ShareAlike Your Data: Self-referential Usage Policies for the Semantic Web

Markus Krötzsch[1] and Sebastian Speiser[2]

[1] Department of Computer Science, University of Oxford, UK
markus.kroetzsch@cs.ox.ac.uk
[2] Institute AIFB, Karlsruhe Institute of Technology, DE
speiser@kit.edu

Abstract. Numerous forms of policies, licensing terms, and related conditions are associated with Web data and services. A natural goal for facilitating the re-use and re-combination of such content is to model usage policies as part of the data so as to enable their exchange and automated processing. This paper thus proposes a concrete policy modelling language. A particular difficulty are *self-referential* policies such as *Creative Commons ShareAlike*, that mandate that derived content is published under some license with the same permissions and requirements. We present a general semantic framework for evaluating such recursive statements, show that it has desirable formal properties, and explain how it can be evaluated using existing tools. We then show that our approach is compatible with both OWL DL and Datalog, and illustrate how one can concretely model self-referential policies in these languages to obtain desired conclusions.

1 Introduction

Semantic technologies facilitate the sharing and re-use of data and associated services, but in practice such uses are often governed by a plethora of policies, licensing terms, and related conditions. Most data and service providers reserve certain rights, but an increasing number of providers also choose usage terms that encourage the re-use of content, e.g. by using a Creative Commons[1] license. Even such policies still impose restrictions, and it has been estimated that 70% – 90% of re-uses of Flickr images with Creative Commons Attribution license actually violate the license terms [29]. A possible reason for frequent violations is that checking license compliance is a tedious manual task that is often simply omitted in the process of re-using data.

A natural goal therefore is to accurately model usage policies as part of the data so as to enable their easy exchange and automated processing. This resonates with multiple topical issues in Semantic Web research. On the one hand, it is increasingly acknowledged that the distribution of semantic data and services may also require transparent licensing for such content [33,10]. This closely relates to the wider goal of semantically representing *provenance* information about the origin and context of data items. Not surprisingly, the W3C Incubator Group on Provenance also lists support for usage policies and licenses of artefacts in their requirements report [9].

[1] http://creativecommons.org/

L. Aroyo et al. (Eds.): ISWC 2011, Part I, LNCS 7031, pp. 354–369, 2011.

On the other hand, modelling of policy information is also promising as an application area for semantic technologies [17,7]. Capturing the variety of relevant conditions involves domain-specific concepts such as "non-commercial" or "fair use" but also (when thinking about distribution policies that are internal to an organisation) levels of confidentiality, and personal access permissions. Semantic technologies offer powerful tools and methodologies for developing shared conceptualisations for such complex modelling problems.

This paper presents a new policy modelling language to address the specific challenges of this domain. A primary task is to enable the computation of policy containment, i.e. the automatic decision whether all uses that are allowed by one policy are also allowed by another [8]. But some policies go a step further and require such containments to hold *as part of their condition*. A well-known example are the Creative Commons ShareAlike licenses which mandate that content is published under some license that involves the same permissions and requirements – including the requirement to share under such licenses only. Such self-referential policies introduce recursive dependencies and a form of meta-modelling not found in ontology languages like OWL.

Our main contributions to solving this problem are as follows.

(1) We develop the syntax and semantics of a general policy modelling language. Our formalisation is guided by an analysis of the requirements for a policy (meta) model that supports self-referential policies as given by the Creative Commons licenses.
(2) We show that this policy language has desirable formal properties under reasonable syntactic restrictions on policy conditions and background theories. In particular we establish how to utilise standard first-order reasoning in a non-trivial way for computing conclusions under our new semantics.
(3) Using this connection to first-order logic, we instantiate this general policy language for the Web Ontology Language OWL and for the basic rule language Datalog. Both cases lead to expressive policy representation languages that can readily be used in practice by taking advantage of existing tools. Concretely, we show how to express the well-known Creative Commons licenses and verify that the expected relationships are derived.

Section 2 introduces our main use case and Section 3 presents a basic vocabulary to model policies. In Section 4 we discuss challenges in modelling self-referential policies formally. We introduce a formal policy semantics in Section 5 and apply it to our use case in Section 6. Related work is discussed in Section 7. The technical results at the core of this paper are not obvious and require a notable amount of formal argumentation. However, the focus of this presentation is to motivate and explain the rationale behind our proposal. Formal proofs and further details are found in an extended report [20].

2 Use Case: Creative Commons ShareAlike

To motivate our formalisation of policies we discuss some common requirements based on the popular Creative Commons (CC) licenses. CC provides a family of license models for publishing creative works on the Web, which share the common goal of enabling re-use as an alternative to the "forbidden by default" approach of traditional copyright

law. Each license specifies how the licensed work may be used by stating, e.g., in which cases it can be further distributed (shared) and if derivative works are allowed.

The most permissive CC license is *Creative Commons Attribution* (CC BY), which allows all types of uses (sharing and derivation) provided that the original creator of the work is attributed. Various restrictions can be added to CC BY:

- NoDerivs (ND): the work can be used and redistributed, but it must remain unchanged, i.e., no derivations can be created.
- NonCommercial (NC): re-use is restricted to non-commercial purposes.
- ShareAlike (SA): derived works have to be licensed under the identical terms.

The CC ShareAlike restriction is particularly interesting, as it does not only restrict processes using the protected data artefact, but the policy of artefacts generated by those processes. ShareAlike is formulated in legal code as follows:

> "You may Distribute or Publicly Perform an Adaptation only under: (i) the terms of this License; (ii) a later version of this License [...]; (iii) a Creative Commons jurisdiction license [...] that contains the same License Elements as this License [...]"[2]

Thus derived artefacts can only be published under some version of the exact same CC license. This could easily be formalised by simply providing an exhaustive list of all licenses that are currently admissible for derived works. In this case, policies would be identified by their name, not by the permissions and restrictions that they impose.

This effect can be desired, e.g. for the GPL which thus ensures its "viral" distribution. However, the name-based restriction is not intended for Creative Commons, as noted by Lessig who originally created CC: rather, it would be desirable to allow the combination of licenses that share the same intentions but that have a different name, e.g. to specify that an artefact must be published under a license that allows only non-commercial uses instead of providing a list of all (known) licenses to which this characterisation applies [21]. To overcome this incompatibility problem, we propose *content-based* policy restrictions that are based on the allowed usages of a policy.

3 Schema for Modelling Policies

Before we can formally specify the semantics of a policy language that can formalise the "intention" of a policy like CC, we need some basic conceptual understanding of the modelling task, and also some shared vocabulary that enables the comparison of different licenses. In this section, we provide a high-level schema that we use for modelling policies in this paper.

In general, we understand a policy as a specification that defines what one is allowed to *do* with an artefact that has this policy. Thus, a policy can be viewed as a collection of admissible usages. In order to align with the terminology of the Open Provenance Model OPM [23] below we prefer to speak of admissible "processes" as the most general type of use. The admissible processes can be viewed as "desired states"

[2] Section 4(b) in `http://creativecommons.org/licenses/by-nc-sa/3.0/legalcode`

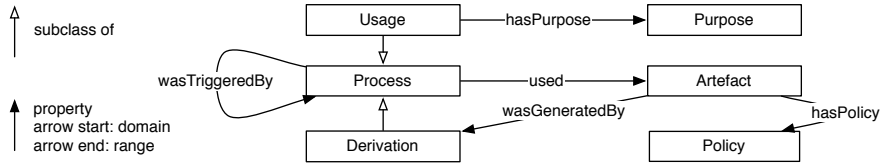

Fig. 1. Informal view of a simple provenance model

(in the sense of "states of the world" such as when an artefact has been published with suitable attribution), which corresponds to the notion of goal-based policies as defined by Kephart and Walsh [19].

To specify the conditions of a policy, we need a model for further describing such usage processes and their relationships to concrete artefacts. This model in particular must represent the origin of the artefact, and the context in which it has been published. Such *provenance* information can be described in various ways, e.g. with a provenance graph that specifies the dependencies between processes and the artefacts they use and generate. Here we use the very simple provenance model illustrated informally in Fig. 1. This base model can of course be further specialised for specific applications and other use cases; we just require a minimal setup for our examples.

The provenance model re-uses the vocabulary elements *artefact*, *process*, *used*, *wasGeneratedBy*, and *wasTriggeredBy* from the Open Provenance Model. For our particular application, we further split processes into *derivation*s (processes that generate a new artefact) and other *usage*s that only use artefacts without change. To cover the CC use case, we introduce the *hasPurpose* property relating a usage to its *purpose*, e.g., stating that a usage was non-commercial. The *hasPolicy* property assigns to an artefact a *policy*, which means that all processes using the artefact are (legally) required to comply to its policy.

According to OPM, a process p_1 *wasTriggeredBy* another process p_2, if p_1 can only have started after p_2 started. So, somewhat contrary to intuition, the "triggering" is rather a precondition but not a necessary cause of the triggered one. A usage restriction that requires attribution would thus be formalised as a policy requiring that the usage process *wasTriggeredBy* an attribution process, and not the other way around.

The provenance model provides a basic vocabulary for specifying information about artefacts and policies. To realise content-based restrictions we further want to talk about the relationships of policies. For example, ShareAlike requires the value of *hasPolicy* to refer to a policy which allows exactly the same uses as the given CC SA license. This subsumption between policies is called *policy containment*, and we introduce a predicate containedIn to express it. Informally speaking, the fact containedIn(p, q) can also be read as: any process that complies with policy p also complies with policy q. When allowing policy conditions to use containedIn, the question whether or not a process complies to a policy in turn depends on the evaluation of containedIn. Our goal therefore is to propose a formal semantics that resolves this recursive dependency in a way that corresponds to our intuitive understanding of the policies that occur in practice.

4 Challenges of Defining a Semantics for Policies

For formalising our above understanding of policies, we use the syntax of first-order logic as a general framework. Thus, our earlier 'classes' and 'properties' become predicates of arity 1 and 2, respectively. A policy that represents a set of allowed processes then corresponds to a formula $\varphi[x]$ with one free variable x, representing the set of individuals that make $\varphi[x]$ true when assigned as values to x.[3] For example, a policy p that allows no uses other than derivations that generate artefacts with policy p can be described as:

$$p : \mathsf{Derivation}(x) \wedge \exists y.(\mathsf{wasGeneratedBy}(y, x) \wedge \mathsf{hasPolicy}(y, p)). \tag{1}$$

More generally, we can use containedIn to allow derived artefacts to use any policy that is at least as restrictive as p:

$$\begin{aligned} p : \mathsf{Derivation}(x) \wedge \exists y.(&\mathsf{wasGeneratedBy}(y, x) \wedge \\ &\exists z.(\mathsf{hasPolicy}(y, z) \wedge \mathsf{containedIn}(z, p))). \end{aligned} \tag{2}$$

A collection of such policy definitions $p : \varphi_p[x]$ will be called a *policy system*. Given a policy system with definitions $p : \varphi_p$ for all policy names $p \in N_P$, we can formalise some general restrictions that conform to our intuition:

$$\forall x.\mathsf{conformsTo}(x, p) \leftrightarrow \varphi_p[x] \qquad \text{for all } p \in N_P, \tag{3}$$

$$\forall x, y.\mathsf{containedIn}(x, y) \leftrightarrow \forall z.(\mathsf{conformsTo}(z, x) \rightarrow \mathsf{conformsTo}(z, y)). \tag{4}$$

Formula (3) defines conformsTo to relate processes to the policies they conform to. Please note the difference between conformsTo (actual semantic conformance) and hasPolicy (legally required conformance). Formula (4) ensures that containedIn relates two policies exactly if fewer (or at most the same) processes conform to the first, i.e. if the first policy is at least as restrictive as the second. The set of these two types of sentences (for a given set of policy names N_P) is denoted by T_{ct}.

Unfortunately, these formulae under first-order semantics do not lead to the intended interpretation of policies. Consider the policy (2), and a second policy q that is defined by exactly the same formula, but with p replaced by q. Intuitively, p and q have the same conditions but merely different names, so they should be in a mutual containedIn relationship. Indeed, there are first-order models of T_{ct} where this is the case: if $\mathsf{containedIn}(p, q)$ holds, then $\forall x.\varphi_p[x] \rightarrow \varphi_q[x]$ is also true. However, this is not the only possible interpretation: if $\mathsf{containedIn}(p, q)$ does not hold, then $\forall x.\varphi_p[x] \rightarrow \varphi_q[x]$ is not true either. First-order logic does not prefer one of these interpretations, so in consequence we can conclude neither $\mathsf{containedIn}(p, q)$ nor $\neg\mathsf{containedIn}(p, q)$.

Working with first-order interpretations still has many advantages for defining a semantics, in particular since first-order logic is widely known and since many tools and knowledge representation languages are using it. This also enables us to specify additional background knowledge using first-order formalisms of our choice, e.g. the OWL DL ontology language. However, we would like to restrict attention to first-order

[3] We assume basic familiarity with first-order logic. Formal definitions are given in [20].

models that conform to our preferred reading of containedIn. Logical consequences can still be defined as the statements that are true under all of the preferred interpretations, but undesired interpretations will be ignored for this definition. Our goal of defining the semantics of self-referential policies thus boils down to defining the "desired" interpretations of a given first-order theory that uses containedIn. To do this, we propose a semantics for policy containment that, intuitively speaking, always prefers containedIn(p, q) to hold if this is possible without making additional unjustified assumptions. For illustration, consider the following policy q that further restricts p from (2) to non-commercial uses:

$$q : \mathsf{Derivation}(x) \wedge \forall w.(\mathsf{hasPurpose}(x, w) \rightarrow \mathsf{NonCommercial}(\mathsf{w})) \wedge \exists y.(\mathsf{wasGeneratedBy}(y, x) \wedge \exists z.(\mathsf{hasPolicy}(y, z) \wedge \mathsf{containedIn}(z, q))). \tag{5}$$

Though the policy q is clearly more restrictive than p, there still is a first-order interpretation that satisfies containedIn(p, q) by simply assuming that all things that conform to p happen to have non-commercial uses only. Nothing states that this is not the case, yet we do not want to make such assumptions to obtain more containedIn conclusions.

We thus distinguish *basic predicates* such as NonCommercial and hasPolicy from the two "special" predicates containedIn and conformsTo. Basic predicates are given by the data, and represent the available information, and their interpretation should not be considered a matter of choice. Special predicates in turn should be interpreted to reflect our intended understanding of policy containment, and as shown in the above example it is often desirable to maximise containedIn entailments. In other words, we would like to ensure that the consideration of a policy system does not lead to new logical consequences over basic predicates – merely defining license conditions should not increase our knowledge of the world. More formally: the policy semantics should be *conservative* over first-order semantics w.r.t. sentences that use only basic predicates.

Unfortunately, this is not easy to accomplish, and indeed Theorem 1 only achieves a limited version of this. One reason is that even T_{ct} may entail undesired consequences. Consider policies as follows (we use abstract examples to highlight technical aspects):

$$p : \ A(x) \wedge \mathsf{containedIn}(p, q) \qquad\qquad q : \ B(x). \tag{6}$$

This policy system entails containedIn(p, q). Indeed, if containedIn(p, q) would not hold, then nothing would conform to p by (3). But the empty set is clearly a subset of every other set, hence containedIn(p, q) would follow by (4). Thus all interpretations that satisfy T_{ct} must satisfy $\forall x.A(x) \wedge \mathsf{containedIn}(p, q) \rightarrow B(x)$, and thus $\forall x.A(x) \rightarrow B(x)$ is a consequence over basic predicates. Clearly, the mere definition of licenses should not entail that some otherwise unrelated class A is a subclass of B.

5 A Formal Language for Policy Definitions

In order to address the challenges discussed in the previous section, we now formally define a policy language. More precisely, we define a language for policies *and* a first-order language that is to be used for background theories. These definitions are intended to be very general to impose only those restrictions that we found necessary to obtain a

well-behaved semantics. Section 6 shows how this general framework can be instantiated in various well-known modelling languages.

The basic restriction that we impose on the logic is *connectedness*. Intuitively, this ensures that a formula can only refer to a connected relational structure of individuals. In our setting the conformance of a process to a policy thus only depends on the characteristics of individuals directly or indirectly reachable from the process. We argue that this is a small restriction. It might even be a best practice for "controlled" modelling in an open environment like the Web, as it ensures that the classification of any object is based only on its "environment" and not on completely unrelated individuals.

Our formal definition is reminiscent of the *Guarded Fragment* (GF) of first-order logic [4] and indeed it can be considered as a generalization of GF, though without the favourable formal properties that motivated GF. We first define open connected formulae (with free variables) and then closed ones. We write $\varphi[\boldsymbol{x}]$ to indicate that φ has at most the free variables that occur in $\boldsymbol{x}$ (or possibly less). For technical reasons, our first definition distinguishes "guard predicates" that must not use constant symbols from "non-guard predicates" where constants are allowed:

Definition 1. *Consider a first-order signature Σ where each predicate in Σ is marked as a* guard predicate *or as a* non-guard predicate. *The* connected open fragment *COF of first-order logic over Σ is the smallest set of formulae over Σ that satisfies the following properties:*

1. *Every atomic formula $p(\boldsymbol{t})$ with $\boldsymbol{t}$ a vector of terms that contain at least one variable belongs to COF, provided that $\boldsymbol{t}$ contains only variables if p is a guard predicate.*
2. *If φ_1 and φ_2 are in COF then so are $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, and $\varphi_1 \rightarrow \varphi_2$.*
3. *Consider a formula $\varphi[\boldsymbol{x}, \boldsymbol{y}]$ in COF, and a conjunction $\alpha[\boldsymbol{x}, \boldsymbol{y}] = \alpha_1[\boldsymbol{x}, \boldsymbol{y}] \wedge \ldots \wedge \alpha_n[\boldsymbol{x}, \boldsymbol{y}]$ of atomic formulae α_i that contain only guard predicates and variables, such that $\boldsymbol{x}$, $\boldsymbol{y}$ are both non-empty and do not share variables. Then the formulae*

$$\exists \boldsymbol{y}.\alpha[\boldsymbol{x}, \boldsymbol{y}] \wedge \varphi[\boldsymbol{x}, \boldsymbol{y}] \qquad \forall \boldsymbol{y}.\alpha[\boldsymbol{x}, \boldsymbol{y}] \rightarrow \varphi[\boldsymbol{x}, \boldsymbol{y}],$$

are in COF provided that for each variable y in $\boldsymbol{y}$, there is some variable x in $\boldsymbol{x}$ and some atom $\alpha_i[\boldsymbol{x}, \boldsymbol{y}]$ where both x and y occur.

The distinction of guard and non-guard predicates is important, but a suitable choice of guard predicates can be easily made for a given formula set of formulae in COF by simply using exactly those predicates as guards that do not occur in atomic formulae with constants. The only predicate that we really need to be a non-guard is containedIn. Therefore, we will omit the explicit reference to the signature Σ in the following and simply assume that one signature has been fixed.

Definition 2. *The* connected fragment *CF of first-order logic consists of the following sentences:*

- *Every formula without variables is in CF.*
- *If $\varphi[x]$ is a COF formula with one free variable x, then $\forall x.\varphi[x]$ and $\exists x.\varphi[x]$ are in CF.*

We will generally restrict to background theories that belong to CF. As discussed in Section 6 below, large parts of OWL DL and Datalog fall into this fragment. A typical example for a non-CF sentence is the formula $\neg\exists x.A(x) \vee \neg\exists x.B(x)$. Also note that the formulae (3) and (4) of T_{ct} are not in CF – we consider them individually in all our formal arguments. On the other hand, the policy conditions (1), (2), (5), and (6) all are in COF. Using the terminology of connected formulae, we can define policy conditions, policy descriptions, and policy systems that we already introduced informally above:

Definition 3. *Let N_P be a set of* policy names. *A* policy condition *φ for N_P is a formula that may use an additional binary predicate* containedIn *that cannot occur in background theories, and where:*

- *φ is a COF formula with one free variable,*
- *φ contains at most one constant symbol $p \in N_P$ that occurs only in atoms of the form* $\mathsf{containedIn}(y, p)$ *or* $\mathsf{containedIn}(p, y)$*,*
- *every occurrence of* containedIn *in φ is positive (i.e. not in the scope of a negation) and has the form* $\mathsf{containedIn}(y, p)$ *or* $\mathsf{containedIn}(p, y)$*.*

A policy description *for a policy $p \in N_P$ is a pair $\langle p, \varphi\rangle$ where φ is a policy condition. A* policy system *P for N_P is a set of policy descriptions that contains exactly one description for every policy $p \in N_P$.*

This definition excludes the problematic policy p in (6) above while allowing (1), (2), and (5). Moreover, it generally requires containedIn to be a non-guard predicate.

We define the semantics of policy containment as the greatest fixed point of an operator introduced next. Intuitively, this computation works by starting with the assumption that all named policies are contained in each other. It then refers to the policy definitions to compute the actual containments that these assumptions yield, and removes all assumptions that cannot be confirmed. This computation is monotone since the assumptions are reduced in each step, so it also has a greatest fixed point.

Definition 4. *Consider a set of CF sentences T (background theory), a set of policy names N_P that includes the* top policy *$p_\top$ and the* bottom policy *$p_\bot$, and a policy system P for N_P such that $\langle p_\top, \top(x)\rangle, \langle p_\bot, \bot(x)\rangle \in P$.*[4] *Let T_{ci} be the following theory:*

$$T_{\mathsf{ci}} = \{\forall x, y, z.\mathsf{containedIn}(x, y) \wedge \mathsf{containedIn}(y, z) \rightarrow \mathsf{containedIn}(x, z),\\ \forall x.\mathsf{containedIn}(x, p_\top), \forall x.\mathsf{containedIn}(p_\bot, x)\}.$$

For a set $C \subseteq N_P^2$, define $\mathsf{CI}(C) := \{\mathsf{containedIn}(p, q) \mid \langle p, q\rangle \in C\}$. An operator $P_T : \mathcal{P}(N_P^2) \rightarrow \mathcal{P}(N_P^2)$,where $\mathcal{P}(N_P^2)$ is the powerset of N_P^2, is defined as follows:

$$P_T(C) = \{\langle p, q\rangle \mid \langle p, \varphi_p\rangle, \langle q, \varphi_q\rangle \in P \text{ and } T \cup T_{\mathsf{ci}} \cup \mathsf{CI}(C) \models \forall x.\varphi_p[x] \rightarrow \varphi_q[x]\}.$$

Proposition 1. *The operator P_T has a greatest fixed point* $\mathsf{gfp}(P_T)$ *that can be obtained by iteratively applying P_T to N_P^2 until a fixed point is reached. More concretely, the greatest fixed point is of the form $P_T^n(N_P^2)$ for some natural number $n \leq |N_P|^2$ where P_T^n denotes n-fold application of P_T.*

[4] As usual, we consider $\top/\bot$ as unary predicates that are true/false for all individuals.

The fact that P_T requires the existence of policies $p_\top$ and $p_\bot$ is not restricting the applicability of our approach since the according standard policy declarations can always be added. Using the greatest fixed point of P_T, we now define what our "preferred" models for a policy system and background theory are.

Definition 5. *Given a policy system P, a P-*model *for a theory T is a first-order interpretation $\mathcal{I}$ that satisfies the following theory:*

$$\mathcal{I} \models T \cup T_{\mathsf{ci}} \cup \mathsf{CI}(\mathsf{gfp}(P_T)) \cup T_{\mathsf{ct}}, \tag{7}$$

where T_{ci} and $\mathsf{CI}(\mathsf{gfp}(P_T))$ are as in Definition 4, and where T_{ct} is the collection of all sentences of the form (3) *and* (4)*. In this case, we say that $\mathcal{I}$ P-*satisfies *T. A sentence φ is a P-*consequence of *T, written $T \models_P \varphi$, if $\mathcal{I} \models \varphi$ for all P-models $\mathcal{I}$ of T.*

It is essential to note that the previous definition uses a fixed point computation only to obtain a minimal set of containments among named policies that must be satisfied by all P-models. It is not clear if and how the semantics of P-models could be captured by traditional fixed point logics (cf. Section 7). At the core of this problem is that policy conformance is inherently non-monotonic in some policies that we want to express. A policy p might, e.g., require that the policy of all derived artefacts admits *at least* all uses that are allowed by p. Then the fewer uses are allowed under the p, the more policies allow these uses too, and the more uses conform to p. This non-monotonic relationship might even preclude the existence of a model.

The policy semantics that we defined above is formal and well-defined for all policy systems and background theories, even without the additional restrictions of Definition 2 and 3. However, three vital questions have to be answered to confirm that it is appropriate for our purpose: (1) How can we compute the entailments under this new semantics? (2) Does this semantics avoid the undesired conclusions discussed in Section 4? (3) Does the semantics yield the intended entailments for our use cases? The last of these questions will be discussed in Section 6. Questions (1) and (2) in turn are answered by the following central theorem of this paper:

Theorem 1. *Consider a theory T and a policy system P. For every φ that is a CF formula over the base signature, or a variable-free atom (fact) over the predicates* containedIn *or* conformsTo *we have:*

$$T, T_{\mathsf{ci}}, \mathsf{CI}(\mathsf{gfp}(P_T)), T_{\mathsf{ct}}^- \models \varphi \qquad \textit{iff} \qquad T \models_P \varphi, \tag{8}$$

where T_{ci} and $\mathsf{CI}(\mathsf{gfp}(P_T))$ are defined as in Definition 4, and where T_{ct}^- is the collection of all sentences of the form (3)*.*

Let us first discuss how Theorem 1 answers the above questions.

(1) The theorem reduces P-entailment to standard first-order logic entailment. Since $\mathsf{gfp}(P_T)$ can be computed under this semantics as well, this means that reasoning under our semantics is possible by re-using existing tools given that one restricts to fragments of (CF) first-order logic for which suitable tools exist. We pursue this idea in Section 6.

(2) The theorem asserts that all CF formulae that are P-entailments are entailed by the first-order theory $T \cup T_{\mathsf{ci}} \cup \mathsf{Cl}(\mathsf{gfp}(P_T))$. It is easy to see that T_{ci} and $\mathsf{Cl}(\mathsf{gfp}(P_T))$ only affect the interpretation of formulae that use containedIn. All other CF formulae are P-entailments of T if and only if they are first-order entailments of T. Thus, new entailments over base predicates or even inconsistencies are not caused by considering a policy system.

The proof of Theorem 1 is not straightforward. At its core, it hinges on the fact that every model $\mathcal{I}$ of $T \cup T_{\mathsf{ci}} \cup \mathsf{Cl}(\mathsf{gfp}(P_T))$ can be extended into a P-model $\hat{\mathcal{I}}$ of T that satisfies no containedIn or conformsTo facts that have not already been satisfied by $\mathcal{I}$. Constructing this P-model requires a number of auxiliary constructions centred around the idea that, for every policy containment not in $\mathsf{Cl}(\mathsf{gfp}(P_T))$, one can find a witness (a process conforming to the one policy but not to the other) in some model of $T \cup T_{\mathsf{ci}} \cup \mathsf{Cl}(\mathsf{gfp}(P_T))$. This witness (and all of its environment) is then copied into the P-model that we want to construct. This is only feasible since the CF formulae in T are inherently "local" and will not change their truth value when extending the model by new (disjoint) individuals. After enough witnesses have been included to refute all non-entailed containedIn facts, the construction of $\hat{\mathcal{I}}$ is completed by defining suitable extensions for conformsTo where care is needed to do this for "unnamed" policies so that T_{ct} is satisfied. A full formal argument is found in the technical report [20].

6 Practical Policy Languages

In this section, we provide concrete instantiations of the general formalism introduced above. The CF fragment still is overly general for practical use, in particular since the computation of entailments in this logic is undecidable which precludes many desired applications where policy containment would be checked automatically without any user interaction.[5] However, Theorem 1 asserts that we can generally evaluate formal models under the semantics of first-order logic which is used in many practical knowledge representation languages. By identifying the CF fragments of popular modelling formalisms, we can therefore obtain concrete policy modelling languages that are suitable for specific applications.

There are various possible candidates for knowledge representation languages that can be considered under a first-order semantics and for which good practical tool support is available. Obvious choices include the Web Ontology Language OWL under its Direct Semantics [32], and the rule language Datalog under first-order semantics [3] which we will discuss in more detail below.

As we will explain for the case of Datalog, one can also model policy conditions as (conjunctive/disjunctive) queries with a single result, given that the query language uses a first-order semantics. Query evaluation is known to be difficult for expressive modelling languages, but can be very efficient when restricting to a light-weight background theory. A possible example is the combination of SPARQL for OWL [11] with

[5] This is easy to see in many ways, for example since (as noted below) CF allows us to express description logics like $\mathcal{SRIQ}$, whereas CF does not impose the regularity or acyclicity conditions that are essential for obtaining decidability of reasoning in these logics [15].

the lightweight OWL QL or OWL RL languages [32]. The below cases thus can only serve as an illustration of the versatility of our approach, not as a comprehensive listing.

6.1 Modelling Policies in OWL DL

The Direct Semantics of OWL 2 is based on description logics which in turn are based on the semantics of first-order logic [32]. The ontology language OWL 2 DL for which this semantics is defined can therefore be viewed as a fragment of first-order logic to which we can apply the restrictions of Section 5. The standard translation to first-order logic (see, e.g., [14]) produces formulae that are already very close to the syntactic form of CF sentences described above. Moreover, OWL class expressions are naturally translated to first-order formulae with one free variable, and are thus suitable candidates for expressing policies. Policy containment then corresponds to class subsumption checking – a standard inferencing task for OWL reasoners. The binary predicates of our simple provenance model, as well as the special predicates containedIn and conformsTo can be represented by OWL properties, whereas unary predicates from the provenance model correspond to primitive OWL classes.

Some restrictions must be taken into account to ensure that we consider only ontologies that are CF theories, and only classes that are valid policy conditions. Nominals (enumerated classes as provided by `ObjectOneOf` in OWL) are expressed in first-order logic using constant symbols, and must therefore be excluded from background ontologies. On the other hand nominals must be used in containedIn in policy descriptions (in OWL this particular case can conveniently be expressed with `ObjectHasValue`). Besides nominals, the only non-connected feature of OWL 2 that must be disallowed is the universal role (`owl:topObjectProperty`). On the other hand, cardinality restrictions are unproblematic even though they are usually translated using a special built-in equality predicate $\approx$ that we did not allow in first-order logic in Section 5. The reason is that $\approx$ can easily be emulated in first-order logic using a standard equality theory [20], so that all of our earlier results carry over to this extension.

To apply Theorem 1 for reasoning, we still must be able to express T_{ci} of Definition 4 in OWL. Transitivity of containedIn is directly expressible, and the remaining axioms can be written as follows:[6]

$$\top \sqsubseteq \exists\, \mathsf{containedIn}.\{p_\top\} \qquad\qquad \top \sqsubseteq \exists\, \mathsf{containedIn}^-.\{p_\bot\}$$

Note that the represented axioms are not in CF, and likewise the restriction to nominal-free OWL is not relevant here.

Concrete policies are now easily modelled. The public domain (PD) policy that allows every type of usage and derivation is expressed as:

$$\mathrm{PD}\colon \mathsf{Usage} \sqcup \mathsf{Derivation}\,.$$

Processes compliant to CC BY are either usages that were triggered by some attribution, or derivations for which all generated artefacts have only policies that also require

[6] Throughout this section we use the usual DL notation for concisely writing OWL axioms and class expressions; see [14] for an extended introduction to the relationship with OWL 2 syntax.

attributions, i.e., which are contained in BY:

$$\text{BY}: (\mathsf{Usage} \sqcap \exists\, \mathsf{wasTriggeredBy}.\mathsf{Attribution}) \sqcup (\mathsf{Derivation} \sqcap \forall\, \mathsf{wasGeneratedBy}^{-1}.\forall\, \mathsf{hasPolicy}.\exists\, \mathsf{containedIn}.\{\text{BY}\}).$$

To account for the modular nature of CC licenses, it is convenient to re-use class expressions as the one for BY. Thus, we will generally write C_{BY} to refer to the class expression for BY, and similarly for the other policies we define. To define NoDerivs (ND) licenses that allow all processes that are not derivations, we introduce C_{ND} as an abbreviation for $\mathsf{Process} \sqcap \neg\mathsf{Derivation}$. We can thus express CC BY-ND as

$$\text{BY-ND}: C_{\text{BY}} \sqcap C_{\text{ND}}.$$

The ShareAlike (SA) condition cannot be modelled as an independent building block, as it refers directly to the policy in which it is used. As an example, we model the condition for the CC BY-SA policy as a requirement that all policies of all generated artefacts are equivalent to BY-SA, i.e., they are contained in BY-SA and BY-SA is contained in them:

$$\text{BY-SA}: C_{\text{BY}} \sqcap \forall\, \mathsf{wasGeneratedBy}^{-1}.\forall\, \mathsf{hasPolicy}.(\exists\, \mathsf{containedIn}.\{\text{BY-SA}\} \sqcap \exists\, \mathsf{containedIn}^{-1}.\{\text{BY-SA}\}).$$

To validate the basic practicability of this modelling approach, we used the OWL reasoner HermiT[7] to compute the fixed point semantics of the policy system. We then conducted some basic tests with the formalised CC policies.[8] Not surprisingly, it can be observed that the fixed point of P_T is reached after just 2 iterations, which is significantly less than the rough upper bound of $|N_P|^2$ which was 49 in case of the 7 CC licenses. In general, one may presume that even big numbers of policies do rarely expose a linear dependency that would lead to long iterations for reaching a fixed point.

As a basic example of how to apply automated conformance checking, we modelled for every combination $(p_{\text{orig}}, p_{\text{deriv}})$ of Creative Commons licenses a derivation which uses an artefact with policy p_{orig} and generates a new artefact with policy p_{deriv}. If such a derivation is compliant to p_{orig}, we know that p_{deriv} is a valid license for derivations of p_{orig} licensed artefacts. The results (as expected) agree with the official Creative Commons compatibility chart.[9]

It can be noted that, besides its use for conformance checking, the computation of containedIn can also assist in modelling policies. For example, one can readily infer that any ShareAlike (SA) requirement is redundant when a NoDerivs (ND) requirement is present as well: adding SA to any ND license results in an equivalent license, i.e. one finds that the licenses are mutually contained in each other.

[7] http://www.hermit-reasoner.com/

[8] For reasons of space, we did not include all formalisations for all CC licenses here; the complete set of example policies for OWL and Datalog is available at http://people.aifb.kit.edu/ssp/creativecommons_policies.zip

[9] see Point 2.16 in http://wiki.creativecommons.org/FAQ, accessed 15th June 2011

6.2 Modelling Policies in Datalog

Datalog is the rule language of function-free definite Horn clauses, i.e., implications with only positive atoms and a single head atom. It can be interpreted under first-order semantics [3]. The syntax corresponds to first-order logic with the only variation that quantifiers are omitted since all variables are understood to be quantified universally. Datalog rules can thus be used to express a background theory. Policies can be expressed by conjunctive or disjunctive queries, i.e., by disjunctions and conjunctions of atomic formulae where one designated variable represents the free variable that refers to the conforming processes, while the other variables are existentially quantified.

Again we have to respect syntactic restrictions of Section 5. Thus we can only use rules that are either free of variables, or that contain no constants. In the latter case, all variables in the rule head must occur in its body (this is known as *safety* in Datalog), and the variables in the rule body must be connected via the atoms in which they co-occur. For policy queries, we also require this form of connection, and we allow constants in containedIn. The (non-CF) theory T_{ci} of Definition 4 is readily expressed in Datalog.

Containment of conjunctive and disjunctive queries is decidable, and can be reduced to query answering [2]. Namely, to check containment of a query q_1 in a query q_2, we first create for every conjunction in q_1 (which is a disjunction of conjunctive queries) a grounded version, i.e., we state every body atom in the conjunction as a fact by uniformly replacing variables with new constants. If, for each conjunction in q_1, these new facts provide an answer to the query q_2, then q_1 is contained in q_2. Note that Datalog systems that do not support disjunctive query answering directly can still be used for this purpose by expressing disjunctive conditions with multiple auxiliary rules that use the same head predicate, and querying for the instances of this head.

As above, the simplest policy is the public domain (PD) license:

$$\mathrm{PD}\colon \mathsf{Usage}(x) \vee \mathsf{Derivation}(x).$$

Here and below, we always use x as the variable that represents the corresponding process in a policy description. CC BY can now be defined as follows:

$$\begin{aligned}\mathrm{BY}\colon\ & (\mathsf{Usage}(x) \wedge \mathsf{wasTriggeredBy}(x, y) \wedge \mathsf{Attribution}(y)) \vee \\ & (\mathsf{Derivation}(x) \wedge \mathsf{wasGeneratedBy}(z, x) \wedge \\ & \mathsf{hasPolicy}(z, v) \wedge \mathsf{containedIn}(v, \mathrm{BY}))\ .\end{aligned}$$

This formalisation alone would leave room for derivations that are falsely classified as compliant, since the condition only requires that there exists one artefact that has one contained policy. Further artefacts or policies that violate these terms might then exist. We can prevent this by requiring hasPolicy to be *functional* and wasGeneratedBy to be *inverse functional* (as before, we assume that $\approx$ has been suitably axiomatised, which is possible in Datalog; see [20] for details):

$$\begin{aligned}v_1 \approx v_2 &\leftarrow \mathsf{hasPolicy}(x, v_1) \wedge \mathsf{hasPolicy}(x, v_2),\\ z_1 \approx z_2 &\leftarrow \mathsf{wasGeneratedBy}(z_1, x) \wedge \mathsf{wasGeneratedBy}(z_2, x)\ .\end{aligned}$$

Using this auxiliary modelling, we can easily express BY-ND and BY-SA as well [20].

7 Related Work

The formalisation of policies and similar restrictions has been considered in many works, but the relationship to our approach is often limited. For example, restrictions in Digital Rights Management (DRM) systems can be specified in a rights expression language such as ODRL [16]. Policy containment or self-referentiality is not considered there. Similarly, ccREL offers an RDF representation for Creative Commons licenses but uses a static name-based encoding that cannot capture the content-based relationships that we model [1]. Using rules in the policy language AIR [18], the meaning of ccREL terms has been further formalised but without attempting to overcome the restrictions of name-based modelling [30].

Bonatti and Mogavero consider policy containment as a formal reasoning task, and restrict the *Protune* policy language so that this task is decidable [8]. Reasoning about policy conformance and containment also motivated earlier studies by the second author, where policies have been formalised as conjunctive queries [31]. Our present work can be viewed as a generalisation of this approach.

Other related works have focussed on different aspects of increasing the expressiveness of policy modelling. Ringelstein and Staab present the history-aware PAPEL policy language that can be processed by means of a translation to Datalog [27]. 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 [13].

Many knowledge representation formalisms have been proposed to accomplish non-classical semantics (e.g. fixed point semantics) and meta-modelling (as present in our expression of containment as an object-level predicate). However, both aspects are usually not integrated, or come with technical restrictions that do not suit our application.

Fixed point operators exist in a number of flavours. Most closely related to our setting are works on fixed point based evaluation of terminological cycles in description logic ontologies [5,25]. Later works have been based on the relationship to the μ-calculus, see [6, Section 5.6] for an overview of the related literature. As is typical for such constructions, the required monotonicity is ensured on a logical level by restricting negation. This is not possible in our scenario where we focus on the entailment of implications (policy containments). Another approach of defining preferred models where certain predicate extensions have been minimised/maximised is Circumscription [22]. This might provide an alternative way to define a semantics that can capture desired policy containments, but it is not clear if and how entailments could then be computed.

Meta-modelling is possible with first- and higher-order approaches (see, e.g., [24] for an OWL-related discussion) yet we are not aware of any approaches that provide the semantics we intend. Glimm et al. [12], e.g., show how some schema entailments of OWL 2 DL can be represented with ontological individuals and properties, but the classical semantics of OWL would not yield the desired policy containments.

For relational algebra, it has been proposed to store relation names as individuals, and to use an expansion operator to access the extensions of these relations [28]. This allows for queries that check relational containment, but based on a fixed database (closed world) rather than on all possible interpretations (open world) as in our case.

8 Conclusions and Future Work

To the best of our knowledge, we have presented the first formal language for modelling self-referential policies. A particular advantage of our approach is that it can be instantiated in more specific knowledge representation formalisms, such as rule or ontology languages, to take advantage of existing tools for automated reasoning.

This opens up a number of directions for practical studies and exploitations. Refined provenance models, better tool support, and best practices for publishing policies are still required. On the conceptual side it would also be interesting to ask if our CF-based syntactic restrictions could be further relaxed without giving up the positive properties of the semantics.

Acknowledgements. We would like to thank Piero Bonatti, Clemens Kupke and the anonymous reviewers for their comments. Markus Krötzsch is sponsored by EPSRC grant EP/F065841/1. Sebastian Speiser is sponsored by the EU FP7 grant 257641.

References

1. Abelson, H., Adida, B., Linksvayer, M., Yergler, N.: ccREL: The Creative Commons Rights Expression Language. Tech. rep., Creative Commons (2008), http://creativecommons.org/projects/ccREL
2. Abiteboul, S., Duschka, O.M.: Complexity of answering queries using materialized views. In: Proc. 17th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems (PODS 1998), pp. 254–263. ACM (1998)
3. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley (1994)
4. Andréka, H., van Benthem, J., Németi, I.: Back and forth between modal logic and classical logic. Logic Journal of the IGPL 3(5), 685–720 (1995)
5. Baader, F.: Terminological cycles in KL-ONE-based knowledge representation languages. In: 8th National Conf. on Artificial Intelligence (AAAI 1990), pp. 621–626. AAAI Press (1990)
6. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): The Description Logic Handbook, 2nd edn. Cambridge University Press (2007)
7. Bonatti, P.A., De Coi, J.L., Olmedilla, D., Sauro, L.: A rule-based trust negotiation system. IEEE Transactions on Knowledge and Data Engineering 22(11), 1507–1520 (2010)
8. Bonatti, P.A., Mogavero, F.: Comparing rule-based policies. In: 9th IEEE Int. Workshop on Policies for Distributed Systems and Networks (POLICY 2008), pp. 11–18 (2008)
9. Cheney, J., Gil, Y., Groth, P., Miles, S.: Requirements for Provenance on the Web. W3C Provenance Incubator Group (2010), http://www.w3.org/2005/Incubator/prov/wiki/User_Requirements
10. Dodds, L.: Rights statements on the Web of Data. Nodalities Magazine, 13–14 (2010)
11. Glimm, B., Krötzsch, M.: SPARQL beyond subgraph matching. In: Patel-Schneider, et al. [26], pp. 241–256
12. Glimm, B., Rudolph, S., Völker, J.: Integrated metamodeling and diagnosis in OWL 2. In: Patel-Schneider, et al. [26], pp. 257–272
13. Hanson, C., Berners-Lee, T., Kagal, L., Sussman, G.J., Weitzner, D.: Data-purpose algebra: Modeling data usage policies. In: 8th IEEE Int. Workshop on Policies for Distributed Systems and Networks (POLICY 2007), pp. 173–177 (2007)
14. Hitzler, P., Krötzsch, M., Rudolph, S.: Foundations of Semantic Web Technologies. Chapman & Hall/CRC (2009)

15. Horrocks, I., Sattler, U.: Decidability of $\mathcal{SHIQ}$ with complex role inclusion axioms. Artificial Intelligence 160(1), 79–104 (2004)
16. Iannella, R.: Open Digital Rights Language (ODRL) Version 1.1. W3C Note (September 19, 2002), http://www.w3.org/TR/odrl/
17. Kagal, L., Finin, T., Joshi, A.: A policy language for a pervasive computing environment. In: 4th IEEE Int. Workshop on Policies for Distributed Systems and Networks (POLICY 2003), pp. 63–74 (2003)
18. Kagal, L., Hanson, C., Weitzner, D.: Using dependency tracking to provide explanations for policy management. In: 9th IEEE Int. Workshop on Policies for Distributed Systems and Networks (POLICY 2008), pp. 54–61 (2008)
19. Kephart, J.O., Walsh, W.E.: An artificial intelligence perspective on autonomic computing policies. In: 5th IEEE Int. Workshop on Policies for Distributed Systems and Networks (POLICY 2004), pp. 3–12 (2004)
20. Krötzsch, M., Speiser, S.: Expressing self-referential usage policies for the Semantic Web. Tech. Rep. 3014, Institute AIFB, Karlsruhe Institute of Technology (2011), http://www.aifb.kit.edu/web/Techreport3014
21. Lessig, L.: CC in Review: Lawrence Lessig on Compatibility (2005), http://creativecommons.org/weblog/entry/5709 (accessed July 1, 2011)
22. Lifshitz, V.: Circumscriptive theories: A logic-based framework for knowledge representation. Journal of Philosophical Logic 17, 391–441 (1988)
23. Moreau, L., Clifford, B., Freire, J., Futrelle, J., Gil, Y., Groth, P., Kwasnikowska, N., Miles, S., Missier, P., Myers, J., Plale, B., Simmhan, Y., Stephan, E., Van den Bussche, J.: The Open Provenance Model core specification (v1.1). Future Generation Computer Systems 27, 743–756 (2011)
24. Motik, B.: On the properties of metamodeling in OWL. J. of Logic and Computation 17(4), 617–637 (2007)
25. Nebel, B.: Terminological cycles: Semantics and computational properties. In: Sowa, J.F. (ed.) Principles of Semantic Networks: Explorations in the Representation of Knowledge, pp. 331–361. Kaufmann (1991)
26. Patel-Schneider, P.F., Pan, Y., Glimm, B., Hitzler, P., Mika, P., Pan, J., Horrocks, I.: ISWC 2010, Part I. LNCS, vol. 6496. Springer, Heidelberg (2010)
27. Ringelstein, C., Staab, S.: PAPEL: A Language and Model for Provenance-Aware Policy Definition and Execution. In: Hull, R., Mendling, J., Tai, S. (eds.) BPM 2010. LNCS, vol. 6336, pp. 195–210. Springer, Heidelberg (2010)
28. Ross, K.A.: Relations with relation names as arguments: algebra and calculus. In: Proc. 11th ACM SIGACT-SIGMOD-SIGART Symp. on Principles of Database Systems (PODS 1992), pp. 346–353. ACM (1992)
29. Seneviratne, O., Kagal, L., Berners-Lee, T.: Policy-Aware Content Reuse on the Web. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 553–568. Springer, Heidelberg (2009)
30. Seneviratne, O.W.: Framework for Policy Aware Reuse of Content on the WWW. Master thesis. Massachusetts Institute of Technology (2009)
31. Speiser, S., Studer, R.: A self-policing policy language. In: Patel-Schneider, et al. [26], pp. 730–746
32. W3C OWL Working Group: OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (October 27, 2009), http://www.w3.org/TR/owl2-overview/
33. Weitzner, D.J., Hendler, J., Berners-Lee, T., Connolly, D.: Creating a policy-aware Web: Discretionary, rule-based access for the World Wide Web. In: Web and Information Security, ch. I, pp. 1–31. IRM Press (2006)