On Missing Attributes in Access Control: Non-deterministic and Probabilistic Attribute Retrieval[*]

Jason Crampton
Royal Holloway, University of London
jason.crampton@rhul.ac.uk

Charles Morisset
Newcastle University
charles.morisset@ncl.ac.uk

Nicola Zannone
Eindhoven University of Technology
n.zannone@tue.nl

ABSTRACT

Attribute Based Access Control (ABAC) is becoming the reference model for the specification and evaluation of access control policies. In ABAC policies and access requests are defined in terms of pairs attribute names/values. The applicability of an ABAC policy to a request is determined by matching the attributes in the request with the attributes in the policy. Some languages supporting ABAC, such as PTaCL or XACML 3.0, take into account the possibility that some attributes values might not be correctly retrieved when the request is evaluated, and use complex decisions, usually describing all possible evaluation outcomes, to account for missing attributes. In this paper, we argue that the problem of missing attributes in ABAC can be seen as a non-deterministic attribute retrieval process, and we show that the current evaluation mechanism in PTaCL or XACML can return a complex decision that does not necessarily match with the actual possible outcomes. This, however, is problematic for the enforcing mechanism, which needs to resolve the complex decision into a conclusive one. We propose a new evaluation mechanism, explicitly based on non-deterministic attribute retrieval for a given request. We extend this mechanism to probabilistic attribute retrieval and implement a probabilistic policy evaluation mechanism for PTaCL in PRISM, a probabilistic model-checker.

Categories and Subject Descriptors

D.4.6 [**Security and Protection**]: Access Controls; D.2.4 [**Software Software/Program Verification**]: Model checking

[*]This work has been partially funded by the EPSRC/GCHQ funded project ChAISe (EP/K006568/1), the project "Data-Driven Model-Based Decision-Making", part of the NSA funded Centre on Science of Security at University of Illinois at Urbana-Champaign, the EDA project IN4STARS2.0 and by the Dutch national program COMMIT under the THeCS project.

General Terms

Security, Theory

Keywords

Policy evaluation; missing attribute; probabilistic model-checking; PTaCL

1. INTRODUCTION

In recent years there has been considerable interest in attribute-based access control (ABAC), resulting in the development of languages such as XACML [11] and PTaCL [5]. Such languages have moved away from the "classical" view of access control, which was based on (authenticated) users and their respective identities. In the classical view, an access request was modelled as a triple (s, o, a), where s denoted a subject (corresponding to a user identity), o denoted an object (corresponding to the identity of some protected resource) and a denoted an access mode (such as read or write). In ABAC, an access request is modelled as a collection of attribute name-value pairs. ABAC is particularly suitable in "open" computing environments where the user population is not known in advance and access is allowed or denied on the basis of user characteristics, rather than identities.

One problem that arises in ABAC is that a request may not present all the relevant information to the policy decision point (PDP) and different decisions may be generated, depending on the information that is presented. Equally, an ABAC policy may not be able to produce a conclusive decision (an allow or deny) for a given request because the policy is under-specified. Conversely, an ABAC policy may be over-specified and different components of the policy may both allow and deny the request. Finally, a request may be malformed and policy evaluation may fail unexpectedly. Thus, the PDP may return an inconclusive result, indeterminate results, or inconsistent results. XACML and PTaCL handle such possibilities by extending the set of possible decisions that the PDP may return and allowing the PDP to return a subset of that decision set. XACML, for example, introduces the "not-applicable" and "indeterminate" decisions: the former is returned when the policy does not evaluate to any conclusive decision; the latter is used to indicate some error or inconsistency occurred during policy evaluation. PTaCL uses the equivalent of a "not-applicable", but models evaluation errors by returning a set of possible decisions that might have been returned if no errors had occurred.

However, the policy enforcement point (PEP) must, ultimately, take one of two actions, either allowing or denying an access request. In this paper we investigate whether the PEP should (or can) rely on the decision(s) returned by the PDPs defined for XACML and PTaCL. In particular, we focus on the case where inconclusive decisions are sets of possible decisions, which may be generated by considering attributes that may not have been included in the request (perhaps because the requester deliberately withheld them). We will show that the set of possible decisions returned by the PDP is not always meaningful and, therefore, the PEP should not rely on it. We also show that, under certain conditions, the PEP should not even rely on a single conclusive decision returned by the PDP. Of course, this raises the question of why existing PDPs are designed in the way they are. Thus, we establish a new way of thinking about request evaluation and alternative designs for PDPs in ABAC systems. In particular, we make the following contributions.

- We first show that the decision sets returned by a PTaCL/XACML policy do not necessarily correspond with an intuitive interpretation of what those decisions mean.
- We then propose a declarative evaluation mechanism for PTaCL which matches this intuition. The mechanism is based on a *non-deterministic* evaluation [14], which simulates the non-determinism of retrieving the attributes forming the request: if a value for an attribute is missing from a request, we do not know whether it should be in it or not[1].
- Finally, we extend this non-deterministic evaluation to a probabilistic one, and we show how they can be mixed.
- The concepts presented in this paper are supported by an automatic translation of PTaCL policies into PRISM, which is a probabilistic model-checker.

In the next section we briefly review related work and summarize the PTaCL language. In Section 3, we introduce a new way of reasoning about requests in the presence of uncertainty about the inclusion of attributes. We then consider non-deterministic and probabilistic attribute retrieval in Sections 4 and 5, respectively. We conclude with a summary of our contributions and some ideas for future work.

2. BACKGROUND AND RELATED WORK

Our work relies clearly on the PTaCL language [5] together with the definition of the ATRAP tool [7], which automatically analyses the safety of PTaCL policies. The notion of reducing complex decisions to simple, conclusive ones is also addressed in recent work [10], which focuses on decisions and operators, whereas we focus here on attribute retrieval. In terms of methodology, our approach follows that of Tschantz and Krishnamurthi [14], since we first establish some requirement for an access control evaluation mechanism, and we then analyse an existing language against those requirements.

Model-checking has been used in the past for access control, for instance Zhang et al. [18] propose a tool checking whether a particular goal can be reached within an access control policy; Fistler et al. [6] defined the tool Margrave, which can analyse role-based access-control policies; more recently, Ranise et al. [12, 13] have used model checking to analyse the safety problem with administrative policies. In this work, we mostly focus on the attribute retrieval problem rather than on the policy evaluation/analysis problem (although they are quite related). To the best of our knowledge, we are the first to investigate probabilistic attribute retrieval in access control.

In the remainder of this section, we recall the language PTaCL [5], after a brief introduction to 3-valued logic, to establish the notations used throughout the paper.

2.1 3-valued Logic

The truth values in Boolean logic are 0 and 1, where 1 represents *true* and 0 represents *false*; 3-valued logic extends it by considering an additional value $\bot$ [8]. There can be multiple interpretations of this extra symbol, for instance, the *weak conjunction* and *weak disjunction* operators, defined in Fig. 1 by $\sqcap$ and $\sqcup$, respectively, consider $\bot$ as absorbing; on the other hand, the *strong* conjunction and disjunction operators, defined by $\tilde{\sqcap}$ and $\tilde{\sqcup}$ consider $\bot$ as being either 1 or 0, and therefore try to "resolve" $\bot$ as much as possible. Another interpretation is to "ignore" the symbol $\bot$ as much as possible, for instance with the operators $\triangledown$ and $\vartriangle$, which correspond to the XACML operators permit-overrides and deny-overrides, respectively. Finally, we also consider the negation operator $\neg$, where $\neg 1 = 0$, $\neg 0 = 1$ and $\neg\bot = \bot$; and the "weakening" operator $\sim$, where $\sim\bot = \sim 0 = 0$ and $\sim 1 = 1$.

2.2 PTaCL

PTaCL is attribute-based, which means that a request is a set of attribute name-value pairs $\{(n_1, v_1), \ldots, (n_k, v_l)\}$. For instance, in a healthcare context, the request $\{(\mathbf{r}, \mathsf{nurse}), (\mathbf{emg}, \mathit{true})\}$ represents a request made by a nurse during an emergency. In addition, PTaCL uses policy targets [1, 2, 4, 11, 17], which specify the requests to which the policy is applicable.

- An *atomic target* is a pair (n, v), where n is an attribute name and v is an attribute value.
- A *composite target* has the form $\mathsf{op}(t_1, \ldots, t_n)$, where op represents an n-ary 3-valued logical operator. For the sake of simplicity, we focus here on the unary and binary operators defined in Fig. 1.

Given a request, a target evaluates to a single value in $\{1, 0, \bot\}$, intuitively indicating if the request matches, does not match, or does not contain the attributes required to evaluate its applicability, respectively. More formally, the semantics of an atomic target (n, v) for a request $q = \{(n_1, v_1), \cdots, (n_k, v_l)\}$ is given as:

$$[\![(n, v)]\!]_{\mathrm{P}}(q) = \begin{cases} 1 & \text{if } (n, v') \in q \text{ and } v = v', \\ \bot & \text{if } (n, v') \notin q, \\ 0 & \text{otherwise.} \end{cases}$$

Composite targets are inductively evaluated by applying the operator to the result of the evaluation of the sub-targets. Note that the unary operator $\sim$ deals with the absence of an attribute as if the attribute does not match the value[2].

An *authorisation policy* can be:

[1]We assume non-forgeability of attribute values: if a value for an attribute belongs to a request, then it is genuine.

[2]In other words, PTaCL expects all attributes to be present by default, in XACML terminology, but the "indetermi-

Table 1: Binary operators on the set $\{1, 0, \bot\}$

.

d_1	d_2	$d_1 \tilde{\sqcap} d_2$	$d_1 \sqcap d_2$	$d_1 \vartriangle d_2$	$d_1 \tilde{\sqcup} d_2$	$d_1 \sqcup d_2$	$d_1 \triangledown d_2$
1	1	1	1	1	1	1	1
1	0	0	0	0	1	1	1
1	$\bot$	$\bot$	$\bot$	1	1	$\bot$	1
0	1	0	0	0	1	1	1
0	0	0	0	0	0	0	0
0	$\bot$	0	$\bot$	0	$\bot$	$\bot$	0
$\bot$	1	$\bot$	$\bot$	1	1	$\bot$	1
$\bot$	0	0	$\bot$	0	$\bot$	$\bot$	0
$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$

- a single decision, i.e., either 1 (allow) or 0 (deny);
- a targeted policy (t, p), where t is a target;
- a composite policy $\mathsf{op}(p_1, \dots, p_n)$, where op is a n-ary operator. Here again, we focus on the operators defined in Fig. 1.

In general, for a given request, an attribute can be completely missing (for instance, an visitor might not have any official role in a hospital), have exactly one value, or have multiple values (for instance, a nurse might be training as a physician, and in some contexts, activate both roles). However, because it is not necessarily known in advance the number of values an attribute can take for a particular request, it is impossible to know whether some values have been removed or not. PTaCL handles such situations by considering that if the target of a policy (t, p) evaluates to $\bot$, p must evaluate as if the target evaluates to both 1 and 0. More formally, the evaluation of a targeted policy (t, p) for a request q is given by:

$$[\![(t,p)]\!]_{\mathrm{P}}(q) = \begin{cases} [\![p]\!]_{\mathrm{P}}(q) & \text{if } [\![t]\!]_{\mathrm{P}}(q) = 1, \\ \{\bot\} & \text{if } [\![t]\!]_{\mathrm{P}}(q) = 0, \\ \{\bot\} \cup [\![p]\!]_{\mathrm{P}}(q) & \text{otherwise.} \end{cases}$$

where $\bot$ represents the not-applicable decision, $[\![p]\!]_{\mathrm{P}}(q) = 1$ if p is the authorisation policy 1 (allow) and $[\![p]\!]_{\mathrm{P}}(q) = 0$ if p is 0 (deny). It is worth emphasising that even though the evaluation of both targets and policies uses the set $\{1, 0, \bot\}$, these values have a different interpretation in each case: they stand for "match", "non-match" and "indeterminate" when evaluating targets, and for "allow", "deny" and "not-applicable" when evaluating policies.

In order to illustrate PTaCL, we define the following policy for an electronic health record: a physician can access it; a nurse can access it if there is an emergency; but in all cases, it cannot be accessed if the requester has a conflict of interest (e.g., the physician is a relative of the patient). These three policies correspond to the PTaCL policies p_d, p_e and p_c defined below. The global policy formed by combining p_d, p_e and p_c is defined by p_1.

$$\begin{aligned} p_d &= ((\mathbf{r}, \mathsf{phys}), 1) \\ p_e &= ((\mathbf{r}, \mathsf{nurse}) \tilde{\sqcap} \sim(\mathbf{emg}, \mathit{true}), 1) \\ p_c &= (\sim(\mathbf{cf}, \mathit{true}), 0) \\ p_1 &= (p_d \triangledown p_e) \vartriangle p_c \end{aligned}$$

The evaluation of these policies for different requests is given in Table 2, where each row contains a request q, and the evaluation of $[\![p]\!]_{\mathrm{P}}(q)$, for $p \in \{p_d, p_e, p_c, p_1\}$.

nate" value $\bot$ can always be transformed into the non-match value 0.

Table 2: Policy evaluation with $[\![\cdot]\!]_{\mathrm{P}}$.

	Policy			
Request	p_d	p_e	p_c	p_1
$\emptyset$	$\{1, \bot\}$	$\{\bot\}$	$\{\bot\}$	$\{1, \bot\}$
$\{(\mathbf{r}, \mathsf{phys})\}$	$\{1\}$	$\{\bot\}$	$\{\bot\}$	$\{1\}$
$\{(\mathbf{r}, \mathsf{phys}), (\mathbf{cf}, \mathit{true})\}$	$\{1\}$	$\{\bot\}$	$\{0\}$	$\{0\}$
$\{(\mathbf{r}, \mathsf{nurse})\}$	$\{\bot\}$	$\{\bot\}$	$\{\bot\}$	$\{\bot\}$
$\{(\mathbf{r}, \mathsf{nurse}), (\mathbf{emg}, \mathit{true})\}$	$\{\bot\}$	$\{1\}$	$\{\bot\}$	$\{1\}$

3. ACCESS CONTROL WITH INCOMPLETE REQUESTS

Thus, ABAC introduces the possibility that some attributes might be missing from a request. Moreover, missing attributes cannot always be automatically detected. PTaCL addresses this possibility by letting targets evaluate to $\bot$ when an attribute required by the target is missing, which, in turn, can cause a policy to return multiple values.

In this section, we first introduce some general characterisation of requests, after which we establish some requirements for policy evaluation mechanisms. Then, we show that PTaCL does not necessarily meet them and propose an abstract mechanism satisfying them.

3.1 Characterising Missing Information

The major issue with missing information is that it is not necessarily syntactically detectable. Indeed, some attributes might simply not exist in a particular context, or the number of values expected for a given attribute is not necessarily known in advance. For instance, the request $\{(\mathbf{r}, \mathsf{nurse})\}$ has all required information if the subject submitting is only a nurse, but is missing information if she has multiple roles. Similarly, the empty request is missing information if the subject has a role, but is not if the subject is not member of the hospital staff.

Tschantz and Krishnamurthi [14] introduce an ordering relation $\sqsubseteq$ over requests, such that if $q \sqsubseteq q'$, *"then q' contains all the information contained in q and possibly more"*. This relation is used to define policy safety, such that a policy p is safe when, given two requests q and q', if $q \sqsubseteq_P q'$, then $[\![p]\!](q) \leq [\![p]\!](q')$. For the sake of generality, they deliberately under-specify the ordering relations over requests, but claim that *"informally, in a safe language, undue access is impossible provided that requests tell no lies; whereas, in an unsafe language, the requests must additionally tell the whole truth."*

3.2 Well-formed and complete requests

Given an attribute n, a value v and a request q, three cases are possible:

1. it is certain that n has the value v in q;
2. it is certain that n does not have the value v in q;
3. we do not know whether n has the value v in q or not.

In this work, we assume attributes are unforgeable, and therefore, if $(n, v) \in q$, we are in the first case. However, the main idea behind ABAC with incomplete information is to say that if $(n, v) \notin q$, we could be either in the second or the third case. This is typically handled in PTaCL and XACML at the policy level, by considering indeterminate target evaluations as non-matching ones. However, we know that with these approaches, policies are in general unsafe.

We propose here to address this problem at the request level. More precisely, we suggest to consider *negative* attribute values $\overline{v}$, such that for any values v and v', $\overline{v} \neq v'$. Intuitively, a negative attribute value $\overline{v}$ explicitly indicates that an attribute cannot have value v in a given context. In order to avoid any contradiction, we say that a request q is *well-formed* if it does not contain both (n, v) and $(n, \overline{v})$, for any attribute n and any value v, and in this case we write $\mathsf{wf}(q)$.

In other words, the negative value $\overline{v}$ never matches any atomic target, but ensures that the "positive" value v cannot be added to the request. We can now define the three cases above as:

1. (n, v) belongs to q;
2. $(n, \overline{v})$ belongs to q;
3. neither (n, v) nor $(n, \overline{v})$ belong to q.

Intuitively, in the third case, the value v for n has not been retrieved for q yet. Hence, q could either correspond to $q \cup \{(n, v)\}$ or to $q \cup \{(n, \overline{v})\}$. This leads to the idea of *non-deterministic attribute retrieval*: the value for the attribute must be retrieved, but we do not know whether it is going to be positive or negative.

We say that a request q is *complete* when, for any attribute n and any value v, either $(n, v) \in q$ or $(n, \overline{v}) \in q$. In this case, adding a new value to q would create request that is not well-formed. In addition, given a request q, we write $\overline{q}$ for the request where we add the negative values when the positive value is not already present:

$$\overline{q} = q \cup \{(n, \overline{v}) \mid (n, v) \notin q\}$$

Clearly, $\overline{q}$ is complete, and well-formed if q is well-formed.

3.3 Requirements

We are now in position to establish some intuitive requirements for a general evaluation function $[\![\cdot]\!]$, such that given a policy p and a request q, $[\![p]\!](q)$ returns a set of decisions. The first requirement expresses the fact that any decision returned for a request in which information is missing corresponds to a decision that would be returned had the missing information been provided.

REQUIREMENT 1. *For any request q, if $d \in [\![p]\!](q)$, then there exists a well-formed request $q' \supseteq q$ such that $[\![p]\!](q') = \{d\}$.*

The second requirement is the converse of the first one, and expresses that if a decision can be returned when all information in a request is provided, then evaluating the same request but with missing information should at least return that decision.

REQUIREMENT 2. *Given a request q and a decision d, if there exists a well-formed request $q' \supseteq q$ such that $[\![p]\!](q') = \{d\}$, then $d \in [\![p]\!](q)$.*

Intuitively, these two requirements could correspond to the notions of correctness and completeness, respectively. Indeed, Requirement 1 implies that no incorrect decision can be returned when information is missing, i.e., no decision that could not have been returned had all information been provided. A trivial evaluation mechanism satisfying this requirement is one that always returns the empty set of decisions for any request (i.e., a mechanism that never returns a incorrect decision). Conversely, Requirement 2 implies that *all* the correct decisions are returned when information is missing (and potentially more). A trivial mechanism satisfying this requirement is that returning all possible decisions for any query.

Suppose the PEP receives the decisions $\{d_1, \ldots, d_n\}$ from the PDP. If the PEP knows that the PDP is meeting these two requirements, then it can deduce that all d_i are potentially correct decisions, had all information been provided, and that any decision different from any d_i is not a potentially correct decision. It can therefore reduce its choice to selecting a decision in $\{d_1, \ldots, d_n\}$. In particular, when the set returned by the PDP is reduced to a unique decision, then the PEP can be certain that this decision is the correct one.

3.4 PTaCL Analysis

We now show that PTaCL satisfies neither Requirement 1 nor Requirement 2.

Firstly, consider the policy $p_3 = ((\mathbf{r}, \mathsf{nurse}), 1) \vartriangle ((\mathbf{r}, \mathsf{nurse}), 0)$, where $\vartriangle$ stands for the deny-overrides operator. If we evaluate for the empty request, we have

$$\begin{aligned} [\![(\mathbf{r}, \mathsf{nurse}), 1)]\!]_{\mathrm{P}}(\emptyset) &= \{1, \bot\} \\ [\![(\mathbf{r}, \mathsf{nurse}), 0)]\!]_{\mathrm{P}}(\emptyset) &= \{0, \bot\} \\ [\![p_3]\!]_{\mathrm{P}}(\emptyset) &= \{1, 0, \bot\}\,. \end{aligned}$$

However, it is easy to see that there is no request q such that $[\![p_3]\!]_{\mathrm{P}}(q) = \{1\}$: if $((\mathbf{r}, \mathsf{nurse}), 1)$ evaluates to 1, then $((\mathbf{r}, \mathsf{nurse}), 0)$ necessarily evaluates to 0. As these policies are combined in p_3 using a deny-overrides operator, even though $1 \in [\![p_3]\!]_{\mathrm{P}}(q)$, there is no request $q' \supseteq q$ such that $[\![p_3]\!](q') = \{1\}$. Thus, PTaCL does not satisfy Requirement 1.

Let us now consider the policy p_1, defined in Section 2.2, the requests $q = \{(\mathbf{r}, \mathsf{phys})\}$ and $q' = \{(\mathbf{r}, \mathsf{phys}), (\mathbf{cf}, \mathit{true})\}$. As described before, we have $[\![p_1]\!]_{\mathrm{P}}(q) = \{1\}$, while $[\![p_1]\!]_{\mathrm{P}}(q') = \{0\}$. In other words, the decision returned by the extended request q' does not appear in the set returned by the request q, and we can conclude that PTaCL does not satisfy Requirement 2. Note that this kind of situation is described in [5] as an *attribute hiding attack*, where an attacker can gain some advantage by hiding information.

In other words, the decision set received by an access control resolver is not necessarily helpful to make a conclusive decision, and it is worth observing that these two observations also hold for XACML 3.0. One possibility is to constrain the policy language: if the policy is constructed using some specific constraints, then some monotonicity results can be shown [5]. In this paper, we propose a new approach by designing an evaluation function explicitly relying on the non-determinism of the attribute requests.

4. NON-DETERMINISTIC RETRIEVAL

We now introduce a new method for computing the set of decisions returned by the evaluation of a PTaCL policy (t, p) for a request q. Informally, we remove any indeterminism from the evaluation of targets, instead evaluating a set of associated requests (specifically those that are extensions of q). We therefore present two new abstract evaluation functions: $[\![\cdot]\!]_{\mathrm{C}}$, which evaluates a request without considering missing attributes, and $[\![\cdot]\!]_{\mathrm{N}}$, which evaluates a request by considering all possible extensions, and suggest that the function $[\![\cdot]\!]_{\mathrm{N}}$ should be used instead of the function $[\![\cdot]\!]_{\mathrm{P}}$ defined above. We then explain how to use the PRISM model-checker to compute $[\![\cdot]\!]_{\mathrm{N}}$.

4.1 Abstract evaluation

Intuitively, we define $[\![\cdot]\!]_{\mathrm{C}}$ such that the evaluation of a target is either 0 or 1, where 1 is returned if there exists a matching attribute and 0 is returned otherwise. More formally, we first define an evaluation function $[\![\cdot]\!]_{\mathrm{C}}$ as follow:

$$[\![(n,v)]\!]_{\mathrm{C}}(q) = \begin{cases} 1 & \text{if } (n,v') \in q \text{ and } v = v', \\ 0 & \text{otherwise;} \end{cases}$$

and composite targets are evaluated in a similar fashion than with $[\![\cdot]\!]_{\mathrm{P}}$. The evaluation of a request with respect to a policy p guarded by a target t is $\perp$ if the evaluation of the target is 0 (that is, the policy is not applicable to the request), otherwise the result of evaluating p is returned.

$$[\![(t,p)]\!]_{\mathrm{C}}(q) = \begin{cases} [\![p]\!]_{\mathrm{C}}(q) & \text{if } [\![t]\!]_{\mathrm{C}}(q) = 1, \\ \perp & \text{otherwise.} \end{cases}$$

Composite policies are evaluated by applying the corresponding operators to the results of evaluating their respective operands. We can observe that for any policy p and any well-formed request q, we have $[\![p]\!]_{\mathrm{C}}(q) = [\![p]\!]_{\mathrm{C}}(\overline{q})$. In other words, $[\![\cdot]\!]_{\mathrm{C}}$ corresponds to the evaluation of the query assuming it is complete and that any value not explicitly provided is not in the request.

Given a request q, we define the set of extensions to q, denoted by $Ext(q)$ to be $\{q' \in Q \mid q \subseteq q' \wedge \mathsf{wf}(q')\}$. For the sake of simplicity, we assume here that for a given request q, $Ext(q)$ is finite, and we leave for future work the study of infinite sets of request extensions. Then

$$[\![p]\!]_{\mathrm{N}}(q) = \{[\![p]\!]_{\mathrm{C}}(q') : q' \in Ext(q)\}.$$

Thus, we have non-determinism in request evaluation, as with the original evaluation method used in PTaCL. However, the non-determinism now arises because we consider all possible related requests and the decisions associated with those requests.

Table 3 illustrates the evaluation of $[\![\cdot]\!]_{\mathrm{N}}$ for the policies defined in Section 2.2, and the same requests than those in Table 2. Clearly, $[\![\cdot]\!]_{\mathrm{N}}$ introduces much more indeterminism in the evaluation, for instance the request $\{(\mathbf{r}, \mathsf{phys})\}$ now evaluates to $\{1,0\}$, instead of $\{1\}$ according to $[\![\cdot]\!]_{\mathrm{P}}$, since if the attribute value $(\mathbf{cf}, \mathit{true})$ is missing, this request would evaluate to $\{0\}$. However, when a conclusive decision is returned, for instance with the request $\{(\mathbf{r}, \mathsf{phys}), (\mathbf{cf}, \mathit{true})\}$, then there is no doubt that this is the only possible decision.

On the other hand, if we evaluate the policy p_3 defined in Section 3.4, we can see that $[\![p_3]\!]_{\mathrm{N}}(\emptyset) = \{0, \perp\}$, which corresponds to the fact that there is no request in $Ext(\emptyset)$ that evaluates to the decision 1.

In the following proposition, we show that this new evaluation method satisfies Requirements 1 and 2.

PROPOSITION 1. *For any decision d, any policy p and any request q, $d \in [\![p]\!]_{\mathrm{N}}(q)$ if and only if there exists a well formed request $q' \supseteq q$ such that $[\![p]\!]_{\mathrm{N}}(q') = \{d\}$.*

PROOF. $\Rightarrow$) Let d be a decision in $[\![p]\!]_{\mathrm{N}}(q)$. By definition of $[\![\cdot]\!]_{\mathrm{N}}$, there exists $q' \in Ext(q)$ such that $[\![p]\!]_{\mathrm{C}}(q') = d$. As observed above, $[\![p]\!]_{\mathrm{C}}(q') = [\![p]\!]_{\mathrm{C}}(\overline{q'})$. It follows that $[\![p]\!]_{\mathrm{N}}(\overline{q'}) = \{d\}$, since $Ext(\overline{q'}) = \{\overline{q}\}$. By definition, q' is well-formed, and thus so is $\overline{q'}$, and since $\overline{q'} \supseteq q$, we can conclude.

$\Leftarrow$) Let $q' \supseteq q$ be a well-formed request such that $[\![p]\!]_{\mathrm{N}}(q') = \{d\}$. Since q' is well-formed, we know that $q' \in Ext(q')$, and by definition of $[\![\cdot]\!]_{\mathrm{N}}$, it follows that $[\![p]\!]_{\mathrm{C}}(q') = d$. Since $q' \supseteq q$, we have $q' \in Ext(q)$, and we can conclude that $d \in [\![p]\!]_{\mathrm{N}}(q)$. ☐

Table 3: Policy evaluation with $[\![\cdot]\!]_{\mathrm{N}}$

	Policy			
Request	p_d	p_e	p_c	p_1
$\emptyset$	$\{1,\perp\}$	$\{1,\perp\}$	$\{0,\perp\}$	$\{1,0,\perp\}$
$\{(\mathbf{r},\mathsf{phys})\}$	$\{1\}$	$\{1,\perp\}$	$\{0,\perp\}$	$\{1,0\}$
$\{(\mathbf{r},\mathsf{phys}),(\mathbf{cf},\mathit{true})\}$	$\{1\}$	$\{1,\perp\}$	$\{0\}$	$\{0\}$
$\{(\mathbf{r},\mathsf{nurse})\}$	$\{1,\perp\}$	$\{1,\perp\}$	$\{0,\perp\}$	$\{1,0,\perp\}$
$\{(\mathbf{r},\mathsf{nurse}),(\mathbf{emg},\mathit{true})\}$	$\{1,\perp\}$	$\{1\}$	$\{0,\perp\}$	$\{1,0\}$

It is worth observing at this point that it is possible to construct $Ext(q)$ by inspection of the PTaCL policy and q. Morisset and Griesmeyer showed that it is sufficient to only consider requests comprising attribute name-value pairs that explicitly occur in the PTaCL policy [7]. In particular, it is not necessary to consider (n,v) for every possible value of v that n can take.

4.2 Evaluation using model-checking

Model-checking [3], in a nutshell, consists in(i) abstracting a system as a finite state machine, where each state s contains some atomic propositions that are true for that state; and (ii) checking whether some properties holds for this model. These properties can be *temporal*, when they are expressed over the paths of the model, the intuition being that a path represents an execution sequence.

For instance, given a path $(s_0, \ldots, s_n)$, where each s_i represents a state, and a property φ over states, $\mathsf{F}\varphi$ (also denoted as $\Diamond\varphi$ in the literature) holds if there exists i such that for any $j \geqslant i$, $\varphi(s_j)$ holds. We also use here the operator E, such that given a state s and a path property ϕ, $\mathsf{E}\phi$ holds if there exists a path from s such that ϕ holds.

Intuitively, our encoding of the function $[\![\cdot]\!]_{\mathrm{N}}$ relies on the following key points:

- each state of the model corresponds to a single request;
- each transition from one state to another corresponds to adding some (n,v) or some $(n,\overline{v})$;
- the evaluation of a policy according to $[\![\cdot]\!]_{\mathrm{C}}$ is expressed as a property over states;
- the evaluation of a policy according to $[\![\cdot]\!]_{\mathrm{N}}$ is expressed as a path property, starting from the state corresponding to request, and checking for each decision whether there exists a state for which $[\![\cdot]\!]_{\mathrm{C}}$ evaluates to that decision.

The model does not strictly depend on the policy, but only on the attribute values, and all possible requests are modelled, not only the one we want to evaluate.

More precisely, each state contains, for any attribute value (n,v), two atomic Boolean propositions, $\iota_{n,v}$ and $\alpha_{n,v}$. These propositions characterise the request q corresponding to that state:

- $\iota_{n,v}$ is false when neither (n,v) nor $(n,\overline{v})$ belongs to q;
- $\iota_{n,v}$ is true and $\alpha_{n,v}$ is false when $(n,\overline{v})$ belongs to q;
- $\iota_{n,v}$ is true and $\alpha_{n,v}$ are true when (n,v) belongs to q.

For instance, if we only consider the attribute values $(\mathbf{r}, \mathsf{phys})$ and $(\mathbf{cf}, \mathit{true})$, a state is a tuple $(\iota_{\mathbf{r},\mathsf{phys}}, \alpha_{\mathbf{r},\mathsf{phys}}, \iota_{\mathbf{cf},\mathit{true}}, \alpha_{\mathbf{cf},\mathit{true}})$, and the request $\{(\mathbf{r}, \overline{\mathsf{phys}})\}$ corresponds to the state $(\mathit{true}, \mathit{false}, \mathit{false}, \mathit{false})$, while the request $\{(\mathbf{cf}, \mathit{true})\}$ corresponds to the state $(\mathit{false}, \mathit{false}, \mathit{true}, \mathit{true})$.

The transition function is defined such that, given two states $s = (\iota_{n_1,v_1}, \alpha_{n_1,v_1}, \ldots, \iota_{n_k,v_l}, \alpha_{n_k,v_l})$ and $s' = (\iota'_{n_1,v_1}, \alpha'_{n_1,v_1}, \ldots, \iota'_{n_k,v_l}, \alpha'_{n_k,v_l})$, there is a transition from s to s' if, and only if, there exists an attribute value (n, v) such that $\iota_{n,v}$ is false and $\iota'_{n,v}$ is true ($\alpha'_{n,v}$ can be either true or false), and the propositions for all other attribute values are identical in both states. In other words, a transition corresponds to the non-deterministic retrieval of exactly one attribute value.

The evaluation of a policy according to $[\![\cdot]\!]_{\mathrm{C}}$ can be mapped directly from the request to the state using the definition in Section 4.1. Given a policy p and a decision d, we write $\delta_{p,d}$ for the predicate over states such that given a state s, $\delta_{p,d}(s)$ holds if, and only if, $[\![p]\!]_{\mathrm{C}}(q_s) = d$ holds, where q_s is the request corresponding to the state s.

Proposition 2. *Given a policy p, a decision d and a request q, $d \in [\![p]\!]_{\mathrm{N}}(q)$ if and only if the path property $\mathsf{EF}\delta_{p,d}$ holds from the state corresponding to q.*

4.3 PRISM Encoding

4.3.1 PRISM

We only present the basic PRISM elements used for our encoding, and refer the reader to the manual[3] for further details. Intuitively, a PRISM model consists of several modules. Each module can contain some variables, and describes the possible transitions at each step. A transition has the following general form:

```
[] g → p1:(post1) + ... + pn:(postn)
```

where g is a boolean expression representing the guard of the transition, and p_i is the probability that the post-condition (i.e., some conditions on the variables) post_i is selected. For instance, a very simple example of a module representing a coin toss can be defined as:

```
module coin_toss
  head : bool init false;
  [] true → 0.5:(head'=true) +  0.5:(head'=false);
endmodule
```

where (head'=**true**) is a post-condition, indicating that the value of the variable head after the transition (which is indicated by the apostrophe) is **true**. In this particular example, the coin is fair and the probability of setting head to **true** is the same than that of setting head to **false**, which effectively corresponds to obtaining tail. In general, when several transitions are enabled within a given module, one is non-deterministically selected.

It is worth observing that there is a fundamental difference between probabilistic and non-deterministic transitions. For instance, in the above example, if we trigger the coin toss multiple times, in average, we will have as many heads as tails. On the other hand, if we were to define the two following transitions:

```
  [] true → (head'=true);
  [] true → (head'=false);
```

then either transition can be triggered. This would correspond to having a coin potentially biased, for which we do not know the bias. In this paper, we first use only non-deterministic transitions, and we mix both types of transitions in the next section.

4.3.2 Attribute retrieval

We define one module for each attribute value (n, v), which encodes the propositions $(\iota_{n,v})$ and $(\alpha_{n,v})$.

```
module att_n_v
  n_v: bool init false;
  r_n_v: bool init false;

  [] !r_n_v → (n_v'=true) & (r_n_v'=true);
  [] !r_n_v → (n_v'=false) & (r_n_v'=true);
endmodule
```

The variable n_v corresponds to the proposition $\alpha_{n,v}$ and r_n_v corresponds to $\iota_{n,v}$ (the prefix r_ denotes that (n, v) has been retrieved).

In addition, the completeness of a request is encoded with the formula complete, which holds when all r_n_i_v_j are true.

4.3.3 Target and Policies

Following the definition of the function $[\![\cdot]\!]_{\mathrm{C}}$, a target evaluates to a boolean value: an atomic target (n, v) is true if and only if the variable n_v is true, and composite targets are evaluated by applying their corresponding boolean operators on the evaluation of the sub-targets.

Policies evaluate to a value in $\{0, 1, \bot\}$, and therefore we cannot encode a policy directly as a boolean formula. Instead, we encode the set $\{0, 1, \bot\}$ as the integer values 0, 1 and 2, respectively, and we adapt the logical operators accordingly. We also use the PRISM ternary operator `c ? e1 : e2` which evaluates to `e1` is `c` is true and to `e2` otherwise. Given a PTaCL policy p, we note $\widehat{p}$ the PRISM expression corresponding the encoding of the policy p, which is defined as follows:

$$\begin{aligned}
\widehat{0} &\triangleq 0 \\
\widehat{1} &\triangleq 1 \\
\widehat{\neg p} &\triangleq (\widehat{p}= 2)\ ?\ 2 : (1 - \widehat{p})) \\
\widehat{\sim p} &\triangleq \mathrm{mod}(\widehat{p},\ 2) \\
\widehat{p_1 \sqcap p_2} &\triangleq (\widehat{p_1}= 2 \mid \widehat{p_2}= 2)\ ?\ 2 : (\widehat{p_1} * \widehat{p_2}) \\
\widehat{p_1 \sqcup p_2} &\triangleq \max(\widehat{p_1},\ \widehat{p_2}) \\
\widehat{p_1 \tilde{\sqcap} p_2} &\triangleq \min(\widehat{p_1} * \widehat{p_2},\ 2) \\
\widehat{p_1 \tilde{\sqcup} p_2} &\triangleq (\widehat{p_1}= 1 \mid \widehat{p_2}= 1)\ ?\ 1 : \max(\widehat{p_1},\ \widehat{p_2}) \\
\widehat{(t, p)} &\triangleq \widehat{t}\ ?\ \widehat{p} : 2
\end{aligned}$$

For instance, we give below the PTaCL definition for the policy p_1 defined in Section 2.2, using the syntax of the tool ATRAP [7], where Ptar is the constructor for target policies, Ppov for the operator $\triangledown$, Pdov for the operator $\triangle$, Topt for the operator $\sim$ and Tstrongand for the operator $\tilde{\sqcap}$:

```
p_d : (Ptar (Tatom "role" "phys") (Patom one))
p_e : (Ptar (Tstrongand
              (Tatom "role" "nurse")
              (Topt (Tatom "emergency" "1")))
            (Patom one))
p_c : (Ptar (Topt (Tatom "conflict" "1"))
            (Patom Zero))

p3 : Ppov p_d p_e
p4 : Pdov p3 p_c
```

[3]Available at http://www.prismmodelchecker.org/

The intermediary policies are introduced for the sake of readability. This policy is automatically translated to the following PRISM model (for the sake of conciseness, we only detail the module for the attribute value $(\mathbf{r}, \mathsf{nurse})$, the other modules being analogous):

```
module att_role_nurse
        role_nurse: bool init false;
        r_role_nurse: bool init false;
        [] !r_role_nurse → (role_nurse'=true) &
              (r_role_nurse'=true);
        [] !r_role_nurse → (role_nurse'=false) &
              (r_role_nurse'=true);
endmodule

module att_role_phys ... endmodule
module att_emergency_1 ... endmodule
module att_conflict_1 ... endmodule

formula p_d = role_phys? 1: 2;

formula p_e = (role_nurse & emergency_1) ? 1: 2;
formula p_c = conflict_1 ? 0: 2;

formula p3 = (p_d = 1 | p_e = 1)? 1 : min(p_d, p_e);
formula p4 = min(p3, p_c);
```

4.3.4 Policy evaluation

As stated in Proposition 2, in order to evaluate whether $d \in [\![p]\!]_{\mathrm{N}}(q)$, we need to evaluate the path property $\mathsf{EF}\widehat{p_d}$ from the state corresponding to the request q. Based on the previous definition, the path property $\mathsf{EF}\delta_{p,d}$ can be expressed in PRISM as E [F $\widehat{p} = d$].

However, PRISM starts by default from the initial state, which, in our model, corresponds to the empty request (i.e., the state with all propositions set to false). In order to "reach" first the state corresponding to the request q, we first define $\widetilde{q}$, which is the conjunction of all attributes retrieved in q:

$$\widetilde{q} = \begin{cases} true & \text{if } q = \emptyset, \\ \text{r\_}n\_v \;\&\; n\_v \;\&\; \widetilde{q'} & \text{if } q = q' \cup \{(n, v)\} \\ \text{r\_}n\_v \;\&\; !n\_v \;\&\; \widetilde{q'} & \text{if } q = q' \cup \{(n, \overline{v})\} \end{cases}$$

We can use the command filter, such that filter(op, prop, states) computes the value of the property prop for each state satisfying states, and combines these values using the operator op. In this case, we use the operator first, which evaluates prop on the first state that matches states, starting from the initial state and following a lexicographic ordering, where **false** is less than **true**.

Finally, we can check if the path property $\mathsf{EF}\delta_{p,d}$ holds (and thus evaluate $[\![\cdot]\!]_{\mathrm{N}}$, according to Proposition 2) by checking whether the PRISM property filter(first, E [F $\widehat{p} = d$], $\widetilde{q}$) holds over the generated model.

5. PROBABILISTIC RETRIEVAL

Consider the policy p_1 defined in Section 3.4: $[\![p_1]\!]_{\mathrm{N}}(\{\mathbf{r}, \mathsf{phys}\}) = \{1, 0\}$, whereas, according to the original PTaCL semantics, $[\![p_1]\!]_{\mathrm{P}}(\{\mathbf{r}, \mathsf{phys}\}) = \{1\}$. The indeterminacy in the evaluation of $[\![\cdot]\!]_{\mathrm{N}}$ is due to the fact that depending on the value of the attribute $\mathbf{cf}$, the decision could be either 1 or 0, and thus both decisions are returned.

In practice, it could be argued that the likelihood for a physician to be in conflict with a patient is quite low, and even though it is useful to know that this possibility exists, the final decision might take into account this likelihood. In this section, we describe how the attribute values can be associated with a given probability. Intuitively, we propose the following approach:

- an attribute value can be either non-deterministic (as in the previous section) or probabilistic;
- we want to know the probability of reaching a decision from a given request rather than just checking whether this decision is reachable;
- before calculating this probability, we need to resolve first the retrieval of non-deterministic attribute values, which leads to multiple probabilities for each decision, one for each possible resolution.

5.1 Attribute value probability

The probability of an attribute value can be modelled with a partial function $\mathsf{Pr} : \mathcal{N} \times \mathcal{V} \mapsto [0, 1]$, such that if $\mathsf{Pr}(n, v)$ is defined, then $\mathsf{Pr}(n, \overline{v})$ is also defined and $\mathsf{Pr}(n, \overline{v}) = 1 - \mathsf{Pr}(n, v)$. If $\mathsf{Pr}(n, v)$ is undefined, then the retrieval of this attribute value remains non-deterministic. Hence, we do not assume that all attribute values are associated with a probability, instead we propose to include probabilities in the decision evaluation when they are defined. We however assume that all probabilities are independent, and do not depend on non deterministic attributes, and we leave for future work the study of more complex probabilistic models. Let us also point out that the probability does not effectively appear in the request, which means that the previous evaluation functions can still be applied. In other words, probabilities can always be ignored if they are not relevant.

As described above, the first step for the probabilistic evaluation of a request is to retrieve the non-deterministic attribute values. Given a request q, we define $\mathsf{ND}(q)$ to be the set of requests which corresponds to q with each non-deterministic attribute value retrieved. More precisely, $\mathsf{ND}(q)$ is the set of requests $q' \supseteq q$ such that for any (n, v) where $\mathsf{Pr}(n, v)$ is undefined, either (n, v) or $(n, \overline{v})$ belongs to q', and for any (n, v) where $\mathsf{Pr}(n, v)$ is defined, (n, v) belongs to q' if and only if (n, v) belongs to q. If no attribute is probabilistic, then $\mathsf{ND}(q)$ corresponds to the set of complete and well-formed requests that include q.

For instance, consider the previous policy p_1 where we define $\mathsf{Pr}(\mathbf{emg}, true) = 0.1$ and $\mathsf{Pr}(\mathbf{cf}, true) = 0.05$. Given the requests $q_1 = \{(\mathbf{r}, \mathsf{phys})\}$ and $q_2 = \{(\mathbf{cf}, true)\}$, we have:

$$\begin{aligned} \mathsf{ND}(q_1) = \{ & \{(\mathbf{r}, \mathsf{phys}), (\mathbf{r}, \mathsf{nurse})\}, \\ & \{(\mathbf{r}, \mathsf{phys}), (\mathbf{r}, \overline{\mathsf{nurse}})\}\} \\ \mathsf{ND}(q_2) = \{ & \{(\mathbf{cf}, true), (\mathbf{r}, \mathsf{phys}), (\mathbf{r}, \mathsf{nurse})\}, \\ & \{(\mathbf{cf}, true), (\mathbf{r}, \mathsf{phys}), (\mathbf{r}, \overline{\mathsf{nurse}})\}\} \\ & \{(\mathbf{cf}, true), (\mathbf{r}, \overline{\mathsf{phys}}), (\mathbf{r}, \mathsf{nurse})\}\} \\ & \{(\mathbf{cf}, true), (\mathbf{r}, \overline{\mathsf{phys}}), (\mathbf{r}, \overline{\mathsf{nurse}})\}\} \end{aligned}$$

For q_1, the only non-deterministic value is $(\mathbf{r}, \mathsf{nurse})$, and we can add both corresponding values, but we do not add any probabilistic value. For q_2, since $(\mathbf{cf}, true)$ is already retrieved, we do not remove it, but we retrieve instead all possible values for $(\mathbf{r}, \mathsf{phys})$ and $(\mathbf{r}, \mathsf{nurse})$.

Now, given a request in $\mathsf{ND}(q)$, we need to explore all possible extensions with the probabilistic attribute values, and to check the probability with which each decision can be returned. Intuitively, we add all possible probabilistic values not already retrieved, and we multiply the corresponding probability of these values. Hence, given a request $q' \in \mathsf{ND}(q)$, we define $\mathsf{NDP}(q')$ to be the set of requests q'' such that $q'' \supseteq q'$ and q'' is complete and well-formed (i.e., $\mathsf{NDP}(q')$ corresponds to all possible retrieval for the probabilistic values not already in q'). Each q'' is associated with a probability defined as the product of the probabilities of all probabilistic values (n, v) (where v can either be a positive or negative value) belonging to q'' but not to q', which we denote $\mathsf{Pr}(q'' \mid q')$. More formally:

$$\mathsf{Pr}(q'' \mid q') = \prod \left\{ \mathsf{Pr}(n, v) \mid (n, v) \in (q'' \setminus q') \right\}$$

For instance, for each request in $q' \in \mathsf{ND}(q_1)$ defined above, we can defined $\mathsf{NDP}(q')$ as the set of $q' \cup pv_i$, where the sets pv_i are defined as:

- $pv_1 = \{(\mathbf{emg}, \mathit{true}), (\mathbf{cf}, \mathit{true})\}$, with the probability $0.1 * 0.05 = 0.005$;
- $pv_2 = \{(\mathbf{emg}, \mathit{true}), (\mathbf{cf}, \mathit{false})\}$, with the probability $0.1 * 0.95 = 0.095$;
- $pv_3 = \{(\mathbf{emg}, \mathit{false}), (\mathbf{cf}, \mathit{true})\}$, with the probability $0.9 * 0.05 = 0.045$;
- $pv_4 = \{(\mathbf{emg}, \mathit{false}), (\mathbf{cf}, \mathit{false})\}$, with the probability $0.9 * 0.95 = 0.855$.

Similarly, we can add to each request in $\mathsf{ND}(q_2)$ defined above either the set $pv_5 = \{(\mathbf{emg}, \mathit{true})\}$ with the probability 0.1 or the set $pv_6 = \{(\mathbf{emg}, \mathit{false})\}$ with the probability 0.9.

Finally, we can define the probability of a decision d to be reached from a request $q' \in \mathsf{ND}(q)$ by aggregating the probability of the requests in $\mathsf{NDP}(q')$ for which d is returned, which we defined as $\mathsf{Pr}(d \mid q')$:

$$\mathsf{Pr}(d \mid q') = \sum \left\{ \mathsf{Pr}(q'' \mid q') \mid q'' \in \mathsf{NDP}(q') \wedge [\![p]\!]_{\mathrm{C}}(q'') = d \right\}.$$

For instance, in the above example, consider the request $q_{11} = \{(\mathbf{r}, \mathsf{phys}), (\mathbf{r}, \mathsf{nurse})\}$ in $\mathsf{ND}(q_1)$. We have:

$$[\![p_1]\!]_{\mathrm{C}}(q_{11} \cup pv_1) = [\![p_1]\!]_{\mathrm{C}}(q_{11} \cup pv_3) = 0$$
$$[\![p_1]\!]_{\mathrm{C}}(q_{11} \cup pv_2) = [\![p_1]\!]_{\mathrm{C}}(q_{11} \cup pv_4) = 1$$

and it follows that

$$\mathsf{Pr}(0 \mid q_{11}) = 0.005 + 0.095 = 0.05$$
$$\mathsf{Pr}(1 \mid q_{11}) = 0.095 + 0.855 = 0.95$$
$$\mathsf{Pr}(\perp \mid q_{11}) = 0$$

The same results hold for $\{(\mathbf{r}, \mathsf{phys}), (\mathbf{r}, \overline{\mathsf{nurse}})\}$, which corresponds with the fact that if $(\mathbf{r}, \mathsf{phys})$ belongs to the requests, then the request can only be denied if there is a conflict of interest, which can happen with a probability 0.05, otherwise it is granted. In this case, the retrieval of the attribute value $(\mathbf{r}, \mathsf{nurse})$ has no impact.

However, if we consider the request $q_3 = \{(\mathbf{r}, \mathsf{nurse})\}$, the probability of reaching each decision depends on the retrieval of the non-deterministic attribute value $(\mathbf{r}, \mathsf{phys})$. Indeed, consider first the request $q_{31} = \{(\mathbf{r}, \mathsf{nurse}), (\mathbf{r}, \mathsf{phys})\} \in \mathsf{ND}(q_3)$. Clearly, $q_{31} = q_{11}$ and therefore the previous probabilities of reaching each decisions are the same as above.

Table 4: Evaluation of p_1 with $\mathsf{Pr}(\mathbf{emg}, \mathit{true}) = 0.1$ and $\mathsf{Pr}(\mathbf{cf}, \mathit{true}) = 0.05$, where for each q in the first column, the value in the column d corresponds to $[[\![p_1]\!]_{\min}(q, d), [\![p_1]\!]_{\max}(q, d)]$.

	decision d		
Request	0	1	$\perp$
$\emptyset$	[0.05,0.05]	[0,0.95]	[0,0.95]
$\{(\mathbf{r}, \mathsf{phys})\}$	[0.05,0.05]	[0.95,0.95]	[0,0]
$\{(\mathbf{r}, \mathsf{phys}), (\mathbf{cf}, \mathit{true})\}$	[1,1]	[0,0]	[0,0]
$\{(\mathbf{r}, \mathsf{nurse})\}$	[0.05,0.05]	[0.095,0.95]	[0,0.855]
$\{(\mathbf{r}, \mathsf{nurse}), (\mathbf{emg}, \mathit{true})\}$	[0.05,0.05]	[0.95,0.95]	[0,0]

However consider now the request $q_{32} = \{(\mathbf{r}, \mathsf{nurse}), (\mathbf{r}, \overline{\mathsf{phys}})\} \in \mathsf{ND}(q_3)$

$$[\![p_1]\!]_{\mathrm{C}}(q_{32} \cup pv_1) = [\![p_1]\!]_{\mathrm{C}}(q_{32} \cup pv_3) = 0$$
$$[\![p_1]\!]_{\mathrm{C}}(q_{32} \cup pv_2) = 1$$
$$[\![p_1]\!]_{\mathrm{C}}(q_{32} \cup pv_4) = \perp$$

and it follows that

$$\mathsf{Pr}(0 \mid q_{32}) = 0.005 + 0.095 = 0.05$$
$$\mathsf{Pr}(1 \mid q_{32}) = 0.095$$
$$\mathsf{Pr}(\perp \mid q_{32}) = 0.855$$

Hence, the probability of reaching the decisions 1 and $\perp$ depend on the retrieval of the attribute value $(\mathbf{r}, \mathsf{phys})$. Since this value is non-deterministic, the probabilities $\mathsf{Pr}(1 \mid q_{31})$ and $\mathsf{Pr}(1 \mid q_{32})$ correspond to different possible futures, and thus cannot be "merged" together. We therefore define here two new evaluation functions, $[\![\cdot]\!]_{\min}$ and $[\![\cdot]\!]_{\max}$, which compute the minimal and the maximal probability to reach a given decision d from a given query q. More formally:

$$[\![p]\!]_{\min}(q, d) = \min_{q' \in \mathsf{ND}(q)} \mathsf{Pr}(q' \mid d)$$
$$[\![p]\!]_{\max}(q, d) = \max_{q' \in \mathsf{ND}(q)} \mathsf{Pr}(q' \mid d)$$

Table 4 summaries the evaluation of the policy p_1 for the different requests previously considered. It is worth pointing out that the resolution of the non-determinism used to calculate the minimal/maximal probability for a decision can change from one decision to another, so the minimal/maximal probabilities do not necessarily add up to 1.

It is also worth noting that $[\![\cdot]\!]_{\min}$ and $[\![\cdot]\!]_{\max}$ are consistent with $[\![\cdot]\!]_{\mathrm{N}}$, i.e., for any policy p, any request q and any decision d, $[\![p]\!]_{\min}(q, d) = [\![p]\!]_{\max}(q, d) = 0$ if and only if $d \notin [\![p]\!]_{\mathrm{N}}(q)$. The interest of $[\![\cdot]\!]_{\min}$ and $[\![\cdot]\!]_{\max}$ therefore lies with decision that have non null probabilities.

When $[\![p]\!]_{\min}(q, d) = [\![p]\!]_{\max}(q, d)$, the probability of reaching a decision is the same regardless of the resolution of non-determinism, which provides useful information to reach a conclusive decision. For instance, for the request $\{(\mathbf{r}, \mathsf{phys})\}$, it could be a reasonable choice to select the conclusive decision 1, since it is the one with the highest probability.

However, when $[\![p]\!]_{\min}(q, d) \neq [\![p]\!]_{\max}(q, d)$, deciding a conclusive decision can be more complex. For instance, with the empty request, we can observe that the minimal probability for the decision 0 is not null, even though this decision is not returned in the basic PTaCL evaluation (see Table 2). On the other hand, there are some non-deterministic retrieval for which the probability of 1 is null. Clearly, there is no easy way to solve this particular request, and probabilistic

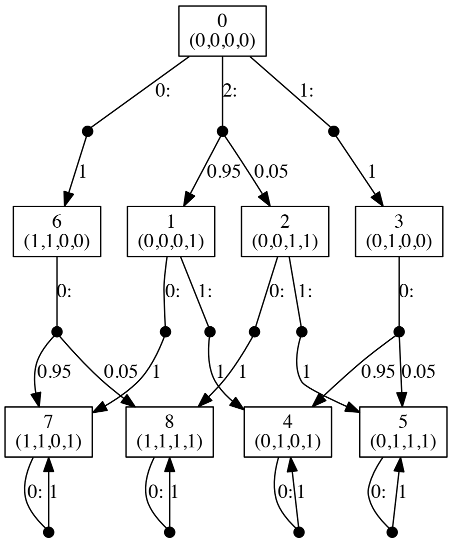


Figure 1: PRISM transition matrix with the values $(\mathbf{r}, \mathsf{phys})$ and $(\mathbf{cf}, \mathit{true})$, and $\mathsf{Pr}(\mathbf{cf}, \mathit{true}) = 0.05$.

modelling in this case should not be seen as a answer for all situations, but as a tool able to help with some situations.

5.2 PRISM encoding

For any attribute value (n, v) such that $\mathsf{Pr}(n, v) = p_{nv}$, we change the generated PRISM module to:

```
const double p_n_v = p_nv;

module att_n_v
 n_v: bool init false;
 r_n_v: bool init false;
 [] !r_n_v → p_n_v:(n_v'=true) & (r_n_v'=true)
        + (1 − p_n_v):(n_v'=false) & (r_n_v'=true);
endmodule
```

The encoding of non-deterministic attribute values and of target and policies is identical to the previous section.

For instance, Figure 1 is automatically generated from the PRISM model containing the values $(\mathbf{r}, \mathsf{phys})$ and $(\mathbf{cf}, \mathit{true})$[4]. Each state is therefore a tuple (r_role_phys, role_phys, r_cf_true, cf_true), transitions with a label i: corresponds to non-deterministic choices (note that there is no actual ordering between these transitions, the label only serve for identification purposes), while the other transitions are labelled with their associated probability. The label inside each square corresponds to the lexicographic ordering of the states.

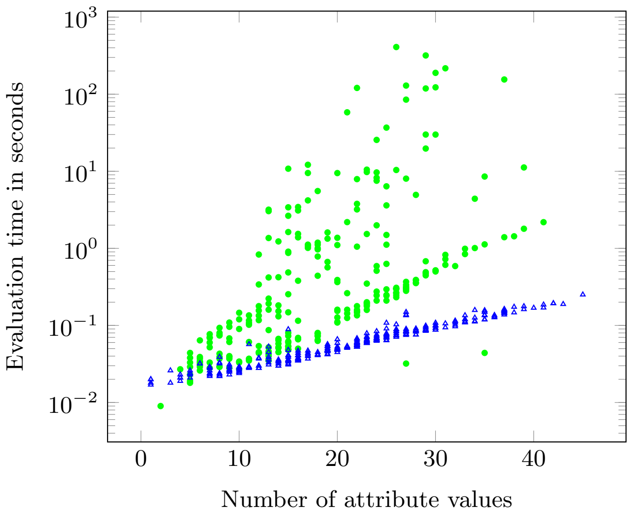
In order to compute the function $[\![\cdot]\!]_{\min}$ and $[\![\cdot]\!]_{\max}$, we use the PRISM operators Pmin and Pmax, respectively. Given a path property ϕ, Pmin =? ϕ and Pmax =? ϕ returns the minimum and maximum probabilities of this property to hold, after resolving non-determinism.

Proposition 3. *Given a policy p, a request q and a decision d, we have:*

$$[\![p]\!]_{\min}(q, d) = \mathsf{filter}(\mathsf{first},\ \mathsf{Pmin} =?\ [\mathsf{F\ complete}\ \&\ \widehat{p} = d],\ \widetilde{q})$$
$$[\![p]\!]_{\max}(q, d) = \mathsf{filter}(\mathsf{first},\ \mathsf{Pmax} =?\ [\mathsf{F\ complete}\ \&\ \widehat{p} = d],\ \widetilde{q})$$

[4]Any larger model is too large to be meaningfully presented here.

Figure 2: Performance analysis of $[\![\cdot]\!]_{\mathrm{N}}$ (blue triangles) and $[\![\cdot]\!]_{\min} + [\![\cdot]\!]_{\max}$ (green triangles) for randomly generated PTaCL policies.



Proof. The combination of first and $\widetilde{q}$ ensures that we only consider paths starting from the state corresponding exactly to the request q. Since we impose in the path property for complete to hold, we know that we only check the decision on complete requests that extend q. Finally, the operators Pmin and Pmax automatically resolve the non-determinism by considering all possible non-deterministic retrievals. □

6. PERFORMANCE EVALUATION

In order to measure the performances of our PRISM encoding, we have generated 261 random PTaCL policies, with a number of attribute values included between 1 and 42, such that some attribute values are associated with a random probability, while the others are defined as non-deterministic.

For each policy p, we first measure the time required to compute $[\![p]\!]_{\mathrm{N}}(\emptyset)$. Note that the empty request is the request requiring the most space exploration. These values are depicted with the blue triangle points in Figure 2, and have been measured with PRISM v4.2.beta1 and the MTBDD engine, and a Macbook Air with 2 GHz Intel Core i7 and 8 GB or RAM. We can observe that the computation time of $[\![\cdot]\!]_{\mathrm{N}}$, although exponentially increasing with the number of attribute values (the y axis uses a logarithmic scale), is consistently below 0.1 second.

For each policy, we then compute $[\![p]\!]_{\min}(\emptyset, d)$ and $[\![p]\!]_{\max}(\emptyset, d)$, for all $d \in \{0, 1, \bot\}$, and each green circle in Figure 2 represents the sum of the evaluation of all corresponding PRISM properties. We can observe that the evaluation time also increases exponentially with the number of attribute values. There is more variety in the evaluation time compared to $[\![\cdot]\!]_{\mathrm{N}}$, and for some policy, it can take more than 100 seconds to evaluate the empty requests. However, it is worth pointing out that 88 out 261 policies evaluated under 0.1 seconds, 189 under 1 second and 240 under 10 seconds. Finally, for each policy, the time required by PRISM by building and loading in memory the model is very close to that required to compute $[\![\cdot]\!]_{\mathrm{N}}$, and therefore is not shown in Figure 2. Note that this time is the same, regardless of the evaluation function chosen.

These results show first that that $[\![\cdot]\!]_{\mathrm{N}}$ is relatively efficient, with little variation, while $[\![\cdot]\!]_{\min}$ and $[\![\cdot]\!]_{\max}$ can be still practical, but can also be very long to compute. However, it is also worth pointing out that concrete access control policies tend to be more structured than randomly generated ones, and as such, model checking can be more efficient. In addition, some decisions can be cached [16]. Hence, these results should not necessarily be interpreted as providing an average computation time based on the size of the policy, but rather as an indication that $[\![\cdot]\!]_{\mathrm{N}}$ can be first used for evaluation, since it is relatively fast, and in case of indeterminacy, $[\![\cdot]\!]_{\min}$ and $[\![\cdot]\!]_{\max}$ can be used to try to decide on a conclusive decision based on the probabilities of the decisions. Of course, as illustrated in the previous section, these probabilities are not necessarily sufficient to decide on a conclusive decision, but our approach consists in providing as much meaningful information as possible.

7. DISCUSSION – CONCLUSION

In the previous sections, given a policy p and a request q, we have defined several evaluation functions, each computing a different set of decisions for the same request:

- $[\![p]\!]_{\mathrm{P}}(q)$ is the original PTaCL definition, which follows the XACML definition, and which considers that if an attribute value (n, v) is required by the target of a sub-policy of p, but the n is not present at all in q, then the target evaluation fails and the sub-policy evaluates both to $\bot$ and to the value the policy would have returned had (n, v) been present in q.

- $[\![p]\!]_{\mathrm{C}}(q)$ considers that in the case described above, the target evaluation should not fail, and the policy should simply evaluate to $\bot$.

- $[\![p]\!]_{\mathrm{N}}(q)$ returns all the possible decisions reachable by adding all missing attributes in q.

- $[\![p]\!]_{\min}(q, d)$ and $[\![p]\!]_{\max}(q, d)$ consider that some attribute values can be probabilistic and return the minimum and maximum probability, respectively, to reach the decision d by adding all missing attributes in q, for any possible retrieval of the non-deterministic attribute values.

We have shown in Section 3.4 that $[\![\cdot]\!]_{\mathrm{P}}$ can be counter intuitive, because it can return some decisions that are not reachable, and not return some decisions that are reachable, where $[\![\cdot]\!]_{\mathrm{N}}$ returns exactly all reachable decisions. $[\![\cdot]\!]_{\min}$ and $[\![\cdot]\!]_{\max}$ are equivalent to $[\![\cdot]\!]_{\mathrm{N}}$ when there is no probabilistic attribute value, and can provide more information of the reachability of the decisions otherwise.

However, although these new evaluation functions are somehow more accurate than $[\![\cdot]\!]_{\mathrm{P}}$, they are also more computationally intensive, since they must explore the space of possible request extensions, as shown in the previous section.

An interesting and general question concerns whether probabilities can be meaningfully used in security systems, and this question can be split in two parts. Firstly, is it possible to define the probabilities for attribute values? Clearly, in general, it is not possible to know the probability of all possible attribute values. However, we believe that a strength of our approach is its ability to mix non-deterministic and probabilistic attribute retrieval, which allows for a "best effort" strategy: by specifying the information we know, we can get a somehow more accurate analysis.

Secondly, can the resolution mechanism use these probabilities to make a final decision? This aspect is particularly important when the resolution mechanism is done by human users, who are known not to be particularly good at understanding probabilities when making choices [15]. In other words, the way we present the result from the function $[\![\cdot]\!]_{\min}$ and $[\![\cdot]\!]_{\max}$ can have an impact on the final choice made by a user, as suggested in [9].

In addition, more complex policy analyses could be considered from the angle of non-deterministic and probabilistic attribute retrieval, such as policy safety [7], or the integration of administrative policies.

8. REFERENCES

[1] P. Bonatti, S. De Capitani Di Vimercati, and P. Samarati. An algebra for composing access control policies. *ACM Transactions on Information and System Security*, 5(1):1–35, 2002.

[2] G. Bruns and M. Huth. Access control via Belnap logic: Intuitive, expressive, and analyzable policy composition. *ACM Transactions on Information and System Security*, 14(1):9, 2011.

[3] E. M. Clarke, O. Grumberg, and D. Peled. *Model checking*. MIT press, 1999.

[4] J. Crampton and M. Huth. An authorization framework resilient to policy evaluation failures. In *ESORICS*, LNCS 6345, pages 472–487, 2010.

[5] J. Crampton and C. Morisset. PTaCL: A language for attribute-based access control in open systems. In *POST*, LNCS 7215, pages 390–409, 2012.

[6] K. Fisler, S. Krishnamurthi, L. A. Meyerovich, and M. C. Tschantz. Verification and change-impact analysis of access-control policies. In *Proceedings of the 27th International Conference on Software Engineering*, pages 196–205. ACM, 2005.

[7] A. Griesmayer and C. Morisset. Automated certification of authorisation policy resistance. In *ESORICS*, LNCS 8134, pages 574–591. Springer, 2013.

[8] S. Kleene. *Introduction to Metamathematics*. D. Van Nostrand, Princeton, NJ, 1950.

[9] C. Morisset, T. Groß, A. P. A. van Moorsel, and I. Yevseyeva. Nudging for quantitative access control systems. In *Human Aspects of Information Security, Privacy, and Trust*, LNCS 8533, pages 340–351. Springer, 2014.

[10] C. Morisset and N. Zannone. Reduction of access control decisions. In *Proceedings of the 19th ACM Symposium on Access Control Models and Technologies*, pages 53–62. ACM, 2014.

[11] OASIS. *eXtensible Access Control Markup Language (XACML) Version 3.0*, 2010. Committee Specification.

[12] S. Ranise and A. T. Truong. Incremental analysis of evolving administrative role based access control policies. In *Data and Applications Security and Privacy*, LNCS 8566, pages 260–275. Springer, 2014.

[13] S. Ranise, A. T. Truong, and A. Armando. Scalable and precise automated analysis of administrative temporal role-based access control. In *Proceedings of*

19th Symposium on Access Control Models and Technologies, pages 103–114. ACM, 2014.

[14] M. C. Tschantz and S. Krishnamurthi. Towards reasonability properties for access-control policy languages. In *Proceedings of 11th ACM Symposium on Access Control Models and Technologies*, pages 160–169. ACM, 2006.

[15] A. Tversky and D. Kahneman. The framing of decisions and the psychology of choice. *Science*, 211(4481):453–458, 1981.

[16] Q. Wei, J. Crampton, K. Beznosov, and M. Ripeanu. Authorization recycling in hierarchical RBAC systems. *ACM Trans. Inf. Syst. Secur.*, 14(1):3, 2011.

[17] D. Wijesekera and S. Jajodia. A propositional policy algebra for access control. *ACM Transactions on Information and System Security*, 6(2):286–235, 2003.

[18] N. Zhang, M. Ryan, and D. P. Guelev. Evaluating access control policies through model checking. In *ISC*, LNCS 3650, pages 446–460, 2005.