

Expressing Self-Referential Usage Policies for the Semantic Web[★]

Technical Report 3014
Institute AIFB, Karlsruhe Institute of Technology, DE

Markus Krötzsch¹ and Sebastian Speiser²

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 Commons³ 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 [30]. 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

[★] Results of this technical report have been published as *Markus Krötzsch, Sebastian Speiser: ShareAlike Your Data: Self-Referential Usage Policies for the Semantic Web. Proceedings of the 10th International Semantic Web Conference (ISWC'10). Springer, 2010.*

³ <http://creativecommons.org/>

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 [34,10]. 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].

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 the Creative Commons 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 and study a formal policy semantics in Section 5 and apply it to obtain concrete policy modelling languages in Section 6 based on OWL and Datalog. Related work is discussed in Section 7.

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 [...]”⁴

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 [22]. 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.

⁴ Section 4(b) in <http://creativecommons.org/licenses/by-nc-sa/3.0/legalcode>

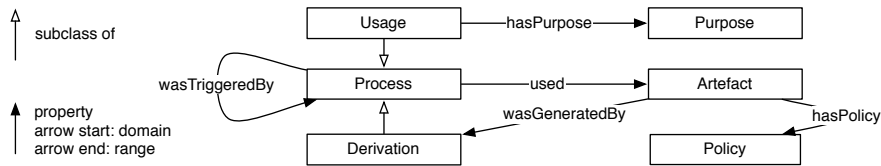


Fig. 1. Informal view of a simple provenance model

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 [24] below we prefer to speak of admissible “processes” as the most general type of use. The admissible processes can be viewed as “desired states” (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 *derivations* (processes that generate a new artefact) and other *usages* 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 .⁵ For example, a policy p that allows no uses other than derivations that generate artefacts with policy p can be described as:

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

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

$$p : \text{Derivation}(x) \wedge \exists y.(\text{wasGeneratedBy}(y, x) \wedge \exists z.(\text{hasPolicy}(y, z) \wedge \text{containedIn}(z, p))). \quad (2)$$

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:

$$\forall x.\text{conformsTo}(x, p) \leftrightarrow \varphi_p[x] \quad \text{for all } p \in N_p, \quad (3)$$

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

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 `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 `containedIn`(p, q) does not hold,

⁵ We assume basic familiarity with the standard syntax and semantics of first-order logic. Formal definitions can be found in Appendix A.

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 $\text{containedIn}(p, q)$ nor $\neg\text{containedIn}(p, q)$.

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 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 $\text{containedIn}(p, q)$ 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:

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

Though the policy q is clearly more restrictive than p , there still is a first-order interpretation that satisfies $\text{containedIn}(p, q)$ 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):

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

This policy system entails $\text{containedIn}(p, q)$. Indeed, if $\text{containedIn}(p, q)$ would not hold, then nothing would conform to p by (3). But the empty set is clearly a subset of every other set, hence $\text{containedIn}(p, q)$ would follow by (4). Thus all interpretations that satisfy T_{ct} must satisfy $\forall x.A(x) \wedge \text{containedIn}(p, q) \rightarrow B(x)$, 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 $\varphi[\mathbf{x}]$ to indicate that φ has at most the free variables that occur in \mathbf{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 $p(\mathbf{t})$ with \mathbf{t} a vector of terms that contain at least one variable belongs to COF, provided that \mathbf{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 $\varphi[\mathbf{x}, \mathbf{y}]$ in COF, and a conjunction $\alpha[\mathbf{x}, \mathbf{y}] = \alpha_1[\mathbf{x}, \mathbf{y}] \wedge \dots \wedge \alpha_n[\mathbf{x}, \mathbf{y}]$ of atomic formulae α_i that contain only guard predicates and variables, such that \mathbf{x}, \mathbf{y} are both non-empty and do not share variables. Then the formulae*

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

are in COF provided that for each variable y in \mathbf{y} , there is some variable x in \mathbf{x} and some atom $\alpha_i[\mathbf{x}, \mathbf{y}]$ 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 `containedIn`(y, p) or `containedIn`(p, y),
- every occurrence of `containedIn` in φ is positive (i.e. not in the scope of a negation) and has the form `containedIn`(y, p) or `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_{\perp} , and a policy system P for N_P such that $\langle p_{\top}, \top(x) \rangle, \langle p_{\perp}, \perp(x) \rangle \in P$.⁶ Let T_{ci} be the following theory:

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

For a set $C \subseteq N_P^2$, define $\text{Cl}(C) := \{ \text{containedIn}(p, q) \mid \langle p, q \rangle \in C \}$. 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:

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

⁶ As usual, we consider \top/\perp as unary predicates that are true/false for all individuals.

Proposition 1. *The operator P_T has a greatest fixed point $\text{gfp}(P_T)$ 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 .*

Proof. Monotonicity of P_T is an easy consequence of monotonicity of first-order logic. It is well known that every monotone function on a complete lattice (in our case the finite lattice of subsets of N_p^2) has a greatest fixed point that can be obtained as claimed. \square

The fact that P_T requires the existence of policies p_\top and p_\perp 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:*

$$\mathcal{I} \models T \cup T_{\text{ci}} \cup \text{Cl}(\text{gfp}(P_T)) \cup T_{\text{ct}}, \quad (7)$$

where T_{ci} and $\text{Cl}(\text{gfp}(P_T))$ 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:*

$$T, T_{\text{ci}}, \text{Cl}(\text{gfp}(P_T)), T_{\text{ct}}^- \models \varphi \quad \text{iff} \quad T \models_P \varphi, \quad (8)$$

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

Before presenting a formal proof for this result, let us first discuss how Theorem 1 answers the above questions.

- (1) The theorem reduces P -entailment to standard first-order logic entailment. Since $\text{gfp}(P_T)$ 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 $T \cup T_{\text{ci}} \cup \text{Cl}(\text{gfp}(P_T))$. It is easy to see that T_{ci} and $\text{Cl}(\text{gfp}(P_T))$ 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.

Theorem 1 is a consequence of Proposition 2 below. To establish this, however, we first require a number of auxiliary results that we develop next.

Definition 6. Consider two interpretations I_1 and I_2 such that, for all constants $a \neq b$, we have $a^{I_i} \neq b^{I_i}$ for $i = 1, 2$.⁷ Assume without loss of generality that the domains of I_1 and I_2 are disjoint. The interpretation $I_1 \searrow I_2$ is defined as follows:

- the domain $\Delta^{I_1 \searrow I_2}$ is the (disjoint) union $\Delta^{I_1} \cup \Delta^{I_2}$,
- for all constants c , set $c^{I_1 \searrow I_2} := c^{I_1}$,
- for all guard predicates p , set $p^{I_1 \searrow I_2} := p^{I_1} \cup p^{I_2}$,
- for all non-guard predicates p , set $p^{I_1 \searrow I_2} := p^{I_1} \cup \{\delta' \mid \delta \in p^{I_2}, \delta \rightsquigarrow \delta'\}$, where we write $\delta \rightsquigarrow \delta'$ to indicate that δ' can be obtained from δ by replacing zero or more elements of the form $c^{I_2} \in \delta$ for some constant c by the element c^{I_1} , as long as δ' still contains at least one element of Δ^{I_2} .

Intuitively speaking, we construct $I_1 \searrow I_2$ as a disjoint union of the two interpretations. However, constants can only be interpreted in one way, and we use I_1 for this, which makes the operation non-commutative. To ensure that enough (for our purposes) facts remain true for the I_2 part of $I_1 \searrow I_2$, we introduce the relation \rightsquigarrow : in addition to the (disjoint) extension of p^{I_2} , it allows us to add to $p^{I_1 \searrow I_2}$ all tuples in which a named element of Δ^{I_2} is replaced by the corresponding named element of Δ^{I_1} . For technical reasons, we have to be careful that the tuple still contains at least one element of Δ^{I_2} . This restriction is suitable for our CF formulae in the sense that the operation preserves all entailments of such formulae. This is established in the following two lemmas.

Lemma 1. Consider a COF formula φ , and interpretations I_1 and I_2 as in Definition 6. For any variable assignment \mathcal{Z} for I_i ($i = 1, 2$) we find

$$I_i, \mathcal{Z} \models \varphi \quad \text{iff} \quad I_1 \searrow I_2, \mathcal{Z} \models \varphi. \quad (9)$$

⁷ This *Unique Name Assumption* (UNA) is no restriction in our case, since we do not consider equality in our logic. Therefore we will always silently assume that we have models where the UNA holds when applying the operation of Definition 6.

Proof. We show the claim by induction over the structure of COF formulae as in Definition 1. The base case is $\varphi = p(\mathbf{t})$. The claim is immediate if p is a guard predicate. For the case that p is a non-guard predicate, first note that every tuple $\delta \in p^{I_1 \searrow I_2}$ is either from p^{I_1} , or contains at least one element of Δ^{I_2} (*). For the case $i = 1$, the “only if” direction is immediate, and the “if” direction follows from (*). Now consider $i = 2$ and assume $I_2, \mathcal{Z} \models p(\mathbf{t})$ (“only if”). Let $\delta := \mathbf{t}^{I_2, \mathcal{Z}}$ and $\delta' := \mathbf{t}^{I_1 \searrow I_2, \mathcal{Z}}$. Clearly $\delta \rightsquigarrow \delta'$ since \mathbf{t} contains at least one variable. Thus $\delta' \in p^{I_1 \searrow I_2}$ as required. For the “if” direction, assume $\delta' \in p^{I_1 \searrow I_2}$. Hence there is some $\delta'' \in p^{I_2}$ with $\delta'' \rightsquigarrow \delta'$. As every named individual c^{I_1} corresponds to exactly one unique named individual c^{I_2} , we find $\delta'' = \mathbf{t}^{I_2, \mathcal{Z}}$ as required.

For the cases where φ is of the form $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$, $\varphi_1 \vee \varphi_2$, or $\varphi_1 \rightarrow \varphi_2$, the claim follows immediately by induction.

Now consider $\varphi = \exists \mathbf{y}. \alpha[\mathbf{x}, \mathbf{y}] \wedge \varphi'[\mathbf{x}, \mathbf{y}]$ with $\alpha[\mathbf{x}, \mathbf{y}] = \alpha_1[\mathbf{x}, \mathbf{y}] \wedge \dots \wedge \alpha_n[\mathbf{x}, \mathbf{y}]$. If $I_i, \mathcal{Z} \models \varphi$ then there is a variable assignment \mathcal{Z}' for I_i that agrees with \mathcal{Z} on all variables but possibly \mathbf{y} , and such that $I_i, \mathcal{Z}' \models \varphi'$ and $I_i, \mathcal{Z}' \models \alpha_j[\mathbf{x}, \mathbf{y}]$ for $j = 1, \dots, n$. By the hypothesis, $I_1 \searrow I_2, \mathcal{Z}' \models \varphi'$ and $I_i, \mathcal{Z}' \models \alpha_j[\mathbf{x}, \mathbf{y}]$ for $j = 1, \dots, n$. Thus $I_1 \searrow I_2, \mathcal{Z} \models \varphi$.

For the other direction, assume that $I_1 \searrow I_2, \mathcal{Z} \models \varphi$. Again there is an assignment \mathcal{Z}' for $I_1 \searrow I_2$ that agrees with \mathcal{Z} on all variables but possibly \mathbf{y} , and such that $I_1 \searrow I_2, \mathcal{Z}' \models \varphi'$ and $I_1 \searrow I_2, \mathcal{Z}' \models \alpha_j[\mathbf{x}, \mathbf{y}]$ for $j = 1, \dots, n$.

Every variable y in \mathbf{y} occurs in some $\alpha_j[\mathbf{x}, \mathbf{y}]$ of the form $\alpha_j[\mathbf{x}, \mathbf{y}] = q(\mathbf{z})$ together with some variable x from \mathbf{x} , where q is a guard predicate. Thus $\mathcal{Z}(x) = \mathcal{Z}'(x) \in \Delta^{I_i}$, so by Definition 6 $\mathbf{z}^{\mathcal{Z}'} \in q^{I_i}$. Therefore $\mathcal{Z}'(\mathbf{y}) \in \Delta^{I_i}$. Since this holds for all y in \mathbf{y} , \mathcal{Z}' is an assignment for I_i , and we find $I_i, \mathcal{Z}' \models \varphi'$ and $I_i, \mathcal{Z}' \models \alpha_j[\mathbf{x}, \mathbf{y}]$ for $j = 1, \dots, n$. Thus $I_i, \mathcal{Z} \models \varphi$ as required.

The case for $\varphi = \forall \mathbf{y}. \alpha[\mathbf{x}, \mathbf{y}] \rightarrow \varphi'[\mathbf{x}, \mathbf{y}]$ is analogous. □

Lemma 2. Consider interpretations I_1 and I_2 as in Definition 6. For every CF formula φ :

- (1) if φ contains no variables then $I_1 \searrow I_2 \models \varphi$ iff $I_1 \models \varphi$,
- (2) if $\varphi = \exists x. \psi[x]$ then $I_1 \searrow I_2 \models \varphi$ iff either $I_1 \models \varphi$ or $I_2 \models \varphi$,
- (3) if $\varphi = \forall x. \psi[x]$ then $I_1 \searrow I_2 \models \varphi$ iff both $I_1 \models \varphi$ and $I_2 \models \varphi$.

Proof. Case (1) is obvious for variable-free atomic formulae by the construction in Definition 6. The claim for non-atomic variable-free formulae follows by an easy induction. Cases (2) and (3) are an easy consequence of Lemma 1. □

The construction of Definition 6 can be turned into an infinitary operation as follows. Given models I_1 and I_2 as in the definition, define $I_1 \searrow_1 I_2 := I_1 \searrow I_2$, and $I_1 \searrow_{i+1} I_2 := (I_1 \searrow_i I_2) \searrow I_2'$ where I_2' is the same as I_2 but with every domain element renamed to ensure the required disjointness of domains. Now we can define $I_1 \searrow_\omega I_2 := \bigcup_{i \geq 1} I_1 \searrow_i I_2$, where the union of interpretations with the same interpretation of constants is obtained by taking the union of domains and predicate extensions. It is easy to see that Lemmas 1 and 2 can also be adapted to the infinitary case.

The theory T_{ci} of Definition 4 is not in CF, so Lemma 2 does not ensure that its satisfiability is preserved. To achieve the latter, it is useful to modify the extension of `containedIn` as in the following lemma.

Lemma 3. *Consider an interpretation \mathcal{I} such that $\mathcal{I} \models T \cup T_{\text{ci}} \cup \text{Cl}(\text{gfp}(P_T)) \cup T_{\text{ct}}^-$, and let $p \in N_P$ be a policy name. Define an interpretation $\mathcal{I} \Downarrow p$ with domain $\Delta^{\mathcal{I} \Downarrow p} := \Delta^{\mathcal{I}}$ as follows:*

- $c^{\mathcal{I} \Downarrow p} := c^{\mathcal{I}}$ for all constants c ,
- $q^{\mathcal{I} \Downarrow p} := q^{\mathcal{I}}$ for all predicates $q \notin \{\text{containedIn}, \text{conformsTo}\}$,
- `containedIn` ^{$\mathcal{I} \Downarrow p$} is the reflexive transitive closure of $\{\langle \delta_1, \delta_2 \rangle \in \text{containedIn}^{\mathcal{I}} \mid \delta_1 = p^{\mathcal{I}} \text{ or } \delta_2 = p^{\mathcal{I}}\} \cup \{\langle q_1^{\mathcal{I}}, q_2^{\mathcal{I}} \rangle \mid \langle q_1, q_2 \rangle \in \text{gfp}(P_T)\} \cup \{\langle p_{\perp}^{\mathcal{I}}, \epsilon \rangle, \langle \epsilon, p_{\top}^{\mathcal{I}} \rangle \mid \epsilon \in \Delta^{\mathcal{I}}\}$,
- for all $p \in N_P$ and $\delta \in \Delta^{\mathcal{I}}$, define $\langle \delta, p^{\mathcal{I}} \rangle \in \text{conformsTo}^{\mathcal{I} \Downarrow p}$ if and only if $\mathcal{I} \Downarrow p, \{x \mapsto \delta\} \models \varphi_p[x]$ where $\varphi_p[x]$ is the policy condition for p .

Then $\mathcal{I} \Downarrow p \models T \cup T_{\text{ci}} \cup \text{Cl}(\text{gfp}(P_T)) \cup T_{\text{ct}}^-$

Moreover, for policy conditions φ_p and ψ where φ_p contains (only) the policy name p , we have that $\mathcal{I} \Downarrow p \models \forall x. \varphi_p[x] \rightarrow \psi[x]$ implies $\mathcal{I} \models \forall x. \varphi_p[x] \rightarrow \psi[x]$. Likewise, for every φ that is a CF formula over the base signature, or a variable-free atom (fact) over the base signature together with the predicates `containedIn` or `conformsTo`, we have that $\mathcal{I} \Downarrow p \models \varphi$ implies $\mathcal{I} \models \varphi$.

Proof. We first show $\mathcal{I} \Downarrow p \models T \cup T_{\text{ci}} \cup \text{Cl}(\text{gfp}(P_T))$. Since $\mathcal{I} \Downarrow p$ agrees with \mathcal{I} on the interpretation of all symbols other than `containedIn` and `conformsTo`, and since these predicates do not occur in T , we have $\mathcal{I} \Downarrow p \models T$. The remaining claim $\mathcal{I} \Downarrow p \models T_{\text{ci}} \cup \text{Cl}(\text{gfp}(P_T)) \cup T_{\text{ct}}^-$ follows directly from our construction.

For the second part of the claim, observe that $\langle \delta, p^{\mathcal{I} \Downarrow p} \rangle \in \text{containedIn}^{\mathcal{I} \Downarrow p}$ if and only if $\langle \delta, p^{\mathcal{I}} \rangle \in \text{containedIn}^{\mathcal{I}}$ by definition. Similarly, $\langle p^{\mathcal{I} \Downarrow p}, \delta \rangle \in \text{containedIn}^{\mathcal{I} \Downarrow p}$ if and only if $\langle p^{\mathcal{I}}, \delta \rangle \in \text{containedIn}^{\mathcal{I}}$. Thus $\mathcal{I} \Downarrow p, \{x \mapsto \delta\} \models \varphi_p[x]$ iff $\mathcal{I}, \{x \mapsto \delta\} \models \varphi_p[x]$ (*).

Furthermore, $\text{containedIn}^{\mathcal{I} \Downarrow p} \subseteq \text{containedIn}^{\mathcal{I}}$ is an easy consequence of $\mathcal{I} \models T_{\text{ci}} \cup \text{Cl}(\text{gfp}(P_T))$. Since `containedIn` occurs only positively in policy conditions, $\mathcal{I} \Downarrow p, \{x \mapsto \delta\} \models \psi[x]$ implies $\mathcal{I}, \{x \mapsto \delta\} \models \psi[x]$. Together with (*) this shows the claim.

The case for arbitrary CF sentences or facts φ is shown similarly, using that the predicates `containedIn` or `conformsTo` can only occur positively in φ . \square

From an arbitrary model $\mathcal{I} \models T$ of a background theory T , it is easy to construct a model $\mathcal{I}' \models T \cup T_{\text{ci}} \cup \text{Cl}(\text{gfp}(P_T)) \cup T_{\text{ct}}^-$ since $T_{\text{ci}} \cup \text{Cl}(\text{gfp}(P_T)) \cup T_{\text{ct}}^-$ can be satisfied by suitable interpretation of `containedIn` and `conformsTo` without affecting the interpretation of T . Finding a P -model of T , however, is more difficult, since it must satisfy T_{ct} as well. In the model \mathcal{I}' , especially the implication $\forall x, y. \text{containedIn}(x, y) \leftarrow \forall z. (\text{conformsTo}(z, x) \rightarrow \text{conformsTo}(z, y))$ may fail (its converse is relatively easy to ensure). To obtain a P -model, we (1) modify the extension of `containedIn` so that only facts in $\text{Cl}(\text{gfp}(P_T))$ hold, and (2) extend the model so that no other containments follow from T_{ct} . This is possible with the operation from Definition 6. An additional technical difficulty is that we need to define the set of conforming elements for all (even unnamed) policies in a way that is consistent with T_{ct} . To overcome this, we introduce an infinite number of copies of interpretations, so that we can conveniently construct extensions that do not entail unwanted containments.

Proposition 2. Consider a theory T and a policy system P . For every model of \mathcal{I} of $T \cup T_{\text{ci}} \cup \text{Cl}(\text{gfp}(P_T)) \cup T_{\text{ct}}^-$, there is a P -model $\hat{\mathcal{I}}$ of T such that, for every φ that is a variable-free atom (fact) over the predicates containedIn or conformsTo we have that $\hat{\mathcal{I}} \models \varphi$ implies $\mathcal{I} \models \varphi$.

Proof. We first describe the construction of $\hat{\mathcal{I}}$ from \mathcal{I} , and then show that it has the required properties.

For every $\langle p, q \rangle \notin \text{gfp}(P_T)$, select an interpretation $\mathcal{I}_{\langle p, q \rangle}$ with $\mathcal{I}_{\langle p, q \rangle} \not\models \forall x. \varphi_p[x] \rightarrow \varphi_q[x]$ and $\mathcal{I}_{\langle p, q \rangle} \models T \cup T_{\text{ci}} \cup \text{Cl}(\text{gfp}(P_T))$. Such an interpretation exists due to the construction of P_T . Now define $\mathcal{I}'_{\langle p, q \rangle} := \mathcal{I}_{\langle p, q \rangle} \uparrow p$. Moreover, enumerate the set $(\mathcal{I}'_{\langle p, q \rangle})_{\langle p, q \rangle \notin \text{gfp}(P_T)}$ in arbitrary order to obtain a sequence $\mathcal{I}'_1, \dots, \mathcal{I}'_l$. Let $\mathcal{I}' := \mathcal{I} \uparrow p$ for an arbitrary policy name $p \in N_P$.

Then construct a model $\mathcal{J} := (\dots ((\mathcal{I}' \searrow_{\omega} \mathcal{I}') \searrow_{\omega} \mathcal{I}'_1) \dots \searrow_{\omega} \mathcal{I}'_l)$. We assume w.l.o.g. that the domains of $\mathcal{I}', \mathcal{I}'_1, \dots, \mathcal{I}'_l$ are disjoint, so we can denote $\Delta^{\mathcal{J}}$ as follows:

$$\Delta^{\mathcal{J}} = \{\delta_i \mid \delta \in \Delta^{\mathcal{I}'}, i \geq 1\} \cup \{\delta_i \mid \delta \in \Delta^{\mathcal{I}'_1}, i \geq 1\} \cup \dots \cup \{\delta_i \mid \delta \in \Delta^{\mathcal{I}'_l}, i \geq 1\}. \quad (10)$$

Define $\hat{\mathcal{I}}$ to coincide with \mathcal{J} on all aspects save the interpretation of containedIn and conformsTo, i.e. $\Delta^{\hat{\mathcal{I}}} := \Delta^{\mathcal{J}}$ for the domain, $c^{\hat{\mathcal{I}}} := c^{\mathcal{J}} (= c^{\mathcal{I}'} = c^{\mathcal{I}'})$ for all constants c , and $p^{\hat{\mathcal{I}}} := p^{\mathcal{J}}$ for all predicates $p \notin \{\text{containedIn}, \text{conformsTo}\}$. Define containedIn $^{\hat{\mathcal{I}}}$ as the reflexive transitive closure of

$$\text{containedIn}^{\mathcal{J}} \cup \{\langle p_{\perp}^{\mathcal{I}}, \epsilon \rangle, \langle \epsilon, p_{\top}^{\mathcal{I}} \rangle \mid \epsilon \in \Delta^{\hat{\mathcal{I}}}\}. \quad (11)$$

To understand the structure of containedIn $^{\hat{\mathcal{I}}}$, we define the following relation \mathcal{R} using the notation (10) for domain elements:

$$\begin{aligned} \mathcal{R} := & \{\langle p^{\mathcal{I}}, q^{\mathcal{I}} \rangle \mid \langle p, q \rangle \in \text{gfp}(P_T)\} \cup \\ & \{\langle \delta_i, p^{\mathcal{I}} \rangle \mid \langle \delta, p^{\mathcal{I}} \rangle \in \text{containedIn}^{\mathcal{I}}, \mathcal{I}' = \mathcal{I} \uparrow p, i \geq 1\} \cup \\ & \{\langle p^{\mathcal{I}}, \epsilon_i \rangle \mid \langle p^{\mathcal{I}}, \epsilon \rangle \in \text{containedIn}^{\mathcal{I}}, \mathcal{I}' = \mathcal{I} \uparrow p, i \geq 1\} \cup \\ & \bigcup_{\langle p, q \rangle \notin \text{gfp}(P_T)} \{\langle \delta_i, p^{\mathcal{I}} \rangle \mid \langle \delta, p^{\mathcal{I}_{\langle p, q \rangle}} \rangle \in \text{containedIn}^{\mathcal{I}_{\langle p, q \rangle}}, i \geq 1\} \cup \\ & \{\langle p^{\mathcal{I}}, \epsilon_i \rangle \mid \langle p^{\mathcal{I}_{\langle p, q \rangle}}, \epsilon \rangle \in \text{containedIn}^{\mathcal{I}_{\langle p, q \rangle}}, i \geq 1\}. \end{aligned} \quad (12)$$

It is not hard to see that containedIn $^{\hat{\mathcal{I}}}$ is the reflexive transitive closure of the relation $\mathcal{R} \cup \{\langle p_{\perp}^{\mathcal{I}}, \epsilon \rangle, \langle \epsilon, p_{\top}^{\mathcal{I}} \rangle \mid \epsilon \in \Delta^{\hat{\mathcal{I}}}\}$. The advantage of \mathcal{R} over containedIn $^{\mathcal{J}}$ above is that \mathcal{R} contains less tuples that are redundant due to completion with respect to p_{\perp}/p_{\top} and the transitive closure.

In particular, we can distinguish the following cases for any element $\eta \in \Delta^{\hat{\mathcal{I}}}$:

- (A) $\eta = p^{\mathcal{I}}$ for some $p \in N_P$, or
 η is not of this form, but there is $p \in N_P$ such that:
- (B) $\langle \eta, p^{\mathcal{I}} \rangle \in \mathcal{R}$ and $\langle p^{\mathcal{I}}, \eta \rangle \in \mathcal{R}$, or
- (C) $\langle \eta, p^{\mathcal{I}} \rangle \in \mathcal{R}$ and $\langle p^{\mathcal{I}}, \eta \rangle \notin \mathcal{R}$, or
- (D) $\langle \eta, p^{\mathcal{I}} \rangle \notin \mathcal{R}$ and $\langle p^{\mathcal{I}}, \eta \rangle \in \mathcal{R}$, or
- (E) η occurs in no tuple of \mathcal{R} ,

where in cases (B), (C), (D) the given tuples are the only occurrences of η in \mathcal{R} . Thus, other than in case (A), η can be related to at most one element of the form $p^{\mathcal{I}}$, and this domain element is unique. This case distinction partitions the domain, and we will speak of *A-elements*, *B-elements*, and so on to refer to individuals η to which the according case applies.

It remains to define $\text{conformsTo}^{\hat{\mathcal{I}}}$. For this let $\mu : \mathcal{A}^{\hat{\mathcal{I}}} \rightarrow \mathbb{N}$ be an injective function that assigns a natural number $\mu(\epsilon) \geq 1$ to each domain element $\epsilon \in \mathcal{A}^{\hat{\mathcal{I}}}$. Such a function exists since we can assume w.l.o.g. that $\mathcal{A}^{\hat{\mathcal{I}}}$ is countable (it is certainly infinite by construction). We now define $\text{conformsTo}^{\hat{\mathcal{I}}}$ by defining for each $\eta \in \mathcal{A}^{\hat{\mathcal{I}}}$ the set $S(\eta) = \{\epsilon \in \mathcal{A}^{\hat{\mathcal{I}}} \mid \langle \epsilon, \eta \rangle \in \text{conformsTo}^{\hat{\mathcal{I}}}\}$. We distinguish the above cases and use notation (10) for domain elements δ_j :

- (A) $S(p^{\mathcal{I}}) := \{\delta_j \in \mathcal{A}^{\hat{\mathcal{I}}} \mid \hat{\mathcal{I}}, \{x \mapsto \delta_j\} \models \varphi_p[x]\}$ where $\varphi_p[x]$ is the policy condition for p ;
- (B) $S(\eta) := S(p^{\mathcal{I}})$;
- (C) $S(\eta) := \{\delta_j \in \mathcal{A}^{\hat{\mathcal{I}}} \mid \delta_j \in S(p^{\mathcal{I}}) \text{ and } j \neq \mu(\eta)\}$;
- (D) $S(\eta) := \{\delta_j \in \mathcal{A}^{\hat{\mathcal{I}}} \mid \delta_j \in S(p^{\mathcal{I}}) \text{ or } j = \mu(\eta)\}$;
- (E) $S(\eta) := \{\delta_j \in \mathcal{A}^{\hat{\mathcal{I}}} \mid j = \mu(\eta)\}$.

This finishes the construction of $\hat{\mathcal{I}}$. We now show that $\hat{\mathcal{I}}$ is a P -model of T . For all $\mathcal{I}_{\langle p, q \rangle}$, we have $\mathcal{I}_{\langle p, q \rangle} \models T \cup T_{\text{ci}} \cup \text{Cl}(\text{gfp}(P_T))$, and thus $\mathcal{I}'_{\langle p, q \rangle} \models T$ by Lemma 3. Similarly $\mathcal{I}' \models T$, so $\mathcal{J} \models T$ by Lemma 2. Thus $\hat{\mathcal{I}} \models T$ since it agrees with \mathcal{J} on all symbols in T . Moreover, $\hat{\mathcal{I}} \models T_{\text{ci}}$ is immediate from our definition of $\text{containedIn}^{\hat{\mathcal{I}}}$, and $\hat{\mathcal{I}} \models \text{Cl}(\text{gfp}(P_T))$ follows from $\mathcal{I}' \models \text{Cl}(\text{gfp}(P_T))$.

It remains to show that $\hat{\mathcal{I}} \models T_{\text{ct}}$. That (3) is satisfied by $\hat{\mathcal{I}}$ is immediate from our definition. To show that (4) is satisfied as well, we show that, for all $\eta, \eta' \in \mathcal{A}^{\hat{\mathcal{I}}}$, we have $\langle \eta, \eta' \rangle \in \text{containedIn}^{\hat{\mathcal{I}}}$ iff $S(\eta) \subseteq S(\eta')$ (\ddagger). We use a number of auxiliary claims to show this.

(Claim 1) First note that the elements that conform to named policies in $\hat{\mathcal{I}}$ are obtained as the disjoint union of the conforming elements in the individual interpretations; formally:

$$S(p^{\mathcal{I}}) = \{\delta_i \mid \langle \delta, p^{\mathcal{I}} \rangle \in \text{conformsTo}^{\mathcal{I}}, i \geq 1\} \cup \bigcup_{j=1}^l \{\delta_i \mid \langle \delta, p^{\mathcal{I}'_j} \rangle \in \text{conformsTo}^{\mathcal{I}'_j}, i \geq 1\}.$$

This follows from Lemma 1.

We first show the “only if” direction of (\ddagger). Since \subseteq is reflexive and transitive, and since $S(p_{\perp}) = \emptyset$ and $S(p_{\perp}) = \mathcal{A}^{\hat{\mathcal{I}}}$, it suffices to show that $\langle \eta, \eta' \rangle \in \mathcal{R}$ implies $S(\eta) \subseteq S(\eta')$. First consider the case that η, η' are of form $\eta = p^{\mathcal{I}}$, $\eta' = q^{\mathcal{I}'}$. Then $S(p^{\mathcal{I}}) \subseteq S(q^{\mathcal{I}'})$ follows from Claim 1 above, since all interpretations $\mathcal{I}', \mathcal{I}'_1, \dots, \mathcal{I}'_l$ satisfy the corresponding containment. The latter is a requirement for $\langle p, q \rangle \in \text{gfp}(P_T)$.

For the case that either η or η' are not of the form $p^{\mathcal{I}}$ for any $p \in N_P$, the cases (B)–(E) of our definition apply. In particular, the \mathcal{R} relations that are explicitly mentioned

in these cases are the only ones for which the claim must be shown. We readily verify: (B) $S(\eta) = S(p^I)$, (C) $S(\eta) \subseteq S(p^I)$, (D) $S(\eta') \supseteq S(p^I)$. Case (E) does not impose any requirements. This finishes the proof of the “only if” direction of (\ddagger) .

(Claim 2) If $S(p^I) \subseteq S(q^I)$ then $\langle p, q \rangle \in \text{gfp}(P_T)$. We show the contrapositive. Assume that $\langle p, q \rangle \notin \text{gfp}(P_T)$. Then we selected an interpretation $\mathcal{I}_{\langle p, q \rangle}$ with $\mathcal{I}_{\langle p, q \rangle} \not\models \forall x. \varphi_p[x] \rightarrow \varphi_q[x]$. By Lemma 3, $\mathcal{I}'_{\langle p, q \rangle} \not\models \forall x. \varphi_p[x] \rightarrow \varphi_q[x]$. Using Claim 1, we find that $S(p^I) \not\subseteq S(q^I)$. This establishes Claim 2.

It remains to show the “if” direction of (\ddagger) . First note that, for every policy $p \in N_P$, we have $\delta_j \in S(p^I)$ iff, for all $k \geq 1$, $\delta_k \in S(p^I)$ (Claim 3). This is a direct consequence of Claim 1. Now assume $S(\eta) \subseteq S(\eta')$. We distinguish cases as follows:

- (a) η is an A-element $\eta = p^I$ for policy p .
 - (a.a) η' is an A-element $\eta' = q^I$. Then $\langle p^I, q^I \rangle \in \text{containedIn}^{\hat{I}}$ follows from Claim 2.
 - (a.b) η' is a B-element for policy q . Then according to (a.a) $S(\eta) \subseteq S(q^I)$ implies $\langle p^I, q^I \rangle \in \text{containedIn}^{\hat{I}}$. From the assumption $\langle q^I, \eta' \rangle \in \text{containedIn}^{\hat{I}}$ we obtain $\langle p^I, \eta' \rangle \in \text{containedIn}^{\hat{I}}$.
 - (a.c) η' is a C-element for policy q . We first show $S(\eta) = \emptyset$. Indeed, $S(\eta')$ does not include any element of the form δ_j for some number $j = \mu(\eta')$. Claim 3 and $S(\eta) \subseteq S(\eta')$ together imply $S(\eta) = \emptyset$. By Claim 1, $S(\eta) = \emptyset$ implies that $\langle p, p_{\perp} \rangle \in \text{gfp}(P_T)$. This implies $\langle p^I, \eta' \rangle \in \text{containedIn}^{\hat{I}}$ as required.
 - (a.d) η' is a D-element for policy q . We first show $S(p^I) \subseteq S(q^I)$. For a contradiction, suppose that there is $\delta_j \in S(p^I) \setminus S(q^I)$. By Claim 3, we can assume w.l.o.g. that $j \neq \mu(\eta')$. But then $\delta_j \notin S(\eta')$ – contradiction. Thus, $S(p^I) \subseteq S(q^I)$, and by (a.a) $\langle p^I, q^I \rangle \in \text{containedIn}^{\hat{I}}$. From the assumption $\langle q^I, \eta' \rangle \in \text{containedIn}^{\hat{I}}$ we obtain $\langle p^I, \eta' \rangle \in \text{containedIn}^{\hat{I}}$.
 - (a.e) η' is an E-element. Then again $S(\eta) = \emptyset$ follows from Claim 3 and $S(\eta) \subseteq S(\eta')$. The claim follows as in (a.c).
- (b) η is a B-element for policy p . Then $S(p^I) \subseteq S(\eta')$, so $\langle p^I, \eta' \rangle \in \text{containedIn}^{\hat{I}}$ by (a). From the assumption $\langle \eta, p^I \rangle \in \text{containedIn}^{\hat{I}}$ we obtain $\langle \eta, \eta' \rangle \in \text{containedIn}^{\hat{I}}$.
- (c) η is a C-element for policy p .
 - (c.a) η' is an A-element for policy q . Then $S(p^I) \subseteq S(q^I)$ as in (a.d), so $\langle p^I, q^I \rangle \in \text{containedIn}^{\hat{I}}$ by (a.a). From the assumption $\langle \eta, p^I \rangle \in \text{containedIn}^{\hat{I}}$ we obtain $\langle \eta, q^I \rangle \in \text{containedIn}^{\hat{I}}$.
 - (c.b) η' is a B-element for policy q . Then $S(\eta) \subseteq S(q^I)$ and thus also $\langle \eta, q^I \rangle \in \text{containedIn}^{\hat{I}}$ by (c.a). From the assumption $\langle q^I, \eta' \rangle \in \text{containedIn}^{\hat{I}}$ we obtain $\langle \eta, \eta' \rangle \in \text{containedIn}^{\hat{I}}$.
 - (c.c) η' is a C-element for policy q . If $S(\eta) = \emptyset$, then $\langle p, p_{\perp} \rangle \in \text{gfp}(P_T)$ and the claim follows as in (a.c). Otherwise the claim follows since $\eta' = \eta$. For a contradiction, suppose $\eta \neq \eta'$. Since $S(\eta) \neq \emptyset$, there is $\delta_j \in S(\eta)$. By Claim 3 we can assume w.l.o.g. that $j = \mu(\eta')$ (since $\mu(\eta') \neq \mu(\eta)$ by definition of μ). But then $\delta_j \notin S(\eta')$ – contradiction.
 - (c.d) η' is a D-element for policy q . Then $S(\eta) \subseteq S(q^I)$ can be shown as in (a.d). The claim then follows as in (c.b).

- (c.e) η' is an E-element. Then again $S(\eta) = \emptyset$ follows from Claim 3 and $S(\eta) \subseteq S(\eta')$. This implies $\langle \eta, p_{\perp}^{\mathcal{I}} \rangle \in \text{containedIn}^{\hat{\mathcal{I}}}$ by (c.a). Together with $\langle p_{\perp}^{\mathcal{I}}, \eta' \rangle \in \text{containedIn}^{\hat{\mathcal{I}}}$ we obtain $\langle \eta, \eta' \rangle \in \text{containedIn}^{\hat{\mathcal{I}}}$.
- (d) η is a D-element for policy p . The reasoning for the individual cases is analogous to (c). We omit the details.
- (e) η is an E-element. The claim is immediate if $\eta = \eta'$. Otherwise, Claim 3 and $S(\eta) \subseteq S(\eta')$ together imply $S(\eta') = \mathcal{A}^{\hat{\mathcal{I}}}$. By the definition of $S(\cdot)$, this can only happen if η' is an A-element or a B-element for a policy q with $\langle p_{\top}, q \rangle \in \text{gfp}(P_T)$. Using (a) and (b) above, we thus find $\langle p_{\top}^{\mathcal{I}}, \eta' \rangle \in \text{containedIn}^{\hat{\mathcal{I}}}$. Together with $\langle \eta, p_{\top}^{\mathcal{I}} \rangle \in \text{containedIn}^{\hat{\mathcal{I}}}$, we obtain $\langle \eta, \eta' \rangle \in \text{containedIn}^{\hat{\mathcal{I}}}$.

This finishes the “if” direction of (\ddagger) , and thus completes the proof that $\hat{\mathcal{I}}$ is a P -model of T . It remains to show that $\hat{\mathcal{I}} \models \varphi$ implies $\mathcal{I} \models \varphi$ for the facts φ as in the claim. By Lemma 2, $\hat{\mathcal{I}} \models \varphi$ implies $\mathcal{I}' \models \varphi$. And by Lemma 3, this implies $\mathcal{I} \models \varphi$ as required. \square

We are now finally in a position to prove Theorem 1.

Proof (of Theorem 1). The “only if” direction is immediate, since all P -models of T are also models of $T \cup T_{\text{ci}} \cup \text{Cl}(\text{gfp}(P_T)) \cup T_{\text{ct}}^-$. For the “if” direction, consider any formula φ as in the claim. We show the contrapositive of the claim. Assume that $T \cup T_{\text{ci}} \cup \text{Cl}(\text{gfp}(P_T)) \cup T_{\text{ct}}^- \not\models \varphi$. Then there is an interpretation \mathcal{I} with $\mathcal{I} \models T \cup T_{\text{ci}} \cup \text{Cl}(\text{gfp}(P_T)) \cup T_{\text{ct}}^- \cup \{\neg\varphi\}$.

If φ is a CF formula, then so is $\neg\varphi$ (strictly speaking, this is not a CF formula, but it is clearly equivalent to one). Let $T' := T \cup \{\neg\varphi\}$. Then $\mathcal{I} \models T'$. Moreover, there is an interpretation \mathcal{J} that agrees with \mathcal{I} on all symbols other than possibly `containedIn` and `conformsTo`, such that $\mathcal{J} \models T' \cup T_{\text{ci}} \cup \text{Cl}(\text{gfp}(P_{T'})) \cup T_{\text{ct}}^-$, i.e. which satisfies $\text{Cl}(\text{gfp}(P_{T'}))$ rather than just $\text{Cl}(\text{gfp}(P_T))$. This follows since `containedIn` and `conformsTo` do not occur in T' , and since `conformsTo` only occurs positively in T_{ct}^- and nowhere else. Applying Proposition 2 to \mathcal{J} , we find a P -model $\hat{\mathcal{J}}$ of T' . By monotonicity of first-order logic, $\text{gfp}(P_T) \subseteq \text{gfp}(P_{T'})$, and thus $\hat{\mathcal{J}}$ is a P -model of T . But $\hat{\mathcal{J}} \models \neg\varphi$, so φ cannot be a P -consequence of T .

If φ is a fact over `containedIn` or `conformsTo`, then we can apply Proposition 2 to obtain a P -model $\hat{\mathcal{I}}$ of T such that $\hat{\mathcal{I}} \not\models \varphi$, so again φ cannot be a P -consequence of T . \square

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.⁸ However, Theorem 1 asserts that we can generally evaluate formal

⁸ This is easy to see in many ways, for example since (as noted below) CF allows us to express description logics like *SRIQ*, whereas CF does not impose the regularity or acyclicity condi-

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 [33], 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 the lightweight OWL QL or OWL RL languages [33]. 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 [33]. 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 as shown in Appendix A, so that all of our earlier results carry over to this extension.

tions that are essential for obtaining decidability of reasoning in these logics [15]. An example of a more direct proof is by reduction of the Post Correspondence Problem as demonstrated in [20, Section 4.2.2].

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:⁹

$$\top \sqsubseteq \exists \text{containedIn}.\{p_{\top}\} \quad \top \sqsubseteq \exists \text{containedIn}^{-1}.\{p_{\perp}\}$$

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:

$$\text{PD: Usage} \sqcup \text{Derivation} .$$

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

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

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 `Process` \sqcap \neg `Derivation`. We can thus express CC BY-ND as

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

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:

$$\begin{aligned} \text{BY-SA: } & C_{BY} \sqcap \forall \text{wasGeneratedBy}^{-1}.\forall \text{hasPolicy}.\{ \exists \text{containedIn}.\{\text{BY-SA}\} \sqcap \\ & \quad \exists \text{containedIn}^{-1}.\{\text{BY-SA}\} \}. \end{aligned}$$

To validate the basic practicability of this modelling approach, we used the OWL reasoner Hermit¹⁰ to compute the fixed point semantics of the policy system. We then conducted some basic tests with the formalised CC policies.¹¹ Not surprisingly, it can

⁹ 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.

¹⁰ <http://www.hermit-reasoner.com/>

¹¹ 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

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.¹²

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.

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:

$$\text{PD: Usage}(x) \vee \text{Derivation}(x).$$

¹² see Point 2.16 in <http://wiki.creativecommons.org/FAQ>, accessed 15th June 2011

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:

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

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 [21] for details):

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

Using this auxiliary modelling, we can easily express CC BY-ND and CC BY-SA:

$$\begin{aligned} \text{BY-ND} : & \text{Usage}(x) \wedge \text{wasTriggeredBy}(x, y) \wedge \text{Attribution}(y) \\ \text{BY-SA} : & (\text{Usage}(x) \wedge \text{wasTriggeredBy}(x, y) \wedge \text{Attribution}(y)) \vee \\ & (\text{Derivation}(x) \wedge \text{wasGeneratedBy}(z, x) \wedge \text{hasPolicy}(z, v) \wedge \\ & \text{containedIn}(v, \text{BY-SA}) \wedge \text{containedIn}(\text{BY-SA}, v)) . \end{aligned}$$

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 [31].

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 [32]. 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 [28]. 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 [5,26]. 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 [23]. 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., [25] 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 [29]. 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 of the related ISWC publication 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), available at <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'98). 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'90). pp. 621–626. AAAI Press (1990)
6. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P. (eds.): *The Description Logic Handbook*. Cambridge University Press, second edn. (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'08). pp. 11–18 (2008)
9. Cheney, J., Gil, Y., Groth, P., Miles, S.: Requirements for Provenance on the Web. Available at http://www.w3.org/2005/Incubator/prov/wiki/User_Requirements, W3C Provenance Incubator Group (2010)
10. Dodds, L.: Rights statements on the Web of Data. *Nodalities Magazine* pp. 13–14 (2010)
11. Glimm, B., Krötzsch, M.: SPARQL beyond subgraph matching. In: Patel-Schneider et al. [27], pp. 241–256
12. Glimm, B., Rudolph, S., Völker, J.: Integrated metamodeling and diagnosis in OWL 2. In: Patel-Schneider et al. [27], 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'07). 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 *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 (19 September 2002), available at <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'03). 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'08). 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'04). pp. 3–12 (2004)
20. Krötzsch, M.: *Description Logic Rules, Studies on the Semantic Web*, vol. 008. IOS Press (2010)
21. Krötzsch, M., Speiser, S.: Expressing self-referential usage policies for the Semantic Web. Tech. Rep. 3014, Institute AIFB, Karlsruhe Institute of Technology (2011), available online at <http://www.aifb.kit.edu/web/Techreport3014>
22. Lessig, L.: CC in Review: Lawrence Lessig on Compatibility. Available at <http://creativecommons.org/weblog/entry/5709> (accessed 1st July 2011) (2005)
23. Lifshitz, V.: Circumscriptive theories: A logic-based framework for knowledge representation. *Journal of Philosophical Logic* 17, 391–441 (1988)

24. 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)
25. Motik, B.: On the properties of metamodeling in OWL. *J. of Logic and Computation* 17(4), 617–637 (2007)
26. 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)
27. Patel-Schneider, P.F., Pan, Y., Glimm, B., Hitzler, P., Mika, P., Pan, J., Horrocks, I. (eds.): *Proc. 9th Int. Semantic Web Conf. (ISWC’10)*, LNCS, vol. 6496. Springer (2010)
28. Ringelstein, C., Staab, S.: PAPEL: A language and model for provenance-aware policy definition and execution. In: *Proc. 8th Int. Conf. on Business Process Management (BPM’10)*. LNCS, vol. 6336, pp. 195–210. Springer (2010)
29. 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’92)*. pp. 346–353. ACM (1992)
30. 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.) *Proc. 8th Int. Semantic Web Conf. (ISWC’09)*. LNCS, vol. 5823, pp. 553–568. Springer (2009)
31. Seneviratne, O.W.: *Framework for Policy Aware Reuse of Content on the WWW*. Master thesis, Massachusetts Institute of Technology (2009)
32. Speiser, S., Studer, R.: A self-policing policy language. In: Patel-Schneider et al. [27], pp. 730–746
33. W3C OWL Working Group: *OWL 2 Web Ontology Language: Document Overview*. W3C Recommendation (27 October 2009), available at <http://www.w3.org/TR/owl2-overview/>
34. 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*, chap. I, pp. 1–31. IRM Press (2006)

A First-Order Logic

In this section, we provide a short introduction to our notation for first-order logic. We generally consider first-order logic without function symbols.

Definition 7. A signature Σ of first-order logic (without function symbols) consists of two disjoint finite sets \mathbf{C} of constants and \mathbf{P} of predicates, each associated with a natural number that defines its arity. Moreover, we assume a countably infinite set \mathbf{V} of variables that is disjoint from the signature. A term t is a constant or a variable $t \in \mathbf{C} \cup \mathbf{V}$.

An atomic formula (or simply atom) over Σ is an expression $p(\mathbf{t})$ where $p \in \mathbf{P}$ is a predicate and \mathbf{t} is a list of terms the length of which is the arity of p . The set of first-order formulae over Σ , and the set $\text{free}(\varphi)$ of free variables in formulae φ are inductively defined as the smallest set such that:

- every atom $p(\mathbf{t})$ over Σ is a formula, and $\text{free}(\varphi)$ is the set of variables in \mathbf{t} ;
- if φ is a formula, then so is $\neg\varphi$, and $\text{free}(\neg\varphi) := \text{free}(\varphi)$;
- if φ_1 and φ_2 are formulae, then so are $(\varphi_1 \circ \varphi_2)$ for $\circ \in \{\wedge, \vee, \rightarrow\}$, and $\text{free}(\varphi_1 \circ \varphi_2) := \text{free}(\varphi_1) \cup \text{free}(\varphi_2)$;

- if \mathbf{x} is a (finite) list of variables and φ is a formula, then so are $\exists \mathbf{x}.\varphi$ and $\forall \mathbf{x}.\varphi$, and $\text{free}(\exists \mathbf{x}.\varphi) := \text{free}(\forall \mathbf{x}.\varphi) := \text{free}(\varphi) \setminus \mathbf{x}$.

We write $\varphi[\mathbf{x}]$ to indicate that φ contains at most the free variables in \mathbf{x} (but possibly less).

A variable that is not free in φ is bound. A sentence is a formula without free variables. A theory is a set of sentences.

We generally consider the signature Σ to be fixed, and so will not mention it explicitly.

Definition 8. A first-order logic interpretation \mathcal{I} is a pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, consisting of a non-empty interpretation domain $\Delta^{\mathcal{I}}$ and an interpretation function $\cdot^{\mathcal{I}}$. The domain is a set of individuals that defines the (abstract) world within which all symbols are interpreted. Symbols of the signature are interpreted as follows:

- If $a \in \mathbf{C}$ is a constant, then $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$.
- If $p \in \mathbf{P}$ is a predicate of arity n , then $p^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}})^n$.

Here, $(\Delta^{\mathcal{I}})^n$ denotes the set of n -tuples of elements of $\Delta^{\mathcal{I}}$. A variable assignment \mathcal{Z} for \mathcal{I} is a mapping $\mathcal{Z} : \mathbf{V} \rightarrow \Delta^{\mathcal{I}}$. Given an element $\delta \in \Delta^{\mathcal{I}}$ and a variable $x \in \mathbf{V}$, we write $\mathcal{Z}\{x \mapsto \delta\}$ to denote the variable assignment that assigns x to δ , and that agrees with \mathcal{Z} on all other variables.

Given an interpretation \mathcal{I} and a variable assignment \mathcal{Z} for \mathcal{I} , the interpretation $t^{\mathcal{I}, \mathcal{Z}}$ of a term t is defined as follows:

- If $t \in \mathbf{C}$ then $t^{\mathcal{I}, \mathcal{Z}} := t^{\mathcal{I}}$.
- If $t \in \mathbf{V}$ then $t^{\mathcal{I}, \mathcal{Z}} := \mathcal{Z}(t)$.

For a list of terms $\mathbf{t} = \langle t_1, \dots, t_n \rangle$, we write $\mathbf{t}^{\mathcal{I}, \mathcal{Z}}$ to denote $\langle t_1^{\mathcal{I}, \mathcal{Z}}, \dots, t_n^{\mathcal{I}, \mathcal{Z}} \rangle$.

The truth value $\varphi^{\mathcal{I}, \mathcal{Z}}$ of a formula φ is defined as follows:

- Set $\top^{\mathcal{I}, \mathcal{Z}} := \text{true}$ and $\perp^{\mathcal{I}, \mathcal{Z}} := \text{false}$.
- For $\varphi = p(\mathbf{t})$, set $\varphi^{\mathcal{I}, \mathcal{Z}} := \text{true}$ if $\mathbf{t}^{\mathcal{I}, \mathcal{Z}} \in p^{\mathcal{I}}$, and $\varphi^{\mathcal{I}, \mathcal{Z}} := \text{false}$ otherwise.
- For $\varphi = \neg\psi$, define $\varphi^{\mathcal{I}, \mathcal{Z}} := \text{true}$ if $\psi^{\mathcal{I}, \mathcal{Z}} = \text{false}$, and $\varphi^{\mathcal{I}, \mathcal{Z}} := \text{false}$ otherwise.
- For $\varphi = (\psi_1 \wedge \psi_2)$, define $\varphi^{\mathcal{I}, \mathcal{Z}} := \text{true}$ if $\psi_i^{\mathcal{I}, \mathcal{Z}} = \text{true}$ for all $i \in \{1, 2\}$, and $\varphi^{\mathcal{I}, \mathcal{Z}} := \text{false}$ otherwise.
- For $\varphi = (\psi_1 \vee \psi_2)$, define $\varphi^{\mathcal{I}, \mathcal{Z}} := \text{true}$ if $\psi_i^{\mathcal{I}, \mathcal{Z}} = \text{true}$ for some $i \in \{1, 2\}$, and $\varphi^{\mathcal{I}, \mathcal{Z}} := \text{false}$ otherwise.
- For $\varphi = (\psi_1 \rightarrow \psi_2)$, define $\varphi^{\mathcal{I}, \mathcal{Z}} := \text{true}$ if $\psi_1^{\mathcal{I}, \mathcal{Z}} = \text{false}$ or $\psi_2^{\mathcal{I}, \mathcal{Z}} = \text{true}$, and $\varphi^{\mathcal{I}, \mathcal{Z}} := \text{false}$ otherwise.
- For $\varphi = \exists \mathbf{x}.\psi$, define $\varphi^{\mathcal{I}, \mathcal{Z}} := \text{true}$ if there is some list δ of domain elements such that $\psi^{\mathcal{I}, \mathcal{Z}\{x \mapsto \delta\}} = \text{true}$, and set $\varphi^{\mathcal{I}, \mathcal{Z}} := \text{false}$ otherwise.
- For $\varphi = \forall \mathbf{x}.\psi$, define $\varphi^{\mathcal{I}, \mathcal{Z}} := \text{true}$ if, for all lists δ of domain elements, we find that $\psi^{\mathcal{I}, \mathcal{Z}\{x \mapsto \delta\}} = \text{true}$, and set $\varphi^{\mathcal{I}, \mathcal{Z}} := \text{false}$ otherwise.

A formula φ is satisfied (or modelled) by an interpretation \mathcal{I} and a variable assignment \mathcal{Z} for \mathcal{I} if $\varphi^{\mathcal{I}, \mathcal{Z}} = \text{true}$. We also write $\mathcal{I}, \mathcal{Z} \models \varphi$ in this case. The truth value of sentences does not depend on any variable assignment, so we can omit assignments in this case

and simply write $\mathcal{I} \models \varphi$ to say that \mathcal{I} satisfies φ under any variable assignment. A theory T is satisfied (or modelled) by \mathcal{I} if \mathcal{I} satisfies all elements of T . We write $\mathcal{I} \models T$ in this case, and say that \mathcal{I} is a model of T .

Given a formula φ with one free variable x , we write $\varphi^{\mathcal{I}}$ to denote the set $\{\delta \in \Delta^{\mathcal{I}} \mid \mathcal{I}, \{x \mapsto \delta\} \models \varphi\}$. Likewise, if a formula contains only variables \mathbf{x} , then we can write $\mathcal{I}, \{\mathbf{x} \mapsto \delta\} \models \varphi$ to express that $\varphi^{\mathcal{I}, \mathcal{Z}(\mathbf{x} \mapsto \delta)} = \text{true}$ for any variable assignment \mathcal{Z} .

Logical consequences are defined by the standard model-theoretic notions as follows:

Definition 9. Consider first-order logic theories T and T' .

- T is consistent (or satisfiable) if it has a model and inconsistent (or unsatisfiable) otherwise,
- T entails T' , written $T \models T'$, if all models of T are also models of T' .

This terminology is extended to formulae by treating them as singleton theories. A theory or formula that is entailed is also called a logical consequence.

Finally, for every fixed signature Σ , one can introduce equality as an expressive feature into the logic by axiomatising its effects (as opposed to considering equality as a special predicate of the logic that has a hard-coded semantics):

Definition 10. Consider a signature Σ and let \approx be a predicate of arity two that does not occur in Σ . The standard equality theory for Σ consists of the following sentences:

$$\forall x. x \approx x, \quad (13)$$

$$\forall x, y. x \approx y \rightarrow y \approx x, \quad (14)$$

$$\forall x, y, z. x \approx y \wedge y \approx z \rightarrow x \approx z, \quad (15)$$

$$\forall x_1, \dots, x_n, y. p(x_1, \dots, x_l, \dots, x_n) \wedge x_l \approx y \rightarrow p(x_1, \dots, y, \dots, x_n), \quad (16)$$

where (16) is instantiated for all predicates $p \in \mathbf{P}$ of arity n , and all $l \in \{1, \dots, n\}$.

The equality symbol \approx that has thus been defined can then be used in logical theories.