An Algebra for Composing Enterprise Privacy Policies

Michael Backes[1], Markus Dürmuth[2], and Rainer Steinwandt[2]

[1] IBM Zurich Research Laboratory, Switzerland
mbc@zurich.ibm.com
[2] IAKS, Arbeitsgruppe Systemsicherheit, Prof. Dr. Th. Beth,
Universität Karlsruhe, Germany
markus.duermuth@web.de, steinwan@ira.uka.de

Abstract. Enterprise privacy enforcement allows enterprises to internally enforce a privacy policy that the enterprise has decided to comply to. To facilitate the compliance with different privacy policies when several parts of an organization or different enterprises cooperate, it is crucial to have tools at hand that allow for a practical management of varying privacy requirements.
We propose an algebra providing various types of operators for composing and restricting enterprise privacy policies like conjunction, disjunction, and scoping, together with its formal semantics. We base our work on a superset of the syntax and semantics of IBM's Enterprise Privacy Authorization Language (EPAL), which recently has been submitted to W3C for standardization. However, a detailed analysis of the expressiveness of EPAL reveals that, somewhat surprisingly, EPAL is not closed under conjunction and disjunction. To circumvent this problem, we identified the subset of well-founded privacy policies which enjoy the property that the result of our algebraic operations can be turned into a coherent privacy policy again. This enables existing privacy policy enforcement mechanisms to deal with our algebraic expressions. We further show that our algebra fits together with the existing notions of privacy policy refinement and sequential composition of privacy policies in a natural way.

1 Introduction

Not only due to the increasing privacy awareness of costumers, the proper incorporation of privacy considerations into business processes is gaining importance. Also regulatory measures like the Children's Online Privacy Protection Act (COPPA) or the Health Insurance Portability and Accountability Act (HIPAA) illustrate that avoiding violations of privacy regulations is becoming a crucial issue. While the Platform for Privacy Preferences Project (P3P) [21] is a valuable tool for dealing with privacy concerns of web site users, the fine-grained treatment of privacy policies in business-to-business matters is still not settled satisfyingly. E.g., a language for the internal privacy practices of enterprises and for technical privacy enforcement must offer more possibilities for fine-grained distinction of data users, purposes, etc., as well as a clearer semantics. To live up to these requirements, enterprise privacy technologies are emerging [9]. One approach for capturing the privacy requirements of an enterprise – which however does not specify the implementation of these requirements – is the use of formalized enterprise privacy policies [11, 17, 16].

P. Samarati et al. (Eds.): ESORICS 2004, LNCS 3193, pp. 33–52, 2004.

Although the primary purpose of enterprise privacy policies is enterprise-internal use, many factors speak for standardization of such policies. E.g., it would allow certain technical parts of regulations to be encoded into such a standardized language once and for all, and a large enterprise with heterogeneous repositories of personal data could then hope that enforcement tools for all these repositories become available that allow the enterprise to consistently enforce at least the internal privacy practices chosen by the CPO (chief privacy officer). For these reasons, IBM has proposed an Enterprise Privacy Authorization Language (EPAL) as an XML specification, which has been submitted to W3C for standardization. EPAL allows for a fine-grained description of privacy requirements in enterprises and could become a valuable tool for (business) processes that span several enterprises or different parts of a larger organization.

An enterprise privacy policy often reflects different legal regulations, promises made to customers, as well as more restrictive internal practices of the enterprise. Further, it may allow customer preferences. Hence it may be authored, maintained, replaced, and audited in a distributed fashion. In other words, one will need a life-cycle management system for the collection of enterprise privacy policies. However, despite considerable advancement in this area, current approaches are based on monolithic and complete specifications, which is very restrictive given that several policies might have to be enforced at once while being under control of different authorities. Having in mind actual use cases where sensitive data obeying different privacy regulations has to be merged or exchanged, this situation calls for a composition framework that allows for integrating different privacy policies while retaining their independence. While such thoughts occur as motivation in most prior work on enterprise privacy policies, the few tools provided so far are not very expressive, and even intuitively simple operations have not been formalized yet.

Motivated by successful applications of algebraic tools in access control [7, 24, 8, 25], our goal is to provide an expressive algebra over enterprise privacy policies together with its formal semantics, offering operators for combining and restricting policies, along with suitable algebraic laws that allow for a convenient policy management. We do this concretely for the IBM EPAL proposal. However, for a scientific paper it is desirable to avoid the lengthy XML syntax and use a corresponding abstract syntax presented in [2, 4] and known as E-P3P (which, like EPAL, is based on [17]).

To employ existing privacy policy enforcement mechanisms to our algebraic expressions, it is necessary to represent the results of the operators as a syntactically correct privacy policy again. While achieving this representation has been identified as a core property in previous work on algebras for access control policies [7], and also explored there in detail, achieving the same result for enterprise privacy policies as in EPAL seems rather involved because of the treatment of obligations, different policy scopes, default values as well as a sophisticated treatment of deny-rules. In fact, our analysis of the expressiveness of EPAL shows that EPAL is not closed under operations like conjunction and disjunction, hence the aforementioned representation can often not be achieved. To circumvent this problem, we identify the set of well-founded policies, which constitutes a subset of all EPAL policies, for which we can give a constructive algorithm that represents the results of our algebraic operations as a well-founded EPAL policy again.

The first operators we define are policy conjunction and disjunction, which serve as the basic building blocks for constructing larger policies. For instance, an enterprise might first take all applicable regulations and combine them into a minimum policy by means of the conjunction operator. A general promise made to the customers, e.g., an existing P3P translated into the more general language, may be a further input. As one expects, these operators are not a simple logical AND respectively OR for expressive enterprise privacy policies for the reasons depicted above. We show that these operators enjoy the expected algebraic laws like associativity or distributivity. Our third operator – scoping – allows for confining the scope of a policy to sub-hierarchies of a policy. This is of major use in practice as it enables managing respectively reasoning about privacy requirements that involve only certain parts of an organization.

We further sketch some extensions of our algebra; in particular, we incorporate the sequential composition of privacy policies, which has been introduced in [4], and we explore its relationship to our remaining operators.

Further related literature. Policy composition has been treated before, in particular for access control [7, 8, 10, 19, 15, 26], systems management [20], separation-of-duty [23, 13], or IPSEC [12]. The algebra discussed below is clearly motivated by existing work on algebras for access control polices [7, 24, 8, 25]. We are not aware of a similar proposal for privacy policies although certain aspects have been addressed before, e.g., [18] points out possible conflicts if EPAL policies from different origins have to be dealt with. The publication closest to our algebra of privacy policies is [4], which introduces a notion of sequential composition of privacy policies as well as the notion of policy refinement. The present paper tries to extend this pool of algebraic tools. Compared with existing access-control languages, the core contribution of new privacy-policy languages [11, 17, 16] is the notion of purpose and purpose-bound collection of data, which is essential to privacy legislation. Other necessary features that prevent enterprises from simply using their existing access-control systems and the corresponding algebras are obligations and conditions on context information. Individually, these features were also considered in literature on access control, e.g., purpose hierarchies in [6], obligations in [5, 14, 22], and conditions on context information in [26].

2 Syntax and Semantics of E-P3P Enterprise Privacy Policies

Informally speaking, the aim of a privacy policy is to define by whom, for which purposes, and in which way collected data can be accessed. Further on, a privacy policy may impose obligations onto the organization using the data. Privacy policies formalize privacy statements like "we use data of a minor for marketing purposes only if the parent has given consent" or "medical data can only be read by the patient's primary care physician". This section mainly recalls the abstract syntax and semantics E-P3P [2, 4] of IBM's EPAL privacy policy language [1] up to some augmentations needed to achieve the desired algebraic properties, e.g., that obligations are already structured in a suitable way. Motivated by recent changes in EPAL, our specification of E-P3P deviates from the one in [4] in the handling of so-called "don't care" rulings: In analogy to EPAL 1.2, we only allow the default ruling to return a "don't care", and we demand that no obligations may be imposed in this case.

2.1 Hierarchies, Obligations, and Conditions

First, we recall the basic notions of hierarchies, obligations, and conditions used in E-P3P, and operations on them as needed in later refinements and operators of our algebra. For conveniently specifying rules, the data, users, etc. are categorized in E-P3P as in many access-control languages. The same applies to the purposes. To allow for structured rules with exceptions, categories are ordered in hierarchies; mathematically they are forests, i.e., multiple trees. For example, a user "company" may group several "departments", each containing several "employees". The enterprise can then write rules for the whole "company" with exceptions for some "departments".

Definition 1 (Hierarchy). *A* hierarchy *is pair $(H, >_H)$ of a finite set H and a transitive, non-reflexive relation $>_H \subseteq H \times H$, where every $h \in H$ has at most one immediate predecessor (parent). As usual we write $\geq_H$ for the reflexive closure.*

For two hierarchies $(H, >_H)$ and $(G, >_G)$, we define

$$(H, >_H) \subseteq (G, >_G) :\Leftrightarrow (H \subseteq G) \wedge (>_H \subseteq >_G);$$
$$(H, >_H) \cup (G, >_G) := (H \cup G, (>_H \cup >_G)^*);$$

where * denotes the transitive closure. Note that a hierarchy union is not always a hierarchy again. ◇

As mentioned above already, E-P3P policies can impose obligations, i.e., duties for an organization/enterprise. Typical examples are to send a notification to the data subject after each emergency access to medical data, or to delete data within a certain time limit. Obligations are not structured in hierarchies, but by an implication relation. E.g., an obligation to delete data within 30 days implies that the data is deleted within 2 months. The overall obligations of a rule in E-P3P are expressed as sets of individual obligations which must have an interpretation in the application domain. As multiple obligations may imply more than each one individually, the implication relation (which must also be realized in the application domain) is specified on these sets of obligations. We also define how this relation interacts with vocabulary extensions.

Definition 2 (Obligation Model). *An* obligation model *is a pair $(O, \rightarrow_O)$ of a set O and a transitive relation $\rightarrow_O \subseteq \mathfrak{P}(O) \times \mathfrak{P}(O)$, spoken* implies*, on the powerset of O, where $\bar{o}_1 \rightarrow_O \bar{o}_2$ for all $\bar{o}_2 \subseteq \bar{o}_1$, i.e., fulfilling a set of obligations implies fulfilling all subsets. For $O' \supset \mathfrak{P}(O)$, we extend the implication to $O' \times \mathfrak{P}(O)$ by $((\bar{o}_1 \rightarrow_O \bar{o}_2) :\Leftrightarrow (\bar{o}_1 \cap O \rightarrow_O \bar{o}_2))$.*

For defining the AND and OR-composition of privacy policies in a meaningful way, we moreover assume that $\mathfrak{P}(O)$ is equipped with an additional operation $\vee$, such that $(\mathfrak{P}(O), \vee, \cup)$ is a distributive lattice; the operator $\vee$ reflects the intuitive notion of OR (in analogy to the set-theoretical union $\cup$ which corresponds to AND). In particular, we require the following:

- *for all $\bar{o}_1, \bar{o}_2 \subseteq O$ we have $\bar{o}_1 \rightarrow_O (\bar{o}_1 \vee \bar{o}_2)$*
- *for all $\bar{o}_1, \bar{o}_2, \bar{o}'_1, \bar{o}'_2 \subseteq O$ we have $(\bar{o}_1 \rightarrow_O \bar{o}_2) \wedge (\bar{o}'_1 \rightarrow_O \bar{o}'_2)$ implies both $(\bar{o}_1 \vee \bar{o}'_1) \rightarrow_O (\bar{o}_2 \vee \bar{o}'_2)$ and $(\bar{o}_1 \cup \bar{o}'_1) \rightarrow_O (\bar{o}_2 \cup \bar{o}'_2)$.*

Finally, we assume that all occurring obligation models $(O, \rightarrow_O)$ *are subsets of a fixed (super) obligation model* $OM_0 = (O_0, \rightarrow_{O_0})$ *such that* $\rightarrow_O$ *is the restriction of* $\rightarrow_{O_0}$ *to* $\mathfrak{P}(O) \times \mathfrak{P}(O)$. ◇

The decision formalized by a privacy policy can depend on context data like the age of a person. In EPAL this is represented by conditions over data in so-called containers [1]. The XML representation of the formulas is taken from [26], which corresponds to a predicate logic without quantifiers. In the abstract syntax in [2], conditions are abstracted into propositional logic, which is too coarse for our purposes. Hence, as in in [4] we use an extension of E-P3P formalizing the containers as a set of variables with domains and the conditions as formulas over these variables.

Definition 3 (Condition Vocabulary). *A* condition vocabulary *is a pair* $Var = (V, Scope)$ *of a finite set* V *and a function assigning every* $x \in V$*, called a* variable*, a set* $Scope(x)$*, called its* scope*.*

Two condition vocabularies $Var_1 = (V_1, Scope_1)$*,* $Var_2 = (V_2, Scope_2)$ *are* compatible *if* $Scope_1(x) = Scope_2(x)$ *for all* $x \in V_1 \cap V_2$*. For that case, we define their* union *by* $Var_1 \cup Var_2 := (V_1 \cup V_2, Scope_1 \cup Scope_2)$. ◇

One may think of extending this to a full signature in the sense of logic, i.e., including predicate and function symbols – in EPAL, this is hidden in user-defined functions that may occur in the XACML conditions. For the moment, we assume a given universe of predicates and functions with fixed domains and semantics.

Definition 4 (Condition Language). *Let a condition vocabulary* $Var = (V, Scope)$ *be given.*

- *The* condition language $C(Var)$ *is the set of correctly typed formulas over* V *using the assumed universe of predicates and functions, and in the given syntax of predicate logic without quantifiers.*
- *An* assignment *of the variables is a function* $\chi \colon V \to \bigcup_{x \in V} Scope(x)$ *with* $\chi(x) \in Scope(x)$ *for all* $x \in V$*. The set of all assignments for the set* Var *is written* $\mathfrak{Ass}(Var)$*.*
- *For* $\chi \in \mathfrak{Ass}(Var)$*, let* $eval_\chi \colon C(Var) \to \{true, false\}$ *denote the evaluation function for conditions given this variable assignment. This is defined by the underlying logic and the assumption that all predicate and function symbols come with a fixed semantics.*
- *For* $\chi \in \mathfrak{Ass}(Var)$*, we denote by* $c_\chi \in C(Var)$ *some fixed formula such that* $eval_\chi(c_\chi) = true$ *and* $eval_{\chi'}(c_\chi) = false$ *for all* $\chi' \in \mathfrak{Ass}(Var) \setminus \{\chi\}$. ◇

We do not consider partial assignments as is done in [4] since they do not occur in EPAL 1.2 any more.

2.2 Syntax of E-P3P Policies

An E-P3P policy is a triple of a vocabulary, a set of authorization rules, and a default ruling. The vocabulary defines element hierarchies for data, purposes, users, and actions,

as well as the obligation model and the condition vocabulary. Data, users, and actions are as in most access-control policies (except that users are typically called "subjects" there, which in privacy would lead to confusion with data subjects), and purposes are an important additional hierarchy for the purpose binding of collected data.

Definition 5 (Vocabulary). *A* vocabulary *is a tuple* $Voc = (UH, DH, PH, AH, Var, OM)$ *where* UH*,* DH*,* PH*, and* AH *are hierarchies called user, data, purpose, and action hierarchy, respectively, and* Var *is a condition vocabulary and* OM *an obligation model.* ◇

As a naming convention, we assume that the components of a vocabulary Voc are always called as in Definition 5 with $UH = (U, >_U)$, $DH = (D, >_D)$, $PH = (P, >_P)$, $AH = (A, >_A)$, $Var = (V, Scope)$, and $OM = (O, \rightarrow_O)$, except if explicitly stated otherwise. In a vocabulary Voc_i all components also get a subscript i, and similarly for superscripts. Differing from [4] we require that a set of authorization rules (short *ruleset*) only contains authorization rules that allow or deny an operation, i.e., we do not allow rules which yield a "don't care" ruling. This reflects the latest version of EPAL. Further on, motivated by EPAL's implicit handling of precedences through the textual order of the rules, we call a privacy policy *well-formed* if rules which allow for contradicting rulings do not have identical precedences (actually, in EPAL two rules can *never* have identical precedences).

Definition 6 (Ruleset and Privacy Policy). *A* ruleset *for a vocabulary* Voc *is a subset of* $\mathbb{Z} \times U \times D \times P \times A \times C(Var) \times \mathfrak{P}(O) \times \{+, -\}$.

A privacy policy *or* E-P3P policy *is a triple* (Voc, R, dr) *of a vocabulary* Voc*, a rule-set* R *for* Voc*, and a* default ruling $dr \in \{+, \circ, -\}$*. The set of these policies is called* $EP3P$*, and the subset for a given vocabulary* $EP3P(Voc)$*. Moreover, we call* $(Voc, R, dr) \in EP3P$ well-formed*, if for all rules* $(i, u, d, p, a, c, \bar{o}, r), (i, u', d', p', a', c', \bar{o}', r') \in R$ *with identical precedences and for all assignments* $\chi \in \mathfrak{Ass}(Var)$ *the implication* $(eval_\chi(c) = true = eval_\chi(c')) \Rightarrow (r = r')$ *holds.* ◇

The rulings $+$, $\circ$, and $-$ mean "allow", "don't care", and "deny"; the value $\circ$ is special in the sense, that it can only be assigned to the default ruling of a policy. As a naming convention, we assume that the components of a privacy policy called Pol are always called as in Definition 6, and if Pol has a sub- or superscript, then so do the components.

2.3 Semantics of E-P3P Policies

An E-P3P request is a tuple (u, d, p, a) which should belong to the set $U \times D \times P \times A$ for the given vocabulary. Note that E-P3P and EPAL requests are not restricted to "ground terms" as in some other languages, i.e., minimal elements in the hierarchies. This is useful if one starts with coarse policies and refines them because elements that are initially minimal may later get children. For instance, the individual users in a "department" of an "enterprise" may not be mentioned in the CPO's privacy policy, but in the department's privacy policy. For similar reasons, we also define the semantics for requests outside the given vocabulary. We assume a superset $\mathcal{S}$ in which all hierarchy sets are embedded; in practice it is typically a set of strings or valid XML expressions.

Definition 7 (Request). *For a vocabulary Voc, we define the set of* valid requests *as $Req(Voc) := U \times D \times P \times A$. Given a superset $\mathcal{S}$ of the sets U, D, P, A of all considered vocabularies, the set of* all requests *is $Req := \mathcal{S}^4$.*

For valid requests $(u, d, p, a), (u', d', p', a') \in Req(Voc)$ we set

$$(u, d, p, a) \leq (u', d', p', a') :\Leftrightarrow u \leq_U u' \text{ and } d \leq_D d' \text{ and } p \leq_P p' \text{ and } a \leq_A a'.$$

Moreover, we set $(u, d, p, a) <_1 (u', d', p', a')$ if and only if there is exactly one $x \in \{u, d, p, a\}$ such that x' is the parent of x and for all $y \in \{u, d, p, a\} \setminus \{x\}$ we have $y = y'$. Finally, we refer to a valid request $(u, d, p, a) \in Req(Voc)$ as leaf *or* leaf node *if u, d, p, and a are leaves in the respective hierarchy. We denote the set of all leaves of $Req(Voc)$ by $L(Voc)$ and for $q \in Req(Voc)$, we set $L(q, Voc) := \{q' \in L(Voc) \mid q' \leq q\} \setminus \{q\}$.* ◇

The semantics of a privacy policy Pol is a function $eval_{Pol}$ that processes a request based on a given assignment. The evaluation result is a pair $(r, \bar{o})$ of a ruling (also called *decision*) and associated obligations; in the case of a "don't care"-ruling ($r = \circ$) we necessarily have $\bar{o} = \emptyset$, i.e., no obligations are imposed in this case. Our semantics follows the E-P3P semantics in [4], but we restrict our definition to the (from the practical point of view most relevant) case of well-formed policies, which simply avoids a separate treatment of conflicts among rules. We further permit the exceptional ruling $scope\_error$ which indicates that a request was out of the scope of the policy.

The semantics is defined by a virtual pre-processing that unfolds the hierarchies followed by a request processing stage. We stress that this is only a compact definition of the semantics and not an efficient real evaluation algorithm.

Definition 8 (Unfolded Rules). *For a privacy policy $Pol = (Voc, R, dr)$, the* unfolded rule set $UR(Pol)$ *is defined as follows:*

$$\begin{aligned} URD(Pol) := \{&(i, u', d', p', a', c, \bar{o}, r) \in R \mid \exists (i, u, d, p, a, c, \bar{o}, r) \in R \\ &\text{with } u \geq_U u' \wedge d \geq_D d' \wedge p \geq_P p' \wedge a \geq_A a'\}; \\ UR(Pol) := &\ URD(Pol) \\ \cup\ \{&(i, u', d', p', a', c, \bar{o}, -) \in R \mid \exists (i, u, d, p, a, c, \bar{o}, -) \in URD(Pol) \\ &\text{with } u' \geq_U u \wedge d' \geq_D d \wedge p' \geq_P p \wedge a' \geq_A a\}. \end{aligned}$$ ◇

A crucial point in this definition is the fact that "deny"-rules are inherited both downwards and upwards along the four hierarchies while "allow"-rules are inherited downwards only. The reason is that the hierarchies are considered groupings: If access is forbidden for some element of a group, it is also forbidden for the group as a whole.

Next, we define which rules are applicable for a request given an assignment of the condition variables. These (unfolded) rules have the user, data, purpose, and action as in the request, and we make

Definition 9 (Applicable Rules). *Let a privacy policy $Pol = (Voc, R, dr)$, a request $q = (u, d, p, a) \in Req(Voc)$, and an assignment $\chi \in \mathfrak{Ass}(Var)$ be given. Then the set of* applicable rules *is*

$$AR(Pol, q, \chi) := \{(i, u, d, p, a, c, \bar{o}, r) \in UR(Pol) \mid eval_\chi(c) = true\}.$$ ◇

To formulate the semantics, it is convenient to define the maximum and minimum precedence of a policy.

Definition 10 (Precedence Range). *For a privacy policy* $Pol = (Voc, R, dr)$*, let* $max(Pol) := \max\{i \mid \exists(i, u, d, p, a, c, \bar{o}, r) \in R\}$ *and* $min(Pol) := \min\{i \mid \exists(i, u, d, p, a, c, \bar{o}, r) \in R\}$. ◇

We can now define the actual semantics, i.e., the result of a request given an assignment:

Definition 11 (Semantics). *Let a well-formed privacy policy* $Pol = (Voc, R, dr)$*, a request* $q = (u, d, p, a) \in Req$*, and an assignment* $\chi \in \mathfrak{Ass}(Var)$ *be given. Then the* evaluation result $(r, \bar{o}) := eval_{Pol}(q, \chi)$ *of policy* Pol *for* q *and* χ *is defined by the following algorithm, where every "return" is understood to abort the processing of the algorithm.*

1. Out-of-scope testing. *If* $q \notin Req(Voc)$*, return* $(r, \bar{o}) := (scope\_error, \emptyset)$.
2. Processing by precedence. *For each precedence level* $i := max(Pol)$ *down to* $min(Pol)$*:*
   - Accumulate obligations. $\bar{o}_{acc} := \bigcup_{(i,u,d,p,a,c,\bar{o},r) \in AR(Pol,q,\chi)} \bar{o}$
   - Normal ruling. *If some rule* $(i, u, d, p, a, c, \bar{o}, r) \in AR(Pol, q, \chi)$ *exists, return* $(r, \bar{o}_{acc})$.
3. Default ruling. *If this step is reached, return* $(r, \bar{o}) := (dr, \emptyset)$.

We also say that policy Pol rules $(r, \bar{o})$ *for* q *and* χ*, omitting* q *and* χ *if they are clear from the context.* ◇

2.4 Refinement and Equivalence of Well-Formed Privacy Policies

Basically, refining a policy Pol means adding more details to it, i.e., enriching the vocabulary and/or the set of rules without changing the meaning of the policy with respect to its original vocabulary. To be useful for actual use cases, it is essential that operators defined on privacy policies behave in a well-specified and an "intuitive" manner with respect to refinement relations. Thus, before we can make concrete statements about the refinement properties of the operators introduced in the next section, we need some additional terminology, and end this section with recalling some definitions from [4].

Definition 12 (Compatible Vocabulary). *Two vocabularies* Voc_1 *and* Voc_2 *are* compatible *if their condition vocabularies are compatible and* $UH_1 \cup UH_2, DH_1 \cup DH_2, PH_1 \cup PH_2, AH_1 \cup AH_2$ *are hierarchies again.* ◇

The notion of compatible vocabularies is a technicality that turns out to be necessary to specify operations that combine different policies which are not necessarily formulated in terms of identical vocabularies, and this leads to

Definition 13 (Union of Vocabularies). *The* union *of two compatible vocabularies* Voc_1 *and* Voc_2 *is defined as* $Voc_1 \cup Voc_2 := (UH_1 \cup UH_2, DH_1 \cup DH_2, PH_1 \cup PH_2, AH_1 \cup AH_2, Var_1 \cup Var_2, OM)$*, where* $OM = (O, \rightarrow_O)$ *is the obligation model with the lattice* $(\mathfrak{P}(O), \vee, \cup)$ *being generated by* $\mathfrak{P}(O_1)$ *and* $\mathfrak{P}(O_2)$*, and* $\rightarrow_O$ *being the restriction of* $\rightarrow_{O_0}$ *to* $\mathfrak{P}(O) \times \mathfrak{P}(O)$. ◇

Next, we need the refinement of obligations whose definition requires some care, as a refined policy may well contain additional obligations, whereas at the same time some others have been omitted. As consequence of this observation, the definition of refinement of obligations makes use of both obligation models, that of the original (coarser) policy and that of the refined policy:

Definition 14 (Refinement and Equivalence of Obligations). *Let two obligation models $(O_i, \rightarrow_{O_i})$ and $\bar{o}_i \subseteq O_i$ for $i = 1, 2$ be given. Then $\bar{o}_2$ is a* refinement *of $\bar{o}_1$, written $\bar{o}_2 \prec \bar{o}_1$ if and only if the following holds:*

$$\exists \bar{o} \subseteq O_1 \cap O_2 : \bar{o}_2 \rightarrow_{O_2} \bar{o} \rightarrow_{O_1} \bar{o}_1.$$

We call $\bar{o}_1$ and $\bar{o}_2$ equivalent*, written $\bar{o}_1 \equiv \bar{o}_2$, if and only if $\bar{o}_1 \prec \bar{o}_2$ and $\bar{o}_2 \prec \bar{o}_1$. For $r_1, r_2 \in \{+, -, \circ, scope\_error\}$, we further define $(r_1, \bar{o}_1) \equiv (r_2, \bar{o}_2)$ if and only if $r_1 = r_2$ and $\bar{o}_1 \equiv \bar{o}_2$.* ◇

We can now formalize the notion of (weak) refinement of well-formed policies. Our definition of refinement closely resembles the one presented in [4], but it excludes partial assignments and conflict errors, which are not supported by the latest EPAL version. The notion of weak refinement has not been introduced before.

Definition 15 (Policy Refinement). *Let two well-formed privacy policies $Pol_i = (Voc_i, R_i, dr_i)$ for $i = 1, 2$ with compatible vocabularies be given, and set $Pol_i^* = (Voc_i^*, R_i, dr_i)$ for $i = 1, 2$ where $Voc_i^* := (UH_1 \cup UH_2, DH_1 \cup DH_2, PH_1 \cup PH_2, AH_1 \cup AH_2, Var_i, OM_i)$.*

Let $r_1, r_2 \in \{+, -, \circ, scope\_error\}$ and $\bar{o}_i \subseteq O_i$ for $i = 1, 2$ be arbitrary. We say that $(r_2, \bar{o}_2)$ refines *$(r_1, \bar{o}_1)$ (in OM_1 and OM_2), written $(r_2, \bar{o}_2) \prec (r_1, \bar{o}_1)$, if and only if one of the following two conditions holds*

$$(1) \quad (r_1, \bar{o}_1) \in \{(scope\_error, \emptyset), (\circ, \emptyset)\} \quad (2) \quad r_1 \in \{+, -\}, r_2 = r_1, \bar{o}_2 \prec \bar{o}_1.$$

We say that $(r_2, \bar{o}_2)$ weakly refines *$(r_1, \bar{o}_1)$ (in OM_1 and OM_2), written $(r_2, \bar{o}_2) \tilde{\prec} (r_1, \bar{o}_1)$, if and only if one of the following three conditions holds:*

$$(1) \quad (r_2, \bar{o}_2) \prec (r_1, \bar{o}_1) \quad (2) \quad r_1 = +, r_2 = - \quad (3) \quad (r_1, \bar{o}_1) = (+, \emptyset), r_2 = \circ.$$

We call Pol_2 a refinement *of Pol_1, written $Pol_2 \prec Pol_1$ if and only if for every assignment $\chi \in \mathfrak{Ass}(Var_1 \cup Var_2)$ and every authorization request $q \in Req$, we have $eval_{Pol_2^*}(q, \chi) \prec eval_{Pol_1^*}(q, \chi)$. We call Pol_2 a* weak refinement *of Pol_1 if the same holds with $\prec$ replaced by $\tilde{\prec}$.* ◇

Intuitively, a privacy policy that weakly refines another policy is at least as restrictive as the coarser one: Even if the original policy rules "allow" for a certain request, after a weak refinement the same request may be denied, or – provided that no obligations get lost – an "allow" can be transformed into a "don't care".

Finally, the equivalence of two well-formed privacy policies is defined in the obvious manner:

Definition 16 (Policy Equivalence). *Two well-formed privacy policies* Pol_1 *and* Pol_2 *are called* equivalent*, written* $Pol_1 \equiv Pol_2$*, if and only if they are mutual refinements, i.e.,* $Pol_1 \equiv Pol_2 :\Leftrightarrow (Pol_1 \prec Pol_2 \wedge Pol_2 \prec Pol_1)$. ◇

While this notion of policy equivalence is rather intuitive, it turns out that in some situations only a weaker form of equivalence can be achieved, and we therefore conclude this section with the definition of weak policy equivalence.

Definition 17 (Weak Policy Equivalence). *Two well-formed privacy policies* Pol_1 *and* Pol_2 *are called* weakly equivalent*, written* $Pol_1 \approx Pol_2$*, if and only if they are equivalent on their joint vocabulary, i.e., if and only if* $(Voc_1 \cup Voc_2, R_1, dr_1) \equiv (Voc_1 \cup Voc_2, R_2, dr_2)$. ◇

3 Defining Operators

Basically, defining symmetric operations on privacy policies reflecting the intuitive notions of conjunction (AND) and disjunction (OR) looks rather simple. Unfortunately, with a straightforward yet intuitive approach it happens that the conjunction or disjunction of two privacy policies might no longer constitute a syntactically correct privacy policy. From a practical point of view such a behavior is not desirable: First, available tools to enforce a(n EPAL) privacy policy are designed to handle privacy policies only. Thus, to handle compositions of privacy policies these tools had to be modified or new tools had to be developed. The obvious solution to this problem – making use of a wrapper program that queries several policies by means of existing tools and combines their results appropriately – is not always acceptable. In particular such a workaround might violate conditions that were necessary to pass some (expensive) certification process. Secondly, the combined privacy policies can originate in rather different sources which are separated though significant geographical distances. Consequently, in larger, say multinational, projects, where policies of many different organizations have to be combined, it can be infeasible or at least very inconvenient to store all (component) policies that contribute to the ruling of the composition.

To circumvent these problems, it is desirable to work in a subset of $EP3P$ that is on the one hand closed under conjunction and disjunction as well as other suitable algebraic operations, and on the other hand is still expressive enough to capture typically used privacy policies. The following lemma, whose proof we omit due to lack of space, characterizes the expressiveness (and therewith also limits) of E-P3P policies.

Lemma 1 (Expressiveness of E-P3P). *Let* Voc *be a vocabulary and* $\varphi\colon Req(Voc) \times \mathfrak{Ass}(Var) \to \{+, \circ, -\} \times \mathfrak{P}\{O\}$ *be an arbitrary function. Then there exists a well-formed privacy policy* $Pol = (Voc, R, dr)$ *with* $eval_{Pol}(q, \chi) = \varphi(q, \chi)$ *for all* $(q, \chi) \in Req(Voc) \times \mathfrak{Ass}(Var)$ *if and only if for all valid requests* $q \in Req(Voc)$ *and all assignments* $\chi \in \mathfrak{Ass}(Var)$*, the following four conditions are satisfied:*

1. $\varphi(q, \chi) = (+, \bar{o}) \Rightarrow \forall q' \leq q : \varphi(q', \chi) = (+, \bar{o}')$ *(possibly with* $\bar{o}' \neq \bar{o}$*).*
2. $\varphi(q, \chi) = (-, \bar{o}) \Rightarrow \forall q' \geq q : \varphi(q', \chi) = (-, \bar{o}')$ *(possibly with* $\bar{o}' \neq \bar{o}$*).*

3. $\varphi(q,\chi) = (-,\bar{o})$ *implies that one of the following conditions holds:*
   (a) $q \in L(Voc)$,
   (b) $\exists q' <_1 q : \varphi(q',\chi) = (+,\bar{o}')$ *(possibly with* $\bar{o}' \neq \bar{o}$*),*
   (c) $\exists C \subseteq \{q' <_1 q, \varphi(q',\chi) = (-,\bar{o}_{q'})\} \colon C \neq \emptyset \wedge \bar{o}' = \bigcup_{q' \in C} \bar{o}_{q'}$.
4. *If* $\varphi(q,\chi) = (\circ,\bar{o})$*, then* $\bar{o} = \emptyset$. □

3.1 Conjunction, Disjunction and the Non-closedness of EPAL

Unlike in typical access control settings, for defining the conjunction and disjunction of privacy policies, we have to take care of the "don't care" ruling $\circ$, whose semantics is different from both "allow" and "deny". Motivated by the intuition behind the ruling $\circ$, we decided for definitions that are in analogy to the conjunction and disjunction in a three-valued Łukasiewicz logic $Ł_3$. To handle the obligations, we use the operator $\vee$ provided by the obligation model.

AND	$(+,\bar{o}')$	$(-,\bar{o}')$	$(\circ,\emptyset)$
$(+,\bar{o})$	$(+,\bar{o}\cup\bar{o}')$	$(-,\bar{o}')$	$(\circ,\emptyset)$
$(-,\bar{o})$	$(-,\bar{o})$	$(-,\bar{o}\cup\bar{o}')$	$(-,\bar{o})$
$(\circ,\emptyset)$	$(\circ,\emptyset)$	$(-,\bar{o}')$	$(\circ,\emptyset)$

OR	$(+,\bar{o}')$	$(-,\bar{o}')$	$(\circ,\emptyset)$
$(+,\bar{o})$	$(+,\bar{o}\vee\bar{o}')$	$(+,\bar{o})$	$(+,\bar{o})$
$(-,\bar{o})$	$(+,\bar{o}')$	$(-,\bar{o}\vee\bar{o}')$	$(\circ,\emptyset)$
$(\circ,\emptyset)$	$(+,\bar{o}')$	$(\circ,\emptyset)$	$(\circ,\emptyset)$

Intuitively, we do not want to give a positive answer to a request if one of the two policies that are to be combined by AND denies the access. Further on, if one policy allows the access, and the other one "does not care", then returning a "don't care" seems plausible and is indeed needed to ensure the distributivity of the operators AND and OR. Similarly, for OR we allow an access, if at least one of the two involved policies allows the request. Moreover, we "do not care", if one of the operands "does not care" – except if the other operand explicitly "allows" the request.

Lemma 2. *Fix some obligation model and denote by* $(\mathfrak{P}(O),\vee,\cup)$ *the corresponding lattice of obligations. Then* $((\{+,-\}\times\mathfrak{P}(O))\cup\{(\circ,\emptyset)\},\mathrm{OR},\mathrm{AND})$ *is a distributive lattice.* □

We omit the proof of this and most of the subsequent lemmas due to space limitations and refer the reader to the long version of this paper [3].

The natural definition of conjunction of two privacy policies Pol_1 and Pol_2 would be that whenever Pol_i rules $(r_i,\bar{o}_i)$ for a given assignment and request, then the conjunction of Pol_1 and Pol_2 should yield $(r_1,\bar{o}_1)$ AND $(r_2,\bar{o}_2)$, and similar for disjunction. However, an easy corollary of Lemma 1 yields that such a policy is not necessarily a valid EPAL policy anymore, i.e., EPAL is neither closed under conjunction nor under disjunction given the above definitions.

Corollary 1 (Non-closedness of EPAL). *There exist policies* Pol_1, Pol_2 *such that for any policy* Pol*, we have that there exists an assignment* $\chi \in \mathfrak{Ass}(Var_1 \cup Var_2)$ *and a request* $q \in Req(Voc_1 \cup Voc_2)$ *such that* $eval_{Pol}(q,\chi) \not\equiv eval_{Pol_1}(q,\chi)$ op $eval_{Pol_2}(q,\chi)$ *where* op $\in \{\mathrm{AND},\mathrm{OR}\}$. □

Proof. For showing the statement for conjunction, we consider the policies depicted at the left-hand side of Figure 1. Let Pol_i for $i = 1,2$ such that each of DH_i, PH_i, AH_i

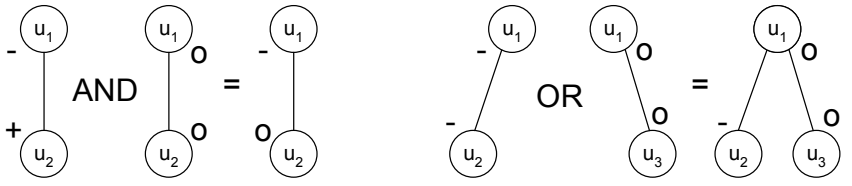

Fig. 1. Examples that EPAL is not closed under conjunction or disjunction.

consists of a single element, and let UH_i contain two elements u_1, u_2 such that u_1 is a parent of u_2. Further assume that the condition vocabulary and the obligation model are empty. The rules in Pol_1 allow user u_2 to access the data whereas u_1 is forbidden to do so. The rules in Pol_2 don't care for both users. The conjunction would then yield a policy in which u_1 is not allowed to access the data whereas the policy does not care for u_2. This immediately yields a contradiction to Lemma 1. The claim can be shown similarly for disjunction and the policies depicted at the right-hand side of Figure 1. ■

It is is easy to see that adapting the definition of AND and OR in obvious ways (like redefining the occurrences of $\circ$) does not solve this problem without violating other essential conditions, e.g., the distributivity of the operators. As a remedy we identify the subset of *well-founded* privacy policies in the next section, which allows for a very intuitive handling in terms of defining conjunction and disjunction of privacy policies. Actually, for practical cases, the restriction to those privacy policies is not really an obstacle, and in the next section we take a closer look at such policies.

3.2 Well-Founded Privacy Policies

The intuition underlying the notion of well-founded policies can be described as follows:

- Suppose the ruling specified for some group is "deny", but none of the group members is denied from accessing the respective data. Then this contradicts the idea that in EPAL the group ruling is to reflect ("to group") the rulings of the individual group members.
- If each member of a group is permitted to perform some action, then intuitively the group as a whole is permitted to perform this action, too.
- Assume that both the ruling specified for a group and for a member of this group is "allow", and assume further that the obligations of the group are not a superset of the obligations of the group member. Then the group member may be able to avoid certain obligations by submitting a query where the user is specified to be the group as a whole. Typically, the availability of such a "workaround" is not desirable. On the other hand, if the obligations of the group are stricter than the union of the obligations of the group members and we (re)define the group obligations to be the union of the individual obligations then no harm (in the sense that a group member can gain additional privileges) is caused by querying the group.

Formally, well-founded policies are captured as follows:

Definition 18 (Well-Founded Policy). *Let Pol be a well-formed policy. Then we call Pol* well-founded *if and only if for all $(q,\chi) \in Req(Voc) \times \mathfrak{Ass}(Var)$ the following conditions are fulfilled:*

- *If q is no leaf node and $eval_{Pol}(q,\chi) = (-,\bar{o})$, then there exists $q' <_1 q$ such that $eval_{Pol}(q',\chi) = (-,\bar{o}')$ for some $\bar{o}'$.*
- *If $eval_{Pol}(q',\chi) = (+,\bar{o}_{q'})$ for each $q' <_1 q$ and arbitrary $\bar{o}_{q'}$, then $eval_{Pol}(q,\chi) = (+,\bar{o})$ for some $\bar{o}$.*
- *If $eval_{Pol}(q,\chi) = (r,\bar{o})$, then $\bar{o} = \bigcup_{q'<_1 q, eval_{Pol}(q',\chi)=(r,\bar{o}')} \bar{o}'$.* ◇

Up to equivalence, well-founded policies are already uniquely determined by the rulings of the leaf nodes:

Lemma 3. *Let Pol_1, Pol_2 be well-founded privacy policies with $Voc_1 = Voc_2$ and let $eval_{Pol_1}(q,\chi) = eval_{Pol_2}(q,\chi)$ for every $q \in L(Voc_1)$ and every $\chi \in \mathfrak{Ass}(Var_1)$. Then $Pol_1 \equiv Pol_2$.* □

Actually, the predetermined allow and deny rulings for the set of leaf nodes can be chosen arbitrarily. The subsequent algorithm demonstrates how in principle a well-founded policy can explicitly be written down that is consistent with any predetermined set of rulings for all leaf nodes. Note however that the algorithm does not aim at generating small policies; optimizing it for practical purposes is considered as future work.

Input: • a vocabulary Voc and
• a ruling $(r_{q,\chi}, \bar{o}_{q,\chi}) \in (\{+,-\} \times \mathfrak{P}(O)) \cup \{(\circ,\emptyset)\}$ for all $q \in L(Voc), \chi \in \mathfrak{Ass}(Var)$.

Output: a well-founded privacy policy $Pol = (Voc, R, dr)$ such that for all $q \in L(Voc), \chi \in \mathfrak{Ass}(Var)$ the equality $eval_{Pol}(q,\chi) = (r_{q,\chi}, \bar{o}_{q,\chi})$ holds.

/* Assign identical precedences to leaf rulings different from $(\circ,\emptyset)$ */
$R := \emptyset$
for each $q := (u,d,p,a) \in L(Voc)$
  if $(r_{q,\chi}, \bar{o}_{q,\chi}) \neq (\circ,\emptyset)$
    then $R := R \cup \{(0,u,d,p,a,c_\chi,\bar{o}_{q,\chi},r_{q,\chi})\}$ (1)
  end if
end for

/* Insert missing positive rulings with low precedence */
for each $\chi \in \mathfrak{Ass}(Var)$ and each $q := (u,d,p,a) \in Req(Voc)$ do
  if $r_{q',\chi} = +$ for all $q' \in L(q, Voc)$
    then $R := R \cup \{(i,u,d,p,a,c_\chi, \bigcup_{q' \in L(q,Voc)} \bar{o}_{q',\chi})\}$ (2)
      such that $i < i'$
      for all $(i',u',d',p',a',c',+,\bar{o}') \in R : q > (u',d',p',a')$
  end if
end for

return $(Voc, R, \circ)$

Lemma 4. *The above algorithm is totally correct, i.e., it terminates and for inputs as specified in the algorithm, it computes a policy as specified in the output description.* □

3.3 Conjunction and Disjunction of Privacy Policies

We now define the conjunction and disjunction of two well-founded privacy policies. From Lemma 3 we know that it is sufficient to define the operations for those requests that are leaves of the considered hierarchies since once the evaluations on the leaves are fixed, the corresponding privacy policy is, up to equivalence, uniquely determined. With the algorithm in Section 3.2 we can then explicitly compute a policy that is consistent with the given evaluations of the leaf nodes. However, to make definitions of the operators independent of an algorithmic specification, we will formulate the actual definitions in such a way that the result of a conjunction/disjunction of two privacy policies constitutes an equivalence class of policies – not a specific privacy policy. For practical purposes this is not really a problem as we can, e.g., use the algorithm from Section 3.2 to derive a concrete policy from the equivalence class.

The motivation for defining an AND operation on privacy policies is rather straightforward: Assume that an enterprise takes part in some project for which data has to be accessed and processed that is controlled by some external project partner. Then the access to and processing of such data shall only be allowed, if none of the individual privacy policies of the participating enterprises is violated.

Definition 19 (Policy Conjunction). *Let Pol_1, Pol_2 be two well-founded privacy policies such that $Pol_i^* = (Voc_i^*, R_i, dr_i)$ for $i = 1, 2$ with $Voc_i^* := (UH_1 \cup UH_2, DH_1 \cup DH_2, PH_1 \cup PH_2, AH_1 \cup AH_2, Var_i, OM_i)$ are also well-founded privacy policies.*

Then the conjunction *of Pol_1 and Pol_2, is the equivalence class (w. r. t. $\equiv$) of all well-founded privacy policies Pol on the joint vocabulary $Voc := Voc_1 \cup Voc_2$ such that for all leaf nodes $q \in L(Voc)$ and for all assignments $\chi \in \mathfrak{Ass}(Var)$ we have $(r_1, \bar{o}_1) \equiv (r_2, \bar{o}_2)$, where*

$$(r_1, \bar{o}_1) := eval_{Pol}(q, \chi) \text{ and}$$
$$(r_2, \bar{o}_2) := eval_{Pol_1^*}(q, \chi) \text{ AND } eval_{Pol_2^*}(q, \chi).$$

By $Pol_1 \& Pol_2$ we denote any representative of this equivalence class (which can, e.g., be computed by means of the algorithm in Section 3.2). ◇

Note that this definition only imposes conditions on the leaf nodes, hence the question arises to what extent "inner" queries obey the defining table for AND, too. Indeed, the desired relations are fulfilled for arbitrary queries:

Lemma 5. *Let Pol_1, Pol_2 be well-founded privacy policies that satisfy the requirements of Definition 19 and let $Pol = Pol_1 \,\&\, Pol_2$. Then for all requests $q \in Req(Voc)$ and for all assignments $\chi \in \mathfrak{Ass}(Var)$ we have the equivalence $eval_{Pol}(q, \chi) \equiv eval_{Pol_1^*}(q, \chi)$ AND $eval_{Pol_2^*}(q, \chi)$ with Pol_i^* as in Definition 19.* □

Similar to conjunction, the disjunction of privacy policies is essential for a variety of use cases. For example, consider two departments of an enterprise that cooperate in some project. For carrying out this project, it should then be possible to access data items

whenever one of the individual privacy policies of the two departments grants such an access. This idea of "joining forces" is captured by the following definition.

Definition 20 (Policy Disjunction). *Let Pol_1, Pol_2 be two well-founded privacy policies such that $Pol_i^* = (Voc_i^*, R_i, dr_i)$ for $i = 1, 2$ with $Voc_i^* := (UH_1 \cup UH_2, DH_1 \cup DH_2, PH_1 \cup PH_2, AH_1 \cup AH_2, Var_i, OM_i)$ are also well-founded privacy policies.*

Then the disjunction *of Pol_1 and Pol_2 is the equivalence class (w. r. t. $\equiv$) of all well-founded privacy policies Pol on the joint vocabulary $Voc := Voc_1 \cup Voc_2$ such that for all leaf nodes $q \in L(Voc)$ and for all assignments $\chi \in \mathfrak{Ass}(Var)$ we have $(r_1, \bar{o}_1) \equiv (r_2, \bar{o}_2)$ where*

$$(r_1, \bar{o}_1) := eval_{Pol}(q, \chi) \text{ and}$$
$$(r_2, \bar{o}_2) := eval_{Pol_1^*}(q, \chi) \text{ OR } eval_{Pol_2^*}(q, \chi).$$

By $Pol_1 + Pol_2$ we denote any representative of this equivalence class (which can, e.g., be computed by means of the algorithm in Section 3.2). ◇

Unfortunately, for the disjunction of privacy policies, we have no analogue to Lemma 5, i.e., in general we cannot achieve an equivalence of the form $eval_{Pol}(q, \chi) \equiv eval_{Pol_1^*}(q, \chi)$ OR $eval_{Pol_2^*}(q, \chi)$ for arbitrary requests q and assignments χ. In fact, it is not difficult to construct examples where imposing such a "node-wise equivalence" yields a contradiction to well-foundedness. Fortunately, also for the "inner nodes" the policy obtained by disjunction is still rather close to what one would expect intuitively:

Lemma 6. *Let Pol_1, Pol_2 be well-founded privacy policies that satisfy the requirements of Definition 19 and let $Pol = Pol_1 + Pol_2$. Then for all $q \in Req(Voc)$ such that $eval_{Pol}(q, \chi) = (-, \bar{o})$ or $eval_{Pol_1^*}(q, \chi)$ OR $eval_{Pol_2^*}(q, \chi) = (+, \bar{o})$ holds for some $\bar{o}$, we have $eval_{Pol_1^*}(q, \chi)$ OR $eval_{Pol_2^*}(q, \chi) \prec eval_{Pol}(q, \chi)$.* □

3.4 Scoping of a Privacy Policy

One of the most desirable operations in practice is to restrict the scope of a policy, i.e., to restrict large policies to smaller parts. Examples for this so-called *scoping* are omnipresent in practical policy management, e.g., deriving a department's privacy policy from the enterprise's global privacy policy, or considering only those rules that specifically deal with marketing purposes. Formally, we define the following scoping operator:

Definition 21 (Scoping). *Let Pol be a well-founded privacy policy and let $V := (UH', DH', PH', AH')$ where UH', DH', PH', andAH' are arbitrary subhierarchies of UH, DH, PH, AH, respectively.*

Then the scoping *of Pol with respect to V is the equivalence class (with respect to $\equiv$) of all well-founded privacy policies Pol' on the vocabulary $Voc' := (UH', DH', PH', AH', Var, OM)$ such that for all leaves $q \in L(Voc')$ and for all assignments $\chi \in \mathfrak{Ass}(Var)$ we have*

$$eval_{Pol'}(q, \chi) \equiv eval_{Pol}(q, \chi).$$

By $Pol|_V$ we denote any representative of this equivalence class (which can, e.g., be computed by means of the algorithm specified in Section 3.2). ◇

Ideally, scoping would yield a privacy policy such that not only for the leaf nodes, but also for the "inner" requests we always obtain equivalent rulings from Pol and $Pol|_V$. However, in general this contradicts the well-foundedness of the privacy policy derived via scoping unless additional assumptions on the considered hierarchies are imposed:

Example 1. Consider a well-founded privacy policy Pol such that each of DH, PH, AH consists of a single element. For the sake of simplicity assume that the condition vocabulary is empty, and let UH contain three users u_0, u_1, u_2. The rules in Pol allow user u_1 to access the data with obligations $\bar{o}_1$, and user u_2 can access the data with obligations $\bar{o}_2$. Finally the "superuser" u_0 – the parent of u_1 and u_2 in the user hierarchy UH – can access the data with obligations $\bar{o}_1 \cup \bar{o}_2$ to ensure the well-foundedness of Pol. If we scope this policy w. r. t., $(\{u_0, u_1\}, DH, PH, AH)$, then user u_1 can still access the data with obligations $\bar{o}_1$, but due to the well-foundedness of $Pol|_{(\{u_0,u_1\},DH,PH,AH)}$, now the superuser u_0 can also access the data with obligations $\bar{o}_1$, in other words the obligations $\bar{o}_2$ are "lost".

However, even without imposing additional constraints on the considered hierarchies, we can exploit the well-foundedness of the policies to establish the following lemma:

Lemma 7. *Let Pol be a well-founded privacy policy with vocabulary Voc and $V := (UH', DH', PH', AH')$ a tuple of subhierarchies of UH, DH, PH, AH, respectively. Then $Pol \precsim Pol|_V$.* □

In dependence on the precise kind of scoping considered, even stronger preservation properties can be proven, e.g., if the scoped policy is well-founded again then we obtain equivalent rulings also for inner requests. This is the case if we for instance apply a scoping operation under which the leaf requests are invariant, i.e., if the considered vocabularies are non-appending. Albeit this looks rather restrictive from a theoretical point of view, for the kind of scoping needed in practice – say extracting a department's privacy policy from an enterprise's privacy policy – this requirement is often met.

Lemma 8. *Let Pol be a well-founded privacy policy, and $V := (UH', DH', PH', AH')$ a tuple of subhierarchies of UH, DH, PH, AH, respectively, such that for $Voc' := (UH', DH', PH', AH', Var, OM)$ and all $q \in Req(Voc')$, we have $L(q, Voc) = L(q, Voc')$. Then for all $q \in Req(Voc')$ and for all assignments $\chi \in \mathfrak{Ass}(Var)$ we have $eval_{Pol}(q, \chi) \equiv eval_{Pol|_V}(q, \chi)$.* □

3.5 Further Extensions of the Algebra

There are certainly further operators one would like to add to the set of available tools. From a practical point of view, it is in particular desirable that the operators discussed in this paper can be combined with the sequential form of composition of E-P3P policies proposed in [4]. Since we try to stay close to the latest version of EPAL that has been submitted to W3C for standardization, the E-P3P variant underlying [4] is slightly different from the variant that we consider here, hence some care has to be taken in combining operations/results from [4] and the ones presented above. Fortunately, carrying over the operator for sequentially composing privacy policies from [4] – there called *ordered composition* – to the situation considered here is straightforward.

As a technical tool, [4] introduces *policies with removed default rulings*, which means that an E-P3P policy is transformed into an equivalent one with default ruling "don't care". However, from Lemma 3 we know how to represent any well-founded privacy policy in this way, and so we can do without this technical trick here. As in [4] we make use of the concept of *precedence shift*, which adds a fixed number to the precedences of all rules in a policy. This can be used, for instance, to shift a department policy downwards, so that it has lower precedences than the CPO's privacy policy.

Definition 22 (Precedence Shift). *Let* $Pol = (Voc, R, dr)$ *be a privacy policy and* $j \in \mathbb{Z}$. *Then* $Pol + j := (Voc, R + j, dr)$ *with* $R + j := \{(i + j, u, d, p, a, c, \bar{o}, r) \mid (i, u, d, p, a, c, \bar{o}, r) \in R\}$ *is called the* precedence shift *of* Pol *by* j. *We define* $Pol - j := Pol + (-j)$. ◇

To formalize the sequential composition of two well-founded policies Pol_1, Pol_2 with compatible vocabularies, we assume that both of them have a "don't care" default ruling. If this is not the case, we first apply an algorithm like the one in Section 3.2 to derive equivalent privacy policies which have a "don't care" default ruling. After that, we shift the two policies accordingly, and then join their vocabularies and rulesets:

Definition 23 (Sequential Composition). *Let* Pol_1, Pol_2 *be well-founded privacy policies with compatible vocabularies, where w. l. o. g.* $dr_1 = \circ = dr_2$. *Let* $(Voc_1, R_1'', \circ) := Pol_1 - max(Pol_1) - 1$ *and* $(Voc_2, R_2', \circ) := Pol_2 - min(Pol_1) + 1$. *Then*

$$Pol_1 \bigcup\!\!\!\!\!\!^{\prec} \; Pol_2 := (Voc_1 \cup Voc_2, R_1'' \cup R_2', \circ)$$

is called the sequential composition *of* Pol_1 *under* Pol_2. ◇

Intuitively, a sequential composition of Pol_1 under Pol_2 should serve as a refinement of Pol_2 which is formally captured in the following lemma.

Lemma 9. *For all well-founded privacy policies* Pol_1 *and* Pol_2 *with compatible vocabularies and* $dr_1 = \circ = dr_2$, *we have* $Pol_1 \bigcup\!\!\!\!\!\!^{\prec} \; Pol_2 \prec Pol_2$. □

Obviously, the sequential composition of two well-founded privacy policies is in general no longer well-founded. So when combining $\bigcup\!\!\!\!\!\!^{\prec}$ with the operators $+$ and $\&$ to form more complex privacy policies, some care has to be taken. In general, the sequential composition of policies should always be the last operation applied, as it is the only one which does not preserve well-foundedness.

4 Algebraic Properties of the Operators

Since the operator definitions proposed in the previous section are quite intuitive, one would not expect "unpleasant surprises" when using these operators to form more complex privacy policies involving three, four, or more operands. As actual use cases often involve more than only one or two different privacy policies, we have to ensure that our operators do not yield non-intuitive behaviors in such scenarios. Fortunately, this is not the case, and the usual algebraic laws apply:

Lemma 10. *Let Pol_1, Pol_2, Pol_3 be well-founded E-P3P policies such that the following expressions are well-defined, i.e., the respective requirements in Definition 19 respectively Definition 20 are met. Then the following holds:*

$$\begin{aligned}
\textit{Idempotency} &: Pol_1 \& Pol_1 \equiv Pol_1, && (1)\\
&\quad\ Pol_1 + Pol_1 \equiv Pol_1,\\
\textit{Commutativity} &: Pol_1 \& Pol_2 \equiv Pol_2 \& Pol_1, && (2)\\
&\quad\ Pol_1 + Pol_2 \equiv Pol_2 + Pol_1,\\
\textit{Associativity} &: Pol_1 \& (Pol_2 \& Pol_3) \equiv (Pol_1 \& Pol_2) \& Pol_3, && (3)\\
&\quad\ Pol_1 + (Pol_2 + Pol_3) \equiv (Pol_1 + Pol_2) + Pol_3,\\
\textit{Distributivity} &: Pol_1 + (Pol_2 \& Pol_3) \equiv (Pol_1 + Pol_2) \& (Pol_1 + Pol_3), && (4)\\
&\quad\ Pol_1 \& (Pol_2 + Pol_3) \equiv (Pol_1 \& Pol_2) + (Pol_1 \& Pol_3),\\
\textit{Strong Absorption} &: Pol_1 + (Pol_1 \& Pol_2) \prec Pol_1. && (5)
\end{aligned}$$

□

It is worth noting that our proof of the strong absorption property relies on both Lemma 5 and Lemma 6, and although it may look tempting, one cannot simply switch the roles of conjunction and discjunction in the proof to derive a "dual" strong absorption law with the roles of $\&$ and $+$ being exchanged.

In addition to purely algebraic properties of the operators, one can also establish several refinement results. In particular we can prove the following relations, which from the intuitive point of view are highly desirable:

Lemma 11. *Let Pol_1, Pol_2 be well-founded privacy policies such that the respective requirements of Definition 19 and Definition 20 are met. Then we have*

$$\begin{aligned}
\textit{Weak Multiplicative Refinement} &: Pol_1 \& Pol_2 \tilde{\prec} Pol_i \quad (i = 1, 2), && (6)\\
\textit{Weak Additive Refinement} &: Pol_i \tilde{\prec} Pol_1 + Pol_2 \quad (i = 1, 2). && (7)
\end{aligned}$$

□

Finally, we state a refinement result which relates the sequential composition operator to the operators for conjunction and disjunction:

Lemma 12. *Let Pol_1, Pol_2 be well-founded policies such that the respective requirements of Definition 19, Definition 20, and Definition 23 are met. Then we have*

$$\textit{Weak Operator Refinement} : Pol_1 \& Pol_2 \tilde{\prec} Pol_1 \bigcup\!\!\!\!\!\!\prec\ Pol_2 \tilde{\prec} Pol_1 + Pol_2. \quad (8)$$

□

5 Conclusion

Motivated by the need for practical life-cycle management systems for the collection of enterprise privacy policies, we have introduced several algebraic operators for combining enterprise privacy policies, and we have shown that they enjoy the expected algebraic laws. Our operators allow for a convenient, modular use of existing EPAL policies

as building blocks for new ones, and they hence avoid the difficulties that naturally arise for the usually very complex monolithic privacy specifications.

An analysis of the expressiveness of EPAL further revealed that, somewhat surprisingly, EPAL policies are not closed under intuitive notions of policy conjunction and policy disjunction; however, such operations are crucial for actual use cases. We have circumvented this problem by identifying a suitable subclass of EPAL policies that is closed under desired algebraic operations. Further on, the introduced tools for combining privacy policies satisfy natural requirements like associativity, commutativity, and distributivity, as well as appropriate refinement relations. In addition to conjunction and disjunction operators, our algebra provides a scoping operation which allows for managing and reasoning about privacy requirements that involve only certain parts of an organization. Finally, we have shown that the already existing notion of sequential composition of privacy policies fits naturally into our setting. As future work we consider it a worthwhile goal to add further operations to our algebra in order to further facilitate a convenient handling of privacy policies.

References

1. P. Ashley, S. Hada, G. Karjoth, C. Powers, and M. Schunter. Enterprise Privacy Authorization Language (EPAL). Research Report 3485, IBM Research, 2003.
2. P. Ashley, S. Hada, G. Karjoth, and M. Schunter. E-P3P privacy policies and privacy authorization. In *Proc. 1st ACM Workshop on Privacy in the Electronic Society (WPES)*, pages 103–109, 2002.
3. M. Backes, M. Dürmuth, and R. Steinwandt. An algebra for composing enterprise privacy policies. Research Report 3557, IBM Research, 2004.
4. M. Backes, B. Pfitzmann, and M. Schunter. A toolkit for managing enterprise privacy policies. In *Proc. 8th European Symposium on Research in Computer Security (ESORICS)*, volume 2808 of *LNCS*, pages 162–180. Springer, 2003.
5. C. Bettini, S. Jajodia, X. S. Wang, and D. Wijesekerat. Obligation monitoring in policy management. In *Proc. 3rd IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY)*, pages 2–12, 2002.
6. P. A. Bonatti, E. Damiani, S. De Capitani di Vimercati, and P. Samarati. A component-based architecture for secure data publication. In *Proc. 17th Annual Computer Security Applications Conference*, pages 309–318, 2001.
7. P. A. Bonatti, S. De Capitani di Vimercati, and P. Samarati. A modular approach to composing access control policies. In *Proc. 7th ACM Conference on Computer and Communications Security*, pages 164–173, 2000.
8. P. A. Bonatti, S. de Capitani di Vimercati, and P. Samarati. An algebra for composing access comtrol policies. *ACM Transactions on Information and System Security*, 5(1):1–35, 2002.
9. A. Cavoukian and T. J. Hamilton. *The Privacy Payoff: How successful businesses build customer trust*. McGraw-Hill/Ryerson, 2002.
10. S. De Capitani di Vimercati and P. Samarati. An authorization model for federated systems. In *Proc. 4th European Symposium on Research in Computer Security (ESORICS)*, volume 1146 of *LNCS*, pages 99–117. Springer, 1996.
11. S. Fischer-Hübner. *IT-security and privacy: Design and use of privacy-enhancing security mechanisms*, volume 1958 of *LNCS*. Springer, 2002.

12. Z. Fu, S. F. Wu, H. Huang, K. Loh, F. Gong, I. Baldine, and C. Xu. IPSec/VPN security policy: Correctness, conflict detection and resolution. In *Proc. 2nd IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY)*, volume 1995 of *LNCS*, pages 39–56. Springer, 2001.
13. V. D. Gligor, S. I. Gavrila, and D. Ferraiolo. On the formal definition of separation-of-duty policies and their composition. In *Proc. 19th IEEE Symposium on Security & Privacy*, pages 172–183, 1998.
14. S. Jajodia, M. Kudo, and V. S. Subrahmanian. Provisional authorization. In *Proc. E-commerce Security and Privacy*, pages 133–159. Kluwer Academic Publishers, 2001.
15. S. Jajodia, P. Samarati, M. L. Sapino, and V. Subrahmanian. Flexible support for multiple access control policies. *ACM Transactions on Database Systems*, 26(4):216–260, 2001.
16. G. Karjoth and M. Schunter. A privacy policy model for enterprises. In *Proc. 15th IEEE Computer Security Foundations Workshop (CSFW)*, pages 271–281, 2002.
17. G. Karjoth, M. Schunter, and M. Waidner. The platform for enterprise privacy practices – privacy-enabled management of customer data. In *Proc. Privacy Enhancing Technologies Conference*, volume 2482 of *LNCS*, pages 69–84. Springer, 2002.
18. S. Lakshminarayanan, R. Ramamoorthy, and P. C. K. Hung. Conflicts in inter-prise epal policies. In *W3C Workshop on the long term Future of P3P and Enterprise Privacy Langugate; Position Papers*. World Wide Web Consortium, 2003.
19. J. McLean. The algebra of security. In *Proc. 9th IEEE Symposium on Security & Privacy*, pages 2–7, 1988.
20. J. D. Moffett and M. S. Sloman. Policy hierarchies for distributed systems management. *IEEE JSAC Special Issue on Network Management*, 11(9):1404–31414, 1993.
21. Platform for Privacy Preferences (P3P). W3C Recommendation, Apr. 2002.
22. C. Ribeiro, A. Zuquete, P. Ferreira, and P. Guedes. SPL: An access control language for security policies with complex constraints. In *Proc. Network and Distributed System Security Symposium (NDSS)*, 2001.
23. R. T. Simon and M. E. Zurko. Separation of duty in role-based environments. In *Proc. 10th IEEE Computer Security Foundations Workshop (CSFW)*, pages 183–194, 1997.
24. D. Wijesekera and S. Jajodia. Policy algebras for access control – the propositional case. In *Proc. 8th ACM Conference on Computer and Communications Security*, pages 38–47, 2001.
25. D. Wijesekera and S. Jajodia. A propositional policy algebra for access control. *ACM Transactions on Information and System Security*, 6(2):286–325, 2003.
26. eXtensible Access Control Markup Language (XACML). OASIS Committee Specification 1.0, Dec. 2002. `www.oasis-open.org/committees/xacml`.