A Data-centric View on Expressing Privacy Policies

Sebastian Speiser
Karlsruhe Service Research Institute (KSRI)
Institute AIFB, Karlsruhe Institute of Technology (KIT), Germany
Email: speiser@kit.edu

***Abstract*—Services often depend on data about users to work at all (e.g. providing quotes for health insurance) or to improve their quality (e.g. product recommendation systems). Storing and giving access to data owned by third parties is in many cases even the core task of services, e.g., for social networks or cloud storage providers. Privacy is an important concern, as users still want to control usage and distribution of their data. Enabled by Internet technologies, services are often provided by dynamically created and frequently changing groups of cooperating providers. This leads to a situation, where often no single entity has a complete view of the process operating on a user's data. In consequence, it is difficult to check compliance of such a process with the user's privacy policy. As an alternative model, we propose a data-centric view on privacy policies, that are attached to data artefacts and are self-contained descriptions of the allowed actions to be performed. Such policies can be passed together with the artefacts to subproviders. A key challenge of such policies is to express restrictions on the policies of derived artefacts, which can also be subject to privacy constraints.**

I. Introduction

The Internet provides access to an ever growing number of services and applications, which rely on collecting, storing, aggregating, analysing, or sharing of user data. Currently the most prominent example for services used by private persons are social networks, that collect personal information, and share them with other users as a basic functionality. More advanced usages of the data include analysis to recommend new contacts, and sharing with third-party applications, e.g. a birthday calendar. Similarly businesses use software as a service (SaaS), e.g. a customer relationship management (CRM) service. Such services can fulfill complex and customer-specific business needs, and are often provided by dynamically created and frequently changing groups of cooperating providers. The common theme for both business and private users is that more services are used, which are provided by entities, that are dynamically selected and often not known to the users. In consequence, large amounts of data are shared with and amongst providers, without centralised control by the data owner.

Data owners, however, still want to limit the sharing and usage of their data, without having to know the processes in which the data is used. Privacy of data is governed in many countries by laws and norms, e.g., the German privacy law (Bundesdatenschutzgesetz, BDSG), or the American Health Insurance Portability and Accountability Act (HIPPA). Such laws apply to sensitive information about persons, but in our approach to protecting privacy, we take a broader view, so that data owners themselves can decide for which data they want to have restricted usages. The right to protect different kinds of data can be founded on laws (e.g., copyright law, privacy law) or social norms (e.g., not exposing a friend's email address to spammers).

We present a method for formalising fine-grained usage restrictions as privacy policies, which can be used to automatically check compliance of data processing. Existing approaches to privacy policies typically take a process-centric view, which is characterised by the following properties: (i) policies restrict the processes performed by a system, (ii) policies apply to a class of data artefacts (e.g., all health records in the system), and (iii) policies are stored in a central repository of the system.

The process-centric view does not adequately address the following requirements of our outlined scenarios:

- the compliance of an isolated data artefact usage must be verifiable, as there is no central instance knowing the complete process in which the usage is embedded;
- data owners must be able to specify individual privacy policies for their artefacts, e.g., some but not all social network users want to share their birthdate with a calendar application;
- policies must be attached to data artefacts, so that they can be transferred together with the artefacts between different providers.

As a solution, we propose a *data-centric view* on privacy policies, where each artefact (e.g., Alice's health record) has an attached policy specifying the allowed usages of the artefact (e.g., sharing the health record with her physician Bob). Certain data usages produce new data artefacts, which are derivations or copies of the used artefacts (e.g., sharing means that the recipient gains a copy of the shared artfact). In the simplest case, such a produced artefact inherits the policy of the used artefact (e.g., a copy of a confidential artefact is also confidential). However, generally the policy of the produced artefact can also allow broader usages (e.g., after anonymising an artefact) or allow narrower usages (e.g., Bob may use his copy of the health record for medical treatment purposes but is not allowed to share it further).

For restricting the allowed policies of produced artefacts there are two possibilities: (i) *name-based* policy restrictions, and (ii) *content-based* policy restrictions. Policies using name-based policy restrictions specify an exhaustive list of admissible policies for a produced artefact, which is problematic in scenarios with independent actors that employ policies with different names (i.e., not in the specified list) but a compatible set of allowed usages (i.e., should be in the list). To overcome such incompatibilities, we use a policy language supporting content-based policy restrictions. Instead of explicitly listing admissible policies for produced artefacts, the content-based approach describes the admissible policies in terms of the minimal or maximal allowed usages. Content-based restrictions increase compatibility by making policies admissible based on the usages they allow independent of their names. Checking whether a policy fulfills at most or at least the same usages as a target policy corresponds to policy containment, which is well-studied for many policy languages as a tool for policy management (e.g., [1], [2]). The approach of content-based policy restrictions goes one step further and uses policy containment as part of the policy definitions themselves, which requires meta-modelling (policies are both sets of allowed usages as well as part of usage descriptions) and self-referentiality (a policy can require that derived artefacts have a policy that allows at most the same uses as the original policy).

In Section II, we present several use cases that illustrate the need for expressive privacy policies. Our proposed solution is based on the following main contributions:

- We introduce the distinction between process-centric and data-centric privacy policies and demonstrate the need for content-based policy restrictions (Section III).
- We present a vocabulary and formalism for data-centric privacy policies (Section IV), based on a logic framework with content-based restrictions presented in [3].
- We show how usage restrictions commonly found in privacy policies can be formalised with our approach (Section VI).

We evaluate our approach in Section VII by applying our approach to express the policies of the use case and describing how policies of the P3P standard can be translated into our language. Finally, we discuss related work in Section VIII and conclude in Section IX.

II. Use Cases

In this section we introduce concrete use cases of actors that want to restrict the usage of their data. Each actor interacts with a different kind of service and has a specific privacy policy.

Alice lives in an apartment equipped with a smart meter, which tracks in detail her consumption of electrical energy. The vision of the Smart Grid builds on the availability of up to date and fine-granular data about energy consumption and production in order to make energy grids more efficient and reduce environmental impact. Besides its obvious uses for billing and statistical purposes, there also various scenarios how energy consumption data can be misused to the disadvantage of individuals. In order to protect her privacy while participating in the Smart Grid, Alice specifies the following usage restrictions for her consumption data when passed to her energy provider: the data can be stored for one year and must be deleted afterwards; the stored data can be used for billing purposes; the data can be anonymised. She further restricts that the anonymised data can be stored forever, used for statistical purposes and shared with third parties under the same conditions (allowed to be stored, used for statistical purposes and shared under the same conditions).

Bob regularly uses shopping sites on the Internet, but he is careful in selecting only providers he trusts. Bob allows the shopping providers to store his birthday and use it for marketing purposes, so that he can be better personalised product recommendations. However, he does not want that the providers share his birthday with third parties, as Bob might not trust them. Provider 1 specifies that the birthday is stored for 6 months to be used for marketing purposes and then deleted. The specified usage is more restricted than what Bob allows, so he will pass his birthday to Provider 1. In contrast, Provider 2 specifies that he stores the birthday and uses it for marketing purposes, but also wants to share it under the same usage conditions. Bob's policy does not approve of further sharing and thus Provider 2 will not get Bob's birthday.

Carol is registered in a social network, which she uses to show photos to her friends. She allows the social network to store her photos and share them with Carol's friends under a policy, that allows the friends to use the photo for non-commercial purposes.

Dave works in the marketing department of a company and regularly writes blog posts about new product features. Dave's company has outsourced the hosting of the blog to a specialised provider. The provider offers to registered users the ability to comment on blog posts and share them via email or social networks with their own contacts. The company wants to track the spread of news in order to track the interest in specific features by its (potential) customers. Therefore, Dave publishes his blog posts with a policy that allows the hosting provider and its registered users to share the posts with other registered users under the condition that a notification is sent to Dave.

III. Data-centric Usage Policies

The use cases of Section II show that data usage restrictions cannot be boiled down to simply allowing or forbidding access to data; access and usage is necessary, respectively desired in all cases, but should be confined to well-defined borders. Usage policies can define these borders in terms of a formal model.

Usages policies, in contrast to traditional access control, still apply after initial access was granted. Specifically, usage policies can (i) impose obligations and (ii) restrict the usage

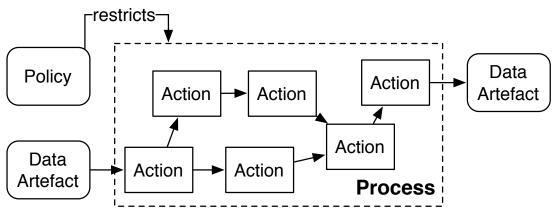

Figure 1. Process-centric Policies

of artefacts generated in dependency of the original protected artefacts.

Obligations are actions that a user is required to fulfill when executing an action allowed by the policy, e.g. a privacy policy can allow a provider to store a data artefact with an obligation to delete the artefact after a certain time span.

Restrictions on the allowed usages of rightfully obtained or derived artefacts are in the simplest case just inherited from the original artefact. As an example, consider that a provider shares an email address that cannot be used for marketing purposes with a subprovider. The artefact generated by the sharing process, i.e. the subprovider's local copy of the email address, can again not be used for marketing purposes. However, in the general case the restrictions of a derived artefact can differ arbitrarily from the restrictions of the original artefact. Depending on the actions performed on an artefact, the restrictions on the resulting artefact, can be both laxer or stricter. Consider again the example of sharing a data artefact, but in this case the subprovider is only allowed to use the artefact for a specified purpose but not to share it with a third party, i.e., the restrictions on the resulting artefact are stricter. Another example is a zip code, which can be freely used and shared by the provider for statistical purposes. However, an artefact derived from both the zip code and the birth date of an individual, cannot be shared anymore, as it possibly allows the identification of the individual. More relaxed restrictions can for example be applicable after an anonymisation action. Additional rights can be admitted on the anonymised artefact, while still some restrictions can apply, e.g. it can only be used for research purposes.

For expressing such usage restrictions, we identify two alternative approaches:

1) Process-centric policies: specify restrictions on the overall process in which a data artefact and its derivations are used. See Figure 1.
2) Data-centric policies: specify which actions are allowed to be performed on an artefact and restrict the possible policies that can be assigned to artefacts generated by these actions, i.e. derived from the protected artefact. See Figure 2.

Process-centric policies model a set of allowed processes that are allowed to be performed on a protected artefact. Sets

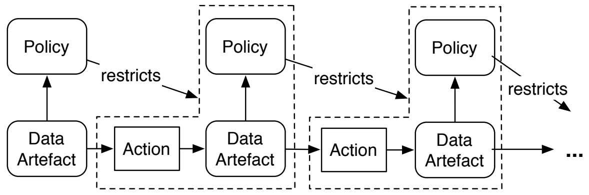

Figure 2. Data-centric Policies

of compliant processes can be described by temporal logic formulae [4], [5].

The drawback of process-centric policies is that always the complete process which works with some data artefact or its derivations must be known in order to judge its compliance. Having such knowledge is not always feasible, let alone practical, for the following reasons:

- data processing can involve sharing the data with external entities, i.e. the data leaves the process and is further used by an external provider, whose own process is just a black box;
- derivations of a data artefact can be used in new processes, which were not known at the time of the data derivation. In order to check compliance of such a new process, the whole processing history of the derived data artefact must be known, which is not always desirable, e.g., if the derivation was an anonymisation of an identifiable artefact.

In general, it is desirable to assign usage policies to all, also intermediate, artefacts consumed and produced in a process. Policies attached to artefacts enable compliance checks when an artefact is passed to a third party or is utilised in a new process.

Data-centric policies attach a usage policy to every protected artefact [6]. Compliance is checked for every action using an artefact in isolation. Aspects to check are: (i) is the action allowed, (ii) will obligations be fulfilled, and (iii) is an appropriate policy assigned to any artefact produced by the action? The isolated view on each action is enabled by two assumptions:

1) Each used artefact has an attached policy, that was either set by the artefact's owner or was assigned in compliance with the policies of the artefacts used to derive the artefact.
2) Produced artefacts will be used in compliance with the assigned policy.

Whereas actions that are artefact usages are regarded in isolation, the conditions on compliant actions can involve the existence of other actions. For example, a policy can allow a storage action under the condition that a deletion action will be triggered within a specified time frame. In this way, obligations can be modelled. Note, however that only concrete instances of actions are restricted, and thus conditions requiring the triggering of another simple action are conceptually simpler than logics for restricting processes, where e.g. deletion must be ensured in all traces that are possible according to the process description.

As discussed above, the policies of derived or produced artefacts can be restricted by the policies of used artefacts.

Policy restrictions can be easily modeled by specifying an exhaustive list of all policies than can be assigned to derived artefacts. The admissible policies would be identified by their name, not by the usages they allow or forbid. Such *name-based* policy restrictions have considerable impact on the interoperability between different providers. Consider e.g. a policy that allows a provider to store an artefact with policy p_1, which allows arbitrary usages of the artefact. The provider that wants to store the artefact, however specifies that he wants to assign policy p_2, which only allows usages for statistical purposes. The storage action is classified as non-compliant, because the provider wants to use policy p_2 instead of policy p_1. However, the intention of the owner of the artefact is most certainly that the storage is also allowed under a policy that allows a restricted set of usages. Even for policies with exactly the same set of allowed usages, a non-compliant classification can be made because different policy names were used. In a scenario with a multitude of heterogeneous providers, it is not realistic to assume canonical names for policies with the same set of compliant usages.

To overcome this interoperability problems, we use *content-based* policy restrictions that are based on the usages allowed by policies. Namely, we enable the use of policy containment conditions in restrictions of allowed usages. A policy p_a is contained in a policy p_b, if every usage compliant to p_a is also compliant to p_b.

With the containment conditions, the previous example can be modified in the following way: the policy allows a provider to store an artefact with a policy that is contained in policy p_1 (which allows arbitrary usages). Now, the storage of the artefact with policy p_2 is compliant, as every usage for statistical purposes, i.e., every usage compliant to p_2, qualifies as an arbitrary usage, i.e., is also compliant to p_1 and thus p_2 is contained in p_1.

Besides setting an upper bound on the actions allowed by a target policy, the containment condition can also be used to set a lower bound. For example, an individual can decide to share his energy consumption patterns with a statistics institute under the condition that derived artefacts are made available for public use and not only for a selected number of paying clients.

IV. Policy Language

In the previous section, we argued for policies that are attached to the data artefacts, which they protect by specifying allowed usages. In this section, we first present a top-level vocabulary for policies, actions, and data artefacts, which we subsequently specialise into an action vocabulary that is suitable for expressing privacy policies. The vocabulary contains a special containedIn predicate that expresses the containment relation between two policies. In the second part of this section, we introduce a formalism that gives the desired semantics to the containedIn predicate.

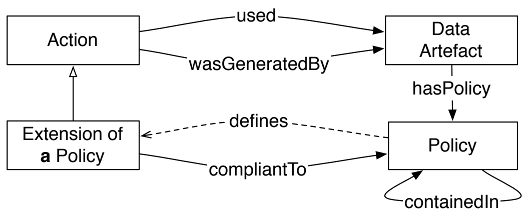

Figure 3. Top-level model of policies, actions and data artefacts.

A. Policy Vocabulary

The top-level model is visualised in Figure 3, where rectangles represent concepts, solid lines with filled arrows are properties between instances of the connected concepts, solid lines with unfilled arrows denote a subconcept relationship, and the dashed line is a special relation expressing that each instance of the concept policy defines its own class of compliant actions. An action uses one or more data artefacts. Artefacts in turn are generated by actions. The policy of a data artefact is both treated as an instance of the class policy and a subclass of action. A policy that is contained in another policy, has an extension, i.e. the class of compliant actions, that is a subclass of the extension of the other policy. A formalism capable of this form of meta modeling is presented in Section IV-B. A subclass of action, defining the actions compliant to a policy, can be defined in terms of all properties of the action, including the policies of generated artefacts, and their containedIn-relations to other policies. An action is compliant, if it is compliant with the policies of all used data artefacts.

For modeling privacy policies, we introduce in the following a model of relevant actions and their basic properties, based on the German privacy law and the use cases introduced in Section II. We extend actions by the actor property that relates each action to an Agent, who is performing the action. The performedAt property assigns a point in time to an action.

We understand a data artefact as an immutable physical manifestation of an abstract information object with a fixed policy. Thus, an action cannot modify a data artefact, but only create a new derived artefact. The same holds for modifying the policy of an artefact, which in our model, means creating a new artefact realising the same information object but with another policy. In this sense, sharing of an artefact creates a new data artefact, as either it is transmitted to another agent (i.e. a new physical manifestation is created), its policy is changed (rights are granted to another agent), or both. Pseudonymisation and anonymisation obviously modify data, and thus also result in new artefacts. Storing results in a new data artefact with the same information as its input artefact, but with a possibly modified policy. A stored artefact is kept until its Deletion. Blocking does not remove the artefact, but assigns a policy that does not allow any further actions to be performed on the artefact. The triggers property between actions, can be used to specify, e.g., that a storing action must require a deletion or blocking action of the stored artefact within a specified time frame.

Usage of a data artefact is a general class of actions that are not data processing (i.e., sharing, blocking, modification, deletion, or storing). Usage actions have an assigned purpose. In contrast, the Germany privacy law assigns also a purpose to processing actions, but in our model processing is not done for its own purpose, but only for obtaining, preserving or transmitting a data artefact for further use. Representing that an artefact is only allowed to be stored for a certain purpose, can be done by allowing a storing action, and restricting the policy of the stored artefact to allow only use for the specific purpose. Use of an artefact can result in new artefacts, e.g., the result of an analysis algorithm, which can have again restricted policies.

B. Policy Formalism

This section instantiates a concrete formalism for data-centric privacy policies, which is based on the more general framework for formalisms supporting the containedIn predicate, which was introduced in [3]. A policy is a specification of the actions allowed to be performed on a protected artefact. The employed formalism corresponds to Datalog with safe and connected rules and a special interpretation for the containedIn predicate that we explain in this section. Datalog is a popular choice as formalism for policy languages (e.g., [7], [8], [9]) We model policies as unions of conjunctive queries (UCQs) with one head variable; the individuals that are answers to the query of a policy form the set of compliant actions.

Example 1: A policy p that allows usage, if the purpose is non-commercial, is formalised as follows:

$$p(X) \leftarrow \mathsf{Usage}(X) \wedge \mathsf{hasPurpose}(X, U) \wedge \mathsf{NonCommercial}(U).$$

The predicate containedIn has the following definition:

$$\forall P_1, P_2.\mathsf{containedIn}(P_1, P_2) \leftrightarrow (\forall X.P_1(X) \rightarrow P_2(X)).$$

Meaning that containedIn holds between policies P_1 and P_2, if every usage compliant to P_1 is also compliant to P_2. Policy containment is thus equivalent to containment of UCQs, which can be reduced to query answering, which is a standard task for Datalog engines.

Example 2: Consider policy p from Example 1 and the policy $q(X) \leftarrow \mathsf{Usage}(X)$. As every non-commercial usage compliant to p is also compliant to q, the following containment holds: $\mathsf{containedIn}(p, q)$.

The interpretation of the containedIn predicate gets more complicated, if it is used in a self-referential manner in policy conditions. Consider the following two policies:

$$\begin{aligned} r(X) \leftarrow& \mathsf{Derivation}(X) \wedge \mathsf{wasGenBy}(A, X) \wedge \\ & \mathsf{hasPolicy}(A, P) \wedge \mathsf{containedIn}(P, r). \\ s(X) \leftarrow& \mathsf{Derivation}(X) \wedge \mathsf{wasGenBy}(A, X) \wedge \\ & \mathsf{hasPolicy}(A, P) \wedge \mathsf{containedIn}(P, s). \end{aligned}$$

The decision, whether r is contained in s depends on the condition $\forall P.\mathsf{containedIn}(P, r) \rightarrow \mathsf{containedIn}(P, s)$, which holds if every policy P contained in r is also contained in s, i.e., $\mathsf{containedIn}(r, s)$. In other words: r is contained in s, if r is contained in s. Thus, we can freely choose whether the containment holds or not. In this paper, we argued that content-based policy restrictions are useful to increase interoperability by making policies compatible that have the same intention but different names. The same intention of r and s is obvious from their definitions, which coincide except for the policy names. Therefore, the semantics of the containedIn predicate is that its extension is maximised, i.e., we assume that containment holds if there is no contradicting evidence (e.g., a policy allowing all usages would not be contained in a policy allowing only usages for a specific purpose). A formal definition of the outlined semantics of the containedIn predicate for general fragments of first-order logic and notes on how to implement it are given in [3].

V. System Overview

In this section, we describe how a policy-aware system can be constructed. Policy-awareness means that the system only executes or allows to execute actions that are compliant with applicable policies. For this the system needs descriptions of actions in the model as presented in Section IV-A. The translation from actual actions to their description is application-specific; we assume in this paper that such descriptions are available. The most basic operation of a policy-aware system, is to classify an action description into compliant or non-compliant, based on a complete description of the actions, the used and generated artefacts and their policies. The basic classification is described in Section V-A. Some actions are classified as non-compliant, but can be made automatically made compliant by an obligation handler, e.g. by scheduling a missing deletion operation in the future. Such obligations are discussed in Section V-B. Related to obligations is the determination of target policies, i.e. the policies that have to be assigned to the results of an action that uses policy-protected artefacts as input. The determination is important, e.g., if an artefact should be shared with a third party, and is described in Section V-C. Another operation is to check whether data requests are allowed and if yes under which conditions. For this a requester tells a data owner which data he needs and what his intended usages are. This process is described in Section

V-D, which introduces data categories and elements, which are useful to request data of a certain kind, without knowing the names of the concrete data artefacts.

Please note that the work presented in this paper focuses on the expression of policies and the automated decision making based on the policies. Technical enforcement of the policies is considered as a complementary task. For example, our system decides that an artefact must be deleted within a year or that access to an artefact is not allowed. The scheduling and execution of the deletion, respectively the prevention of access to an artefact is considered to be taken care of by the underlying policy-aware application or process execution engine.

A. Compliance Checking

Basic classification, checks whether an action description is compliant to the policies of the artefacts, on which, the action is performed. Such an action description includes policies of artefacts which are the results of the action. In the following situations, such a description can be available:

- Compliance checking is done ex-post on action descriptions, that are created from process logs, where policies of input and output artefacts are recorded.
- The resulting artefact is requested by another entity with a specific policy that is required by the requester for the artefact.
- The data processor knows which are the minimum subsequent actions that he needs to perform on the artefact, and thus they can be modeled as the policy of the resulting artefact. This is possible, if either the process in which the data is used is known, or the agent only processes data under some minimum rights, e.g., data is only collected, if it can subsequently be used for analysis.

For checking the compliance of an action c, the system retrieves all policies of the the artefacts, which are used by c by the following query: $\mathsf{pols}_c(P) \leftarrow \mathsf{used}(C, A) \wedge \mathsf{hasPolicy}(A, P)$. Then the system checks whether c is compliant to all policies, i.e., whether $\forall P.\mathsf{pols}_c(P) \rightarrow P(c)$.

If an action is not compliant, then there are several possibilities, depending on the situation:

- Change the application or process in which the action is performed.
- When process logs are checked, give an alert about the violation. The alert can then be the input for an auditor for further investigation on the violation.

Changing of the application can either be done manually or automatically. In both cases, it is useful to know why the action is non-compliant. From a logical viewpoint, an action c can be non-compliant to a policy p in two different ways:

1) the action is inferred to not be an answer to p: this means that $p(c)$ is a logical contradiction. Methods for computing logical justifications can find the minimal sets of facts in the action description that lead to the contradiction. Compliance can be reached, by modifying the application so that all facts from one of the sets is removed from the action description.
2) the action is not inferred to be an answer to p: this means that the facts stated about c are not sufficient to conclude c's compliance. Abductive reasoning is a method to find an explanation why an inference holds[1]. When applied to $p(c)$, an explanation would comprise the facts that would make c compliant; if the currently holding facts are removed from the explanation, we gain a description what is missing in the current state of the application.

The results of both methods are logical formulae that express what leads to the non-compliance of an action. For human users these logical formulae can be hard to understand and thus to be put into the right changes to the application. Various approaches to generate human-understandable, natural language explanation from logical justifications exist, and can be applied to our approach in future work.

Two special cases of actions that have non sufficient descriptions for compliance and can be handled automatically are (i) obligations, and (ii) determination of target policies. Both are discussed in the following sections.

B. Obligations

As discussed above, we understand policies as descriptions of allowed actions, which can viewed as goals for applications that want to process a protected artefact. Goal-based policies are on a higher conceptual level than action-based policies, which state for every situation, which actions have to be performed [13]. An action-based policy would be formulated like the following: if you store the artefact, then you must delete it within a year. In contrast our approach would model the following: an storing action is compliant if the data is deleted within a year. For this simple example, the differences are subtle but the advantage of goal-based policies can be shown nonetheless. Consider, that the disk on which the artefact was stored was destroyed by some hardware failure and no backup exists. After one year, the system with action-based policies, would try to execute the deletion of the artefact, which fails as the artefact can no longer be found. The goal-based system in contrast with the appropriate background knowledge inferred when being told about the loss of the disk that this corresponds to a situation where the artefact is deleted and thus the system is compliant with the privacy policy.

In the previous section, we discussed explanations gained from abductive reasoning, which describe what is missing for an action to be compliant. However, one cannot simply take such an explanation as an obligation, as is illustrated by

[1]The term of abductive reasoning goes back to Peirce [10] and was applied to formal logics, e.g. by Poole [11]. Becker and Nanz use abduction for explanations of Datalog-based policies [12].

the following example: an artefact is protected with a policy that allows managers to use the data, when a notification is sent to the owner. An action description of an employee trying to use the data, would yield an explanation that it is needed (i) that the employee is a manager and (ii) a notification is sent to the owner. Part (i) can clearly not be fulfilled automatically by the system, i.e., the system cannot simply promote the employee to be a manager. If however the manager condition would be fulfilled, then part (ii) could be fulfilled by scheduling a notification with the obligation handling system.

Which parts of an explanation can be seen as an obligation must be decided by a system-specific classification. A good starting point for such a classification is to determine the list of actions that the explanation states must be triggered by the actions that the system wants to perform. The list can be traversed and actions that are a subclass of an obligation class (e.g. comprising notifications, deletions and blockings) can be scheduled automatically.

C. Target Policy Determination

We discussed, that obligations are typically actions that should be triggered by the action to be performed. Another type of missing facts that can be handled automatically is the determination of a policy for a generated artefact. If an action generates a new artefact and it is not assigned a policy, the abductive explanation for gaining compliance will include a list of containedIn statements about the policy of the new artefact.

For a target policy p, the statements are either of the forms $\mathsf{containedIn}(p, \mathsf{containing})$ or $\mathsf{containedIn}(\mathsf{contained}, p)$, for $\mathsf{contained}, \mathsf{containing} \in N_P$. To obtain a target policy p, the conjunction of all containing policies is formed and redundant restrictions removed (e.g. requiring deletion after one year is redundant if deletion after six months is already required). If the conjunction p contains all contained policies, then it is a valid target policy. If there is a contained policy that is not contained in p, there is a contradiction and there exists no valid target policy. For a detailed discussion and algorithms for the target policy determination, we refer the reader to [14]

D. Requesting Data

A data processor that wants to collect data from a data owner, cannot always simply model his intended collection action and check its compliance with the owner's policies, for the following reasons:

- The data owner does not want to publish his full privacy policy, as this could reveal confidential information, about which data exists, and about its access rights.
- The data owner gives access to data, only when collection is allowed, thus the data requestor cannot refer to data artefacts in his collection action, as he does not know their name.

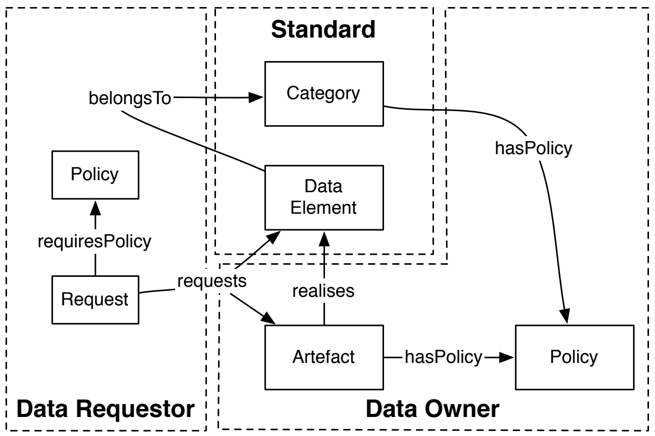

Figure 4. Model of requests, data elements and categories. Dashed boxes denote by their bold labeling where the corresponding knowledge is specified.

As a solution, we introduce a request model, which is visualised in Figure 4. A data request either refers to a data element or directly to a data artefact, and a policy under which the requestor intends to use the data. The term data element is adopted from P3P and denotes a subclass of data artefact, to which a data owner can assign his artefacts. E.g., a user can state that the artefact containing his street address realises the data element user address. If a data requestor asks to collect the user address, the data owner's system knows that granting the request, involves the collection of the street address artefact, and check his policies accordingly. Furthermore, the requestor can state to which categories a data element belongs. This is stated by the requestor, as the same data elements can be used in different ways, by different requestors, e.g., an IP address can be classified in the categories personal identifier or geographical information. The data owner can assign policies to categories, e.g., his IP address may be accessible under different conditions, depending on the category under which it is requested.

Checking compliance of a data request is done with the following procedure:

- Requestor sends request for data either by referring directly to artefacts or indirectly to data elements. Attached to the request is a policy specifying the intended usage of the data and optionally a specification in which category he classifies the requested data elements.
- The data owner selects the applicable artefacts, i.e., the directly addressed artefacts and the ones belonging to the requested data elements.
- Data artefacts with no attached policy but belonging to a data element with an assigned category inherit the policy of the category.
- The data owner checks, whether the request policy is containedIn the attached policies of all selected data artefacts.

- If all containments hold, the data artefacts with their corresponding policies can be shared with the requestor.
- If one containment does not hold, the request is denied.

Depending on the concrete application, the request process can be varied in the following ways:

- Instead sharing all or none requested artefacts, it is also possible to share the subset of artefacts for which the request policy is compliant. This is especially useful if a data element is requested, for which a number of artefacts exist, that are in a hierarchical relation, e.g. differing in detail level. An example would be a request for the birthdate of a person, which can be represented by two different artefacts with different policies, one only containing the year of birth, whereas the other contains the exact day, month and year of birth. In this, case depending on the request policy, either no information, the year of birth, or the day of birth can be given out in return to a request for the birthdate.
- Instead of attaching the user's policy for an artefact or a category to copies of the artefact that are shared with a requestor, it is also possible to attach the policy of the requestor. For a successful request, the requestor's policy is always equal or more strict than the policy of the owner, so it might be desirable to grant only the rights to the requestor, that he needs. This is especially useful, if the data owner's policy contains sensitive information, e.g., about which entities can have access to an artefact.

VI. Policy Patterns

In this section, we discuss several patterns that express common policy restrictions.

Attribute-based Usage Restrictions

The policy vocabulary can be extended with application-specific properties, that can be used for modelling compliant actions. For example a social network can model the hasFriend property between agents participating in the network. Restricting usage to agents that have the attribute "friend of Carol" can be modelled as follows:

$$p(X) \leftarrow \mathsf{Usage}(X) \wedge \mathsf{wasControlledBy}(X, F) \wedge \mathsf{hasFriend}(\mathsf{carol}, F).$$

Each data artefact has its own policy attached, however it can make sense to specify policies that can be attached to a whole class of artefacts and specify compliant usages in terms of attributes of the concrete protected artefact. Consider for example the following policy, which specifies that only a person depicted on a picture can share it with others:

$$p(X) \leftarrow \mathsf{Sharing}(X) \wedge \mathsf{wasControlledBy}(X, Y) \wedge \mathsf{used}(X, I) \wedge \mathsf{depicts}(I, Y).$$

Data Sharing / Rights Delegation

One key feature of our proposed approach is the ability to express restrictions on the policies of shared data. Rights delegation can be allowed by enabling the rights holder to share the data with further parties under the same or more restricted conditions. The following example policy p_1 allows arbitrary usage and the free delegation of this right:

$$\begin{aligned} \mathsf{p}_1(X) \leftarrow & \mathsf{Usage}(X). \\ \mathsf{p}_1(X) \leftarrow & \mathsf{Sharing}(X) \wedge \mathsf{wasGenBy}(A, X) \wedge \\ & \quad \mathsf{hasPolicy}(A, P) \wedge \mathsf{containedIn}(P, \mathsf{p}_1). \end{aligned}$$

Of course, the delegation can be further constrained e.g. on the attributes of the actor or the artefact as shown before. One common aspect of rights delegation is limiting the depth upto which rights receivers can further delegate the rights. This can be implemented by sharing an artefact with policies allowing a decreasing number of further sharings. A general set of policies, which allows n delegations is shown in the following, where p_1 is assigned to the original artefact (for $i \in [2, n-1]$):

$$\begin{aligned} \mathsf{p}_1(X) \leftarrow & \mathsf{Usage}(X) \vee (\mathsf{Sharing}(X) \wedge \mathsf{wasGenBy}(A, X) \wedge \\ & \quad \mathsf{hasPolicy}(A, P) \wedge \mathsf{containedIn}(P, \mathsf{p}_2)). \\ \mathsf{p}_\mathsf{i}(X) \leftarrow & \mathsf{Usage}(X) \vee (\mathsf{Sharing}(X) \wedge \mathsf{wasGenBy}(A, X) \wedge \\ & \quad \mathsf{hasPolicy}(A, P) \wedge \mathsf{containedIn}(P, \mathsf{p}_{\mathsf{i}+1})). \\ \mathsf{p}_\mathsf{n}(X) \leftarrow & \mathsf{Usage}(X). \end{aligned}$$

In future work, we aim to develop a templating language for specifying such a set of policies in a compact way.

Hierarchies and other Domain Knowledge

Hierarchies are widespread when modelling usage policies:

- User groups. E.g. managers are also employees; rights granted to acquaintances are also granted to friends.
- Artefacts. E.g. usage of a birthday implies usage of personal information; allowing the sharing of location data includes the sharing of latitude and longitude.
- Purposes. E.g. marketing is a commercial purpose.

Such hierarchies can be modelled as background knowledge in the form of Datalog rules, e.g. the following rules express that every friend of a person is also an acquaintance and that every manager is also an employee:

$$\begin{aligned} \mathsf{hasAcquaintance}(P, F) &\leftarrow \mathsf{hasFriend}(P, F). \\ \mathsf{Employee}(M) &\leftarrow \mathsf{Manager}(M). \end{aligned}$$

Furthermore, background theories can express domain knowledge, e.g. that a selling action has per definition a commercial purpose:

$$\mathsf{Commercial}(\mathsf{c}). \qquad \mathsf{hasPurpose}(X, \mathsf{c}) \leftarrow \mathsf{Selling}(X).$$

Another example is that a disk failure can be regarded as a deletion action of the artefacts stored on the disk:

$$\begin{aligned} & \mathsf{Deletion}(X) \wedge \mathsf{used}(X, A) \\ \leftarrow & \mathsf{DiskFailure}(X) \wedge \mathsf{affected}(X, D) \wedge \mathsf{isStoredOn}(A, D). \end{aligned}$$

Anonymisation / Pseudonymisation

The extension of rights after an artefact is anonymised or pseudonymised is modelled in a similar way as rights delegation. A corresponding policy condition allows the transformation action and restricts the generated artefact's policy accordingly. Consider for example a policy that allows arbitrary usage (but no sharing) after anonymisation:

$$\begin{aligned} \mathsf{p}_\mathsf{o}(X) \leftarrow & \mathsf{Anonymisation}(X) \wedge \mathsf{wasGenBy}(A, X) \wedge \\ & \quad \mathsf{hasPolicy}(A, P) \wedge \mathsf{containedIn}(P, \mathsf{p}_\mathsf{a}). \\ \mathsf{p}_\mathsf{a}(X) \leftarrow & \mathsf{Usage}(x). \end{aligned}$$

Opt-in and Opt-out

Opt-in means that a user has to explicitly agree to a data usage, whereas opt-out means that data usage is done by default if the user does explicitly disagree with it. A basic assumption of our policy model is that usages are only allowed if they are explicitly described as compliant to a policy. This corresponds to opt-in. Data usages for which a user can decide, whether he wants to allow it to a service provider, often affect other attributes of the service such as costs or performance. In this case, a provider can offer several configurations of his service for which he specifies data requests with different desired policies. Depending on the requested policies and offered configuration, a user can decide whether he wants to provide specific data artefacts.

Obligations

Obligations can be specified using the triggers predicate. As example take a usage $\mathsf{u_1}$ of an artefact with the policy p_1:

$$p_1(X) \leftarrow \mathsf{Usage}(X) \wedge \mathsf{triggers}(X, N) \wedge \mathsf{Notification}(N).$$

The description of $\mathsf{u_1}$ is $\mathsf{Usage}(\mathsf{u_1})$ and the background knowledge that deletions and notifications should be handled as obligations is formalised by the following rule:

$$\forall x.\mathsf{Obligation}(x) \leftarrow \mathsf{Notification}(x) \vee \mathsf{Deletion}(x).$$

A possible explanation how $\mathsf{u_1}$ can be made compliant to p_1 are the following facts: $\mathsf{triggers}(\mathsf{u_1}, \mathsf{n_1})$, $\mathsf{Notification}(\mathsf{n_1})$. An obligation handler can query the explanation for actions that are to be triggered by $\mathsf{u_1}$ and are classified as obligations: $q(X) \leftarrow \mathsf{triggers}(\mathsf{u_1}, X) \wedge \mathsf{Obligation}(X)$. The result is the notification $\mathsf{n_1}$, which can then be scheduled by the obligation handler.

Time Spans

A common condition of obligations is that they have to be fulfilled within a limited time span. Without a time restriction requiring a notification or deletion is not very valuable, because the obliged agent can postpone the fulfillment forever. The following policy requires that a stored artefact is deleted before the end of the year 2012:

$$\begin{aligned} p(X) \leftarrow &\mathsf{Storing}(X) \wedge \mathsf{wasGenBy}(A, X) \wedge \mathsf{triggers}(X, D) \wedge \\ &\mathsf{Deletion}(D) \wedge \mathsf{performedAt}(D, T) \wedge T \leq \text{"2012-12-31"}. \end{aligned}$$

Absolute time restrictions are used, as the policies refer to concrete artefacts that are passed to concrete providers. To see the advantage of absolute times, consider a policy with a relative time restriction that would allow storage, if it triggers a deletion one year after the storage, and allows the same terms for the stored artefact. The deletion obligation could be circumvented by always storing a new copy of the artefact, which again can be kept for one further year.

However, in case of a data owner who gives away his artefacts with the requirement to delete them after one year, it is inconvenient to update the absolute times in his policy every second. Instead, he can specify a template policy using time arithmetic expressions including the now() function. Whenever the compliance of a data request is checked or an artefact given away, the arithmetic expressions are replaced by the absolute time values to which the expressions evaluate. An example for such a template policy allowing one year of storage is given as follows:

$$\begin{aligned} p(X) \leftarrow &\mathsf{Storing}(X) \wedge \mathsf{wasGenBy}(A, X) \wedge \mathsf{triggers}(X, D) \wedge \\ &\mathsf{Deletion}(D) \wedge \mathsf{performedAt}(D, T) \wedge T \leq \mathsf{now}() + 1y. \end{aligned}$$

VII. Evaluation

In this section, we evaluate the suitability of our approach to expressing privacy policies. We first demonstrate how the use cases presented in this paper can be realised and then show how privacy policies expressed in the W3C standard P3P can be translated to our language.

A. Use Case Realisation

The policies for the use cases are specified in terms of the patterns presented in Section VI.

Alice's policy $\mathsf{p_a}$ for protecting her Smart Energy Grid data is is realised as a disjunction allowing either a storage action, an anonymisation or direct usage. The storage action requires deletion (obligation pattern) within one year (time span pattern), and the assignment of a policy p to the stored artefact that allows at most the same terms, i.e. a condition $\mathsf{containedIn}(p, \mathsf{p_a})$. A usage X is restricted to billing purposes, i.e., $\mathsf{hasPurpose}(X, U) \wedge \mathsf{BillingPurpose}(U)$. Artefacts produced by anonymisation must have a policy contained in the $\mathsf{p_{ano}}$ policy (anonymisation pattern), which allows storage, statistical usage and sharing under the same terms (rights delegation pattern).

$$\begin{aligned} \mathsf{p_a}(X) \leftarrow &\mathsf{Storing}(X) \wedge \mathsf{wasGenBy}(A, X) \wedge \mathsf{triggers}(X, D) \wedge \\ &\mathsf{Deletion}(D) \wedge \mathsf{used}(D, A) \wedge \mathsf{performedAt}(D, T) \wedge \\ &T \leq \mathsf{now}() + 1y \wedge \mathsf{hasPolicy}(A, P) \wedge \mathsf{containedIn}(P, \mathsf{p_a}). \\ \mathsf{p_a}(X) \leftarrow &\mathsf{Usage}(X) \wedge \mathsf{hasPurpose}(X, U) \wedge \mathsf{BillingPurpose}(U). \\ \mathsf{p_a}(X) \leftarrow &\mathsf{Anonymisation}(X) \wedge \mathsf{wasGenBy}(A, X) \wedge \\ &\mathsf{hasPolicy}(A, P) \wedge \mathsf{containedIn}(P, \mathsf{p_{ano}}). \end{aligned}$$

$$\begin{aligned} \mathsf{p_{ano}}(X) \leftarrow &\mathsf{Storing}(X) \wedge \mathsf{wasGenBy}(A, X) \wedge \\ &\mathsf{hasPolicy}(A, P) \wedge \mathsf{containedIn}(P, \mathsf{p_{ano}}). \\ \mathsf{p_{ano}}(X) \leftarrow &\mathsf{Usage}(X) \wedge \mathsf{hasPurpose}(X, U) \wedge \\ &\mathsf{StatisticalPurpose}(U). \\ \mathsf{p_{ano}}(X) \leftarrow &\mathsf{Sharing}(X) \wedge \mathsf{wasGenBy}(A, X) \wedge \\ &\mathsf{hasPolicy}(A, P) \wedge \mathsf{containedIn}(P, \mathsf{p_{ano}}). \end{aligned}$$

Bob's online shop privacy policy $\mathsf{p_b}$ allows usage for marketing purposes and storing the artefact under the same terms, i.e. with a policy P such that $\mathsf{containedIn}(P, \mathsf{p_b})$. Provider 1's policy $\mathsf{p_{p1}}$ additionally uses the obligation and time span patterns to specify that data is deleted after six months, whereas Provider 2 requests data with a policy, that also allows sharing with third parties. Provider 1's additional policy terms are a reduction of the allowed usages, meaning that the policy is contained in Bob's policy, thus the data will be released to Provider 1, who will use it in a compliant way. On the other hand, Provider 2's additional terms are an extension of the allowed usages (sharing is not in Bob's policy) and thus containment does not hold, resulting in a denial of Provider 2's data request.

$$\begin{aligned} \mathsf{p_b}(X) \leftarrow &\mathsf{Storing}(X) \wedge \mathsf{wasGenBy}(A, X) \wedge \\ &\mathsf{hasPolicy}(A, P) \wedge \mathsf{containedIn}(P, \mathsf{p_b}). \\ \mathsf{p_b}(X) \leftarrow &\mathsf{Usage}(X) \wedge \mathsf{hasPurpose}(X, U) \wedge \\ &\mathsf{MarketingPurpose}(U). \end{aligned}$$

Policy $\mathsf{p_{p1}}$ of Provider 1:

$$\begin{aligned}\mathsf{p_{p1}}(X) \leftarrow & \mathsf{Storing}(X) \wedge \mathsf{wasGenBy}(A, X) \wedge \\ & \mathsf{triggers}(X, D) \wedge \mathsf{Deletion}(D) \wedge \mathsf{used}(D, A) \wedge \\ & \mathsf{performedAt}(D, T) \wedge T \leq \mathsf{now}() + 6M \wedge \\ & \mathsf{hasPolicy}(A, P) \wedge \mathsf{containedIn}(P, \mathsf{p_{p1}}). \\ \mathsf{p_{p1}}(X) \leftarrow & \mathsf{Usage}(X) \wedge \mathsf{hasPurpose}(X, U) \wedge \\ & \mathsf{MarketingPurpose}(U).\end{aligned}$$

Policy $\mathsf{p_{p2}}$ of Provider 2:

$$\begin{aligned}\mathsf{p_{p2}}(X) \leftarrow & \mathsf{Storing}(X) \wedge \mathsf{wasGenBy}(A, X) \wedge \\ & \mathsf{hasPolicy}(A, P) \wedge \mathsf{containedIn}(P, \mathsf{p_{p2}}). \\ \mathsf{p_{p2}}(X) \leftarrow & \mathsf{Usage}(X) \wedge \mathsf{hasPurpose}(X, U) \wedge \\ & \mathsf{MarketingPurpose}(U). \\ \mathsf{p_{p2}}(X) \leftarrow & \mathsf{Sharing}(X) \wedge \mathsf{wasGenBy}(A, X) \wedge \\ & \mathsf{hasPolicy}(A, P) \wedge \mathsf{containedIn}(P, \mathsf{p_{p2}})).\end{aligned}$$

Carol's policy $\mathsf{p_c}$ for sharing photos in social networks allows storing of the photos and sharing with friends (attribute-based restriction pattern). Sharing is allowed under a policy $\mathsf{p_{friends}}$ that allows at most non-commercial usage by friends (again attribute-based restriction). In our policy model, only actions are allowed that explicitly modelled in the policy; all other actions are forbidden by default, thus Carol's friends are not allowed to further share her photos with other people.

$$\begin{aligned}\mathsf{p_c}(X) \leftarrow & \mathsf{Storing}(X) \wedge \mathsf{wasControlledBy}(X, \mathsf{SocialNetwork}) \wedge \\ & \mathsf{wasGenBy}(A, X) \wedge \mathsf{hasPolicy}(A, P) \wedge \mathsf{containedIn}(P, \mathsf{p_c}). \\ \mathsf{p_c}(X) \leftarrow & \mathsf{Sharing}(X) \wedge \mathsf{wasGenBy}(A, X) \wedge \\ & \mathsf{hasPolicy}(A, P) \wedge \mathsf{containedIn}(P, \mathsf{p_{friends}}) \wedge \\ & \mathsf{recipient}(X, R) \wedge \mathsf{hasFriend}(\mathsf{carol}, R).\end{aligned}$$

$$\begin{aligned}\mathsf{p_{friends}}(X) \leftarrow & \mathsf{Usage}(X) \wedge \mathsf{wasControlledBy}(X, F) \wedge \\ & \mathsf{hasFriend}(\mathsf{carol}, F) \wedge \mathsf{hasPurpose}(X, U) \wedge \\ & \mathsf{NonCommercial}(U).\end{aligned}$$

Dave's policy $\mathsf{p_d}$ for being notified about blog post sharings uses the attribute-based restriction pattern (posts can be shared only with registered users), the obligation pattern (a notification must be send after sharing), and the rights delegation pattern (receivers of the sharing action, can again share under the same conditions).

$$\begin{aligned}\mathsf{p_d}(X) \leftarrow & \mathsf{Sharing}(X) \wedge \mathsf{recipient}(X, R) \wedge \mathsf{RegisteredUser}(R) \wedge \\ & \mathsf{triggers}(X, N) \wedge \mathsf{Notification}(N) \wedge \mathsf{recipient}(N, \mathsf{dave}) \wedge \\ & \mathsf{wasGenBy}(A, X) \wedge \mathsf{hasPolicy}(A, P) \wedge \mathsf{containedIn}(P, \mathsf{p_d}).\end{aligned}$$

B. P3P Translation

P3P is a W3C standard for expressing privacy policies of Web sites [15]. Real world usage of P3P is limited [16], but nonetheless, we think that it is beneficial to show that our approach is able to formalise P3P policies. One of the drawbacks of P3P is the lack of a formally defined semantics, which can be mitigated by the translation to our policy language. Our translation restricts to the requested data usages expressed in P3P; leaving out information about the requesting entity and remedies, which are modelled in P3P, but which we consider orthogonal to our usage model and policy language. The notion of a P3P policy refers to a declaration which data is collected by a Web provider and in which ways it will be used, processed and shared. Thus, in our terminology, a P3P policy corresponds to a data request. For our request model, we adapted the P3P notions of data elements and categories, such that a P3P policy can be naturally transformed into a request for the corresponding data artefacts in our approach. However, P3P allows to specify that certain data elements, usage purposes and other aspects of the policy are optional (either opt-in or opt-out), which results in a set of requests in our formalism: one request for every combination of required and optional elements (see opt-in and opt-out pattern).

We base our translation on the formal semantics given to P3P by Yu et al. [17]. The semantics interprets a P3P policy as a set of tuples for three different relations:

```
d-purpose(data,purpose,required)
d-recipient(data,recipient,required)
d-retention(data,retention)
```

The `d-purpose` relation describes for which purposes some `data` element (identified by its URI) is used. The `required` field is either `opt-in`, `opt-out`, or `always`. The `d-recipient` relation describes with whom (`recipients` as defined in P3P) the `data` element is shared. The `d-retention` relation describes for how long a `data` element is stored, in terms of the P3P-defined `retention` periods. Furthermore Yu et al. define two further relations:

```
d-collection(data,optional)
d-category(data,category)
```

The `d-collection` relation describes whether it is `optional` or `required` to provide the `data` element. The `d-category` relation assigns `data` elements to P3P categories, which can be used to match user policies.

The optionalities given by the `required` and `optional` fields, as well as the specification, which data elements and categories are requested are translated into sets of different requests, as discussed above. What remains is the translation of the desired policy assigned to a request, which is the disjunction of the translation for every desired `purpose` into a usage action, every desired `recipient` into a sharing action, and every desired `retention` into a storage action.

P3P defines a number of purposes, each can be translated into a subclass of Purpose in our approach. Optionally the purposes can be put into a class hierarchy; Lämmel and Pek specify a partial order of purposes in terms of the amount of privacy preserved, which could be used for modelling a hierarchy [18]. For a `purpose` value `U`, we define the purpose concept PurposeU and model the following policy condition:

$$p(X) \leftarrow \mathsf{Usage}(X) \wedge \mathsf{hasPurpose}(X, U) \wedge \mathsf{PurposeU}(U).$$

Notable are the P3P defined purposes `pseudo-decision` and `pseudo-analysis`, when compared to `individual-decision` and `individual-analysis`: the meaning is that the data is pseudonymised before it is used for decisions or analysis, i.e. the same purposes meant by the `individual-...` purposes. As it is possible with our approach to model extended rights after pseudonymising data, we consider only the two purposes `decision` and `analysis`, and in the `pseudo` case model that data is first pseudonymised before used for these purposes.

A value `r` for a requested `recipient` is modelled as a sharing action. The `recipient` in P3P, however, does not refer to instances or classes of specific recipients, but uses some pre-defined constants: public, delivery, other-recipient, unrelated, ours, and same. As Lämmel and Pek state, nothing is known about the privacy policies of recipients other than ours and same [18]. We thus model sharing with one of these recipients as a sharing action with a resulting artefact that has a policy allowing every action. In our model, we do not differentiate between sharing with some unrelated entity that makes no statement about its privacy and the public. The recipient ours is not translated at all, as there is no sharing action to model. The same recipient is translated into a sharing action, where the resulting artefact has a policy, which is contained in the requested policy. This translation shows a strong point of our approach, because the meaning of "same" can be given explicitly to refer to the employed privacy policies.

Values for `retention` are translated into storing actions. Retention in P3P includes values for stating that data is deleted after being used for the stated purpose or after the legal requirement for storing is passed. For such values, it is not possible in a general approach without further background knowledge to specify absolute time spans, after which a deletion should be triggered. Therefore, such retention values are modelled as subclasses of storing. There are also P3P extensions, which allow the specification of concrete time spans, such as e.g. one year, which are translated into obligations to delete the data before the absolute time value to which the retention value evaluates.

VIII. Related Work

Park and Sandhu provide a systematic treatment of usage control with their UCON model [19], [20]. The model focuses on conditions on allowed usages and required obligations, when protected data is released into a system. The system is regarded as a closed environment in which all data usages take place (no sharing of the data with further parties is considered) and can thus be considered as a process-centric approach. The conceptual model was formalised using temporal logics by Janicke et al. [4] and Zhang et al. [5]. P3P is a W3C standard for privacy policies in the Web [15]. It is a XML language without formal semantics, which is supplied by various later publications, e.g. [17]. We discussed how P3P can be translated into our policy approach according to the semantics given by Yu et al. [17]. Process-centric views of P3P were taken to check compliance of BPEL or BPMN processes [21], [22]

Another approach to process-centric privacy policies is presented by Salas and Krishnan [23]. The differences to data-centric policies have been discussed in Section III.

The history-aware PAPEL policy language can be used for privacy policies [9]. To illustrate the difference between history-awareness and our approach, consider a policy presented in [9]: a patient record can only be shared after it is deidentified. PAPEL models the sharing action as compliant, if it was preceded by an deidentification. With our approach, the policy of the health record specifies that a compliant deidentification action can allow a policy for the produced artefact, which permits sharing; the policy engine evaluating the compliance of the sharing action does not have to know about the history of the artefact.

Becker et al. have developed the SecPAL authorization language [8], which supports rights delegation. SecPAL policies are not attached to data artefacts. The S4P language for privacy policies by Becker et al. [24] also evaluates policies on a system level, which requires history-awareness: e.g. the compliance of an action, allowed by a rights delegation requires the system to evaluate the rightfulness of the delegation.

The data-purpose algebra by Hanson et al. allows the modelling of usage restrictions of data and the transformation of the restrictions when data is processed [25]. In their approach, a data item is associated with allowed usages and depending on the process performed on a data item a function is defined that transforms the allowed usages. We share the general idea of having a set of allowed usages for data artefacts, which can change depending on the process performed on the artefact. In our approach, transformation functions are not defined directly but restricted by containedIn conditions on policies of produced artefacts. The data-purpose algebra is particularly suitable for expressing transformations which hold for all data items processed by a specific system. In contrast, our approach integrates the transformation functions, i.e., target policy restrictions, into policies of data artefacts, which means that every artefact can define its own transformation functions.

IX. Conclusions

We presented an approach for data-centric privacy policies. Compared to process-centric policies, our approach has the advantage that policies can be directly attached to data artefacts. A key feature is that policies are self-contained in the sense that a description of the action to be performed and the desired policy for produced artefacts are sufficient to check compliance. There is no need to model the history of an artefact or the further process in which it will be used. This proves to be an advantage in environments of service networks, where processes are designed dynamically and executed by frequently changing providers, as there is no need for a centralised policy monitor.

We have implemented the policy engine for checking compliance of action descriptions. For future work, we plan to integrate the engine with an execution environment, including components for obligation handling.

REFERENCES

[1] B. Parsia, V. Kolovski, and J. Hendler, "Expressing WS-Policies in OWL," in *Policy Management for the Web, Workshop at the World Wide Web Conference (WWW)*, 2005.

[2] P. A. Bonatti and F. Mogavero, "Comparing Rule-Based Policies," in *IEEE Workshop on Policies for Distributed Systems and Networks (POLICY)*, Jun. 2008, pp. 11–18.

[3] M. Krötzsch and S. Speiser, "ShareAlike your data: Self-referential usage policies for the Semantic Web," in *The Semantic Web (ISWC)*, 2011.

[4] H. Janicke, A. Cau, and H. Zedan, "A note on the formalisation of UCON," in *Symposium on Access Control Models and Technologies (SACMAT)*, 2007.

[5] X. Zhang, J. Park, F. Parisi-Presicce, and R. Sandhu, "A logical specification for usage control," in *Symposium on Access Control Models and Technologies (SACMAT)*, 2004.

[6] H. Takabi, J. B. D. Joshi, and G.-J. Ahn, "Security and privacy challenges in cloud computing environments," *IEEE Security & Privacy*, vol. 8, no. 6, pp. 24–31, 2010.

[7] P. A. Bonatti, J. L. De Coi, D. Olmedilla, and L. Sauro, "A Rule-based Trust Negotiation System," *IEEE Transactions on Knowledge and Data Engineering*, vol. 99, 2010.

[8] M. Y. Becker, C. Fournet, and A. D. Gordon, "SecPAL: Design and semantics of a decentralized authorization language," *Journal of Computer Security*, vol. 18, pp. 619–665, 2010.

[9] C. Ringelstein and S. Staab, "PAPEL : A Language and Model for Provenance-Aware Policy Definition and Execution," in *Business Process Management (BPM)*, 2010.

[10] C. S. Peirce, "Abduction and Induction," in *Philosophical writings of Peirce*. Dover Publications, 1955. [Online]. Available: http://books.google.com/books?id=ClSjXRIbxAMC

[11] D. Poole, "Explanation and prediction: an architecture for default and abductive reasoning," *Computational Intelligence*, vol. 5, pp. 97–110, May 1989.

[12] M. Y. Becker and S. Nanz, "The Role of Abduction in Declarative Authorization Policies," in *International Conference on Practical Aspects of Declarative Languages (PADL)*, 2008.

[13] J. O. Kephart and W. E. Walsh, "An Artificial Intelligence Perspective on Autonomic Computing Policies," in *Workshop on Policies for Distributed Systems and Networks (POLICY)*, 2004.

[14] S. Speiser, "Policy of composition $\neq$ composition of policies," in *IEEE International Symposium on Policies for Distributed Systems and Networks (POLICY)*, 2011.

[15] W3C, *The Platform for Privacy Preferences 1.0 (P3P1.0) Specification*. W3C Recommendation, 2002, http://www.w3.org/TR/P3P/.

[16] P. Beatty, I. Reay, S. Dick, and J. Miller, "P3P Adoption on E-Commerce Web sites: A Survey and Analysis," *IEEE Internet Computing*, vol. 11, pp. 65–71, 2007.

[17] T. Yu, N. Li, and A. I. Antón, "A formal semantics for P3P," in *Workshop on Secure Web Services (SWS)*, 2004.

[18] R. Lämmel and E. Pek, "Understanding privacy policies (A study in empirical language usage analysis)," submitted to the ESE Journal (special issue on ICPC 2010). Online since 17 Jan 2011. Major revision since 8 Aug 2011. Available at http://softlang.uni-koblenz.de/p3p/paper.pdf.

[19] J. Park and R. Sandhu, "Towards usage control models: beyond traditional access control," in *Symposium on Access Control Models and Technologies (SACMAT)*, 2002.

[20] J. Park and R. S. Sandhu, "The $\mathrm{UCON}_{\mathrm{ABC}}$ usage control model," *ACM Trans. Inf. Syst. Secur.*, vol. 7, no. 1, pp. 128–174, 2004.

[21] Y. H. Li, H.-Y. Paik, and B. Benatallah, "Formal Consistency Verification between BPEL Process and Privacy Policy," in *International Conference on Privacy, Security and Trust (PST)*, 2006.

[22] M. Chinosi and A. Trombetta, "Integrating privacy policies into business processes," *Journal of Research and Practice in Information Technology*, vol. 41, no. 2, pp. 155–170, 2009.

[23] P. P. Salas and P. Krishnan, "Testing Privacy Policies Using Models," in *IEEE International Conference on Software Engineering and Formal Methods (SEFM)*, 2008.

[24] M. Y. Becker, A. Malkis, and L. Bussard, "A Practical Generic Privacy Language," in *International Conference on Information Systems Security (ICISS)*, 2010.

[25] C. Hanson, T. Berners-Lee, L. Kagal, G. J. Sussman, and D. Weitzner, "Data-Purpose Algebra: Modeling Data Usage Policies," in *Workshop on Policies for Distributed Systems and Networks (POLICY)*, 2007.