced by the access control system consist of the following rules, represented in the Negation model:

- $\langle c_2, \texttt{PERMIT} \rangle$ (students can access the coursework for a course they are currently enrolled in)
- $\langle c_1\overline{c_3}, \texttt{PERMIT} \rangle$ (students can only access the unrestricted coursework for a course that they have previously taken)

Note that since we are representing the above policy in the Negation model, the rule modality is only PERMIT. However, both complemented and uncomplemented conditions are supported in the rules. The coverage of conditional expressions used in the rules are as follows:

- $\mu(c_2) = \{\overline{c_1}c_2\overline{c_3}, \overline{c_1}c_2c_3, c_1c_2\overline{c_3}, c_1c_2c_3\}$
- $\mu(c_1\overline{c_3}) = \{c_1\overline{c_2}\overline{c_3}, c_1c_2\overline{c_3}\}$

Therefore, based on the expression-minterms of the rules, the semantics of the given Negation policy can be formulated as:

$$\mu(c_2) \cup \mu(c_1\overline{c_3}) = \{\overline{c_1}c_2\overline{c_3}, \overline{c_1}c_2c_3, c_1\overline{c_2}\overline{c_3}, c_1c_2\overline{c_3}, c_1c_2c_3\}$$

Now, consider the following set of ordered minterms along with their authorizations indicated in the respective brackets:

$$c_1\overline{c_2}\overline{c_3}(\texttt{PERMIT}) < c_1\overline{c_2}c_3(\texttt{DENY}) < c_1c_2c_3(\texttt{PERMIT})$$

Observe that the condition minterm $c_1\overline{c_2}c_3$ is not permitted. Thus, the set of minterms indicated above is non-convex. Hence, based on Theorem 4.6(1), the given ABAC policy cannot be represented in the DDDO model.

7 DISCUSSIONS

7.1 Supporting Disjunction Operator in Conditional Expressions

In order to provide a concise framework of rule-based models (Section 2), we consider the semantics of a ruleset as a disjunction of rules and only consider the conjunction operator in the conditional expressions. This is common practice in the access control literature [13, 7, 8]. Even though some rule-based models support the disjunction operator in rule expressions [12, 15, 3, 28], we note that they implicitly or explicitly derive the semantics of disjunction based on the conjunction and negation operators, and furthermore, do not support negative authorization. Here, we informally show that even in the presence of negative authorization and various conflict resolution strategies, adding the disjunction operator to conditional expressions does not result in additional expressiveness. Without loss of generality, assume that a conditional expression that supports the disjunction operator is expressed in the disjunctive normal form (DNF). Consider rule x with a DNF expression. Replace it with a set of rules Y each with one of x's conjunctive sub-expressions and the same modality as rule x. In the case of the PERMIT-overrides or DENY-overrides conflict resolution strategies, set of rules Y will result in the same set of permitted/denied minterms as would x. In the case of the first-applicable strategy, we need to ensure that all rules in Y are ordered consecutively as those rules replace x in the sequence of rules in the policy. This will ensure that set of rules Y will permit/deny the same set of minterms as would x.

7.2 Impact of Model Features on Expressiveness

Here, we briefly discuss how the proposed semantics model and expressiveness analysis approach in this paper addresses the questions that we initially posed in the introduction.

Uncomplemented and Complemented Conditions. We showed that the flexibility of supporting both complemented and uncomplemented conditions in rule expressions (as is enjoyed by the Negation model) creates the maximum expressive power. It is trivial to see that further extensions to the Negation model (e.g., support of DENY rules) would not have resulted in a change in its expressiveness. For this reason, the rest of the models that are explored in this paper only support uncomplemented conditions and vary in other aspects such as rule modality and conflict resolution. As shown in Section 5.4, all of those models except DDFA are less expressive than Negation. Therefore, we can infer that adding support for complemented conditions often increases the expressive power of a model. However, we note that all features of a model need to be considered holistically to establish its accurate expressiveness. For example, we showed that DDFA is as expressive as Negation despite not supporting complemented conditions.

PERMIT *and* DENY *Rules.* Even though we did not explicitly consider a rule-based model that only supports uncomplemented conditions and PERMIT rules, we note that the DDPO model is equivalent to such a model. Removing DENY rules from a DDPO policy would not affect its semantics as evidenced by its semantics shown in Table 1. Comparing the expressiveness of DDPO against other models with negative authorizations (see Figure 4), we can infer that support DDPO is less expressive than most of them. The only exception is DPDO which cannot express certain policies expressible in DDPO (which again shows the importance of a formal, holistic semantics model for expressiveness comparison).

Choice of Default Decision and Conflict Resolution. The expressiveness hierarchy established in Section 5.4 provides a comprehensive comparison of how different choices of default decision and conflict resolution strategy impact the expressiveness of rule-based models. One of the most surprising results for us was the expressive power of DDFA despite its seemingly simple conflict resolution strategy.

8 RELATED WORK

In this section, we review the closely related work. First, we examine the characteristics of policy models in the literature based on factors such as complemented/uncomplemented conditions, rule modality, and conflict resolution (which we utilized in Section 2 to categorize different rule-based models). Second, we examine the works that have focused on analyzing and providing theoretical results on the expressiveness of various access control models in the literature.

8.1 Policy Models

Contemporary access control models utilize the notion of conditions to enforce authorizations. In a rule-based policy model, different conditions are combined (using conjunction) to form different authorization rules. Each rule then specifies which users can access what resources depending on the satisfaction of the rule's conditions. Two of the more recent and widely studied rule-based models are attribute-based access control (ABAC) and relationship-based access control (ReBAC). ABAC Policies [29, 27, 28, 15, 10] enforce conditions on the attributes of subjects and objects to specify authorizations. On the other hand, ReBAC policies [11, 12, 7, 8, 6] utilize the concept of relationships, a path-based pattern of edge labels between subjects and objects that are checked against a graph of system entities. Previous literature on ABAC and ReBAC models have employed either uncomplemented conditions only [18, 7, 2, 23, 14] or both of complemented and uncomplemented conditions [29, 15, 11, 1, 4, 5, 10, 8, 24]. In this paper, we consider both complemented and uncomplemented conditional expressions during the categorization of policy models. Besides, we also take into account a special conditional expression denoted as $\top$ that always evaluates to *true* and thus satisfied by every access request.

When a rule is applicable to an access request, the access decision associated with the rule becomes effective. In general, the decision can be either PERMIT (positive authorization) or DENY (negative authorization). Most policy models allow rules with only PERMIT decisions and regard everything else to be denied according to the deny-by-default strategy [29, 11, 12, 1, 4, 3]. However, there are also access control systems that support both positive and negative

authorization rules [10, 7, 8, 24, 14, 2, 23, 6, 30, 5]. Therefore, in this paper, we consider both of those rule modalities.

In models that allow both rule modalities, a conflict resolution strategy is employed to determine the final access decision when multiple rules with different decisions are applicable to the same access request. The policy models in the literature that support both positive and negative authorization rules mostly employ the DENY-overrides and PERMIT-overrides strategies to prioritize the DENY decision and the PERMIT decision, respectively [8, 24]. The eXtensible Access Control Markup Language (XACML) [10], an OASIS standard for specifying attribute-based policies, consists of rule-combining algorithms which correspond to the conflict resolution strategy component in our rule-based policy model. Standard combining algorithms in XACML include first-applicable and only-one-applicable, along with PERMIT-overrides and DENY-overrides. In this paper, in order to cover a wide range of conflict resolution strategies from the literature, we consider the DENY-overrides, PERMIT-overrides, and first-applicable strategies.

8.2 Analysis of Policies and Policy Models

We can roughly categorize the literature on policy analysis into two categories: those analyzing policies in the context of a given access control model, and those focusing on analyzing models as a whole such as expressiveness analysis. The work presented in this paper is closer to the latter category as it enables us to analyze the expressive power of access control models. However, note that the proposed semantics model in this paper can be used for individual policy analysis as well.

The former category involves either analysis of static policies (e.g., the problem of combining policies [25] or verifying policy properties [22, 32]) or verification of dynamic properties of policies. For example, the well-studied *safety problem* [13] determines whether an access control system can reach an unsafe state in which a presumably untrusted subject has access to a certain object. More recent works have considered the problem of *security analysis* [19, 21, 20, 26] which is a generalization of safety analysis and considers, along with safety analysis, other analysis problems such as availability, bounded safety, containment, etc.

The latter category (i.e., policy model analysis) is closer to the proposed work in this paper. In this category, some work show that policies expressed in older models in the literature can be expressed in the newly proposed model in order to demonstrate the general applicability of the proposed model. For example, Jin et al. [15] show that their proposed attribute-based access control model, $ABAC_\alpha$, can sufficiently express discretionary access control, mandatory access control, and role-based access control policies. Unlike such a targeted approach, in this paper, we present a general framework that allows comparison of semantics and expressiveness among a wide range of policy models. Our work is also distinguishable from those that target very specific expressiveness problems within a particular model. For example, Joshi et al. [16] analyze the expressiveness of the temporal constraints proposed for the generalized temporal role-based access control model [17] and conclude that there is a minimal set of those constraints that can sufficiently express all. As a general framework for comparing the expressive power of access control models, Tripunitara and Li [31] propose a theory that when one scheme can represent all types of policies that another can, then the former is deemed to be at least as expressive as the latter. Their concept is based on two notions of simulations, namely reductions and state-matching reductions, which preserve security properties such as availability, mutual exclusion, and bounded safety. In fact, they consider a more dynamic version of expressiveness comparison. While in this work, we are concerned about expressive power in terms of static policies, Tripunitara and Li look into how access control schemes, considering their administrative policies, can simulate and preserve security properties from one model to another. Therefore, we are targeting a different research question. We should also note that their approach requires an extensive formulation of mappings and reduction proofs in order to establish expressiveness comparison between a pair of models. Relying on a unified semantics model, our approach has an advantage in formulating expressiveness questions and conciseness of proofs.

Crampton and Williams [9] formally analyze how decisions are combined in rule and policy combining schemes of XACML. They conclude that those schemes suffer from redundancies (as well as incompleteness). We note that our current framework of rule-based models does not support the hierarchical composition of XACML policies. Therefore, their results are not directly applicable in our context.

9 CONCLUSION

As shown in this paper, rule-based access control models can provide flexible policy constructs such as conditional expressions and choice of rule modality. A major reason for utilizing those constructs is to enable expressive policies that can adequately capture the authorization semantics of the target applications. However, as we showed in this paper different combinations of policy constructs can produce different expressiveness results. In this paper, we proposed a formal policy semantics model and associated theory that capture semantics of policies as sets of *condition minterms*. In addition to being useful for comparing individual policies, the proposed approach can be used for reasoning about the expressiveness of a policy model as a whole. We studied six particular rule-based models that represented the various combinations of model constructs in the context of our theory. We showed that the Negation model (allowing complemented conditions, but only PERMIT rules), as well as DDFA (allowing only uncomplemented conditions, both PERMIT and DENY rules, with the first-applicable conflict resolution strategy) were both capable of expressing any policy semantics. Other models such as DDDO (similar to DDFA but with DENY-overrides conflict resolution) have less expressive power. In particular, DDDO can express all and only policies that are semantically characterized by convex-shaped sets of condition minterms.

We are excited about several directions of future work. A research question that we have started studying is whether a given policy instance of a more expressive model is expressible in a less expressive model. Answering that question is important, for example, when considering migrating to a new environment and wondering if the target policy model is sufficiently expressive for a given policy instance. We note that a policy instance may not necessarily use

the full expressive power of a model. Therefore, such migrations could be practical without sacrificing the semantics of the given policy. In the future, we will also explore policy constructs that are not currently supported by the framework in this paper such as hierarchical policies.

ACKNOWLEDGMENTS

We are thankful to the anonymous reviewers for their feedback that helped us improve the paper.

REFERENCES

[1] G. Bruns, P. Fong, I. Siahaan, and M. Huth. 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*, CODASPY '12, pages 117–124, New York, NY, USA. ACM, 2012.

[2] B. Carminati, E. Ferrari, R. Heatherly, M. Kantarcioglu, and B. Thuraisingham. Semantic web-based social network access control. *Computers Security*, 30(2):108–115, Mar. 2011. ISSN: 01674048.

[3] Y. Cheng, J. Park, and R. Sandhu. An access control model for online social networks using user-to-user relationships. *IEEE Transactions on Dependable and Secure Computing*, 13(4):424–436, July 2016. ISSN: 1545-5971.

[4] Y. Cheng, J. Park, and R. Sandhu. Relationship-based access control for online social networks: beyond user-to-user relationships. In *Proc. 2012 International Conference on Privacy, Security, Risk and Trust and 2012 International Confernece on Social Computing*, pages 646–655, Sept. 2012.

[5] J. Crampton and C. Morisset. PTaCL: a language for attribute-based access control in open systems. In *International Conference on Principles of Security and Trust*, pages 390–409. Springer, 2012.

[6] J. Crampton and J. Sellwood. ARPPM: administration in the RPPM model. In *Proceedings of the Sixth ACM Conference on Data and Application Security and Privacy*, CODASPY '16, pages 219–230, New York, NY, USA. ACM, 2016.

[7] J. Crampton and J. Sellwood. Path conditions and principal matching: a new approach to access control. In *Proceedings of the 19th ACM Symposium on Access Control Models and Technologies*, SACMAT '14, pages 187–198, New York, NY, USA. ACM, 2014.

[8] J. Crampton and J. Sellwood. Relationships, Paths and Principal Matching: A New Approach to Access Control. arXiv:1505.07945 [cs], May 29, 2015. arxiv:1505.07945;

[9] J. Crampton and C. Williams. On completeness in languages for attribute-based access control. In *Proceedings of the 21st ACM on Symposium on Access Control Models and Technologies*, SACMAT '16, pages 149–160, New York, NY, USA. ACM, 2016.

[10] eXtensible Access Control Markup Language (XACML) Version 3.0, OASIS, 2013.

[11] P. W. Fong. Relationship-based access control: protection model and policy language. In *Proc. CODASPY '11*, pages 191–202, San Antonio, TX, USA. ACM, 2011.

[12] P. W. Fong and I. Siahaan. Relationship-based access control policies and their policy languages. In *Proc. 16th ACM Symposium on Access Control Models and Technologies*, SACMAT '11, pages 51–60, Innsbruck, Austria. ACM, 2011.

[13] M. A. Harrison, W. L. Ruzzo, and J. D. Ullman. Protection in operating systems. *Commun. ACM*, 19(8):461–471, Aug. 1976. ISSN: 0001-0782.

[14] H. Hu and G.-J. Ahn. Multiparty authorization framework for data sharing in online social networks. In Y. Li, editor, *Proceedings of the 25th annual IFIP WG 11.3 conference on Data and applications security and privacy*, volume 6818 of *Lecture Notes in Computer Science*, pages 29–43. Springer Berlin / Heidelberg, 2011.

[15] X. Jin, R. Krishnan, and R. Sandhu. A unified attribute-based access control model covering DAC, MAC and RBAC. In *Data and Applications Security and Privacy XXVI*. IFIP Annual Conference on Data and Applications Security and Privacy, Lecture Notes in Computer Science, pages 41–55. Springer, Berlin, Heidelberg, July 11, 2012.

[16] J. B. D. Joshi, E. Bertino, and A. Ghafoor. An analysis of expressiveness and design issues for the generalized temporal role-based access control model. *IEEE Transactions on Dependable and Secure Computing*, 2(2):157–175, 2005.

[17] J. B. D. Joshi, E. Bertino, U. Latif, and A. Ghafoor. A generalized temporal role-based access control model. *IEEE Transactions on Knowledge and Data Engineering*, 17(1):4–23, 2005. ISSN: 1041-4347.

[18] S. Kandala, R. Sandhu, and V. Bhamidipati. An attribute based framework for risk-adaptive access control models. In *2011 Sixth International Conference on Availability, Reliability and Security*, pages 236–241. IEEE, 2011.

[19] N. Li, J. C. Mitchell, and W. H. Winsborough. Design of a role-based trust-management framework. In *Proceedings 2002 IEEE Symposium on Security and Privacy*, pages 114–130. IEEE, 2002.

[20] N. Li and M. V. Tripunitara. Security analysis in role-based access control. *ACM Trans. Inf. Syst. Secur.*, 9(4):391–420, Nov. 2006. ISSN: 1094-9224.

[21] N. Li and W. H. Winsborough. Beyond proof-of-compliance: safety and availability analysis in trust management. In *Proceedings of the 2003 Symposium on Security and Privacy*, pages 123–139, May 2003.

[22] D. Lin, P. Rao, E. Bertino, N. Li, and J. Lobo. EXAM: a comprehensive environment for the analysis of access control policies. *International Journal of Information Security*, 9(4):253–273, Aug. 2010. ISSN: 1615-5262.

[23] A. Masoumzadeh and J. Joshi. OSNAC: an ontology-based access control model for social networking systems. In *Proc. 2nd IEEE Int'l Conference on Information Privacy, Security, Risk and Trust (PASSAT 2010)*, pages 751–759, Minneapolis, MN, USA, Aug. 2010.

[24] E. Pasarella and J. Lobo. A datalog framework for modeling relationship-based access control policies. In *Proceedings of the 22Nd ACM on Symposium on Access Control Models and Technologies*, pages 91–102, New York, NY, USA. ACM, 2017.

[25] P. Rao, D. Lin, E. Bertino, N. Li, and J. Lobo. An algebra for fine-grained integration of XACML policies. In *Proceedings of the 14th ACM symposium on Access control models and technologies*, SACMAT '09, pages 63–72, New York, NY, USA. Association for Computing Machinery, June 3, 2009.

[26] A. Sasturkar, P. Yang, S. D. Stoller, and C. R. Ramakrishnan. Policy analysis for administrative role based access control. In *Proceedings of the 19th IEEE Computer Security Foundations Workshop (CSFW'06)*, 13 pp.–138, 2006.

[27] D. Servos and S. L. Osborn. Current research and open problems in attribute-based access control. *ACM Comput. Surv.*, 49(4):65:1–65:45, Jan. 2017. ISSN: 0360-0300.

[28] D. Servos and S. L. Osborn. HGAA: an architecture to support hierarchical group and attribute-based access control. In *Proceedings of the Third ACM Workshop on Attribute-Based Access Control*, ABAC'18, pages 1–12, New York, NY, USA. ACM, 2018.

[29] D. Servos and S. L. Osborn. HGABAC: towards a formal model of hierarchical attribute-based access control. In *Foundations and Practice of Security*. International Symposium on Foundations and Practice of Security, Lecture Notes in Computer Science, pages 187–204. Springer, Cham, Nov. 3, 2014.

[30] S. D. Stoller. An administrative model for relationship-based access control. In *SpringerLink*. IFIP Annual Conference on Data and Applications Security and Privacy, pages 53–68. Springer, Cham, July 13, 2015.

[31] M. V. Tripunitara and N. Li. A theory for comparing the expressive power of access control models. *Journal of Computer Security*, 15(2):231–272, Jan. 1, 2007. ISSN: 0926-227X.

[32] F. Turkmen, J. den Hartog, S. Ranise, and N. Zannone. Formal analysis of XACML policies using SMT. *Computers & Security*, 66:185–203, May 1, 2017. ISSN: 0167-4048.