Sticky Policies in OWL2: Extending $\mathcal{PL}$ with Fixpoints and Transitive Closure

Piero A. Bonatti , Luigi Sauro ,
Università di Napoli Federico II
{pab, luigi.sauro}@unina.it

Abstract

$\mathcal{PL}$ is a low-complexity profile of OWL2, expressly designed to encode data usage policies and personal data protection regulations - such as the GDPR - in a machine understandable way. With $\mathcal{PL}$, the compliance of privacy policies with the GDPR and with the data subjects' consent to processing can be checked automatically and in real time. In this paper, we extend $\mathcal{PL}$ to support "sticky policies". They are a sort of license that applies to data transfers, and specifies how the recipient can use the data. Sticky policies may be "recursive", i.e. they may apply not only to the first data transfer, but also to all subsequent transfer operations that the (direct or indirect) recipients may execute in the future. Recursive sticky policies can be encoded with fixpoints or transitive role closure. In this paper we prove that such extensions make compliance checking intractable. Since the scalability of compliance checking is a major requirement in this area, these results justify a specialized, polynomial-time approach to encoding sticky policies.

1 Introduction

The European General Data Protection Regulation (GDPR) has changed the landscape of personal data processing. Due to the heavy sanctions and reputation loss incurred in case of violations, the legal entities that process personal data (*controllers* in GDPR's terminology), are calling for automated support to compliance. The European H2020 projects SPECIAL[1] and TRAPEZE[2] tackle this need by means of semantic technnologies, that effectively yield reliability, interoperability, extensibility, flexibility, usability and scalability (Bonatti, Sauro, and Langens 2021). In particular, usability and scalability have been addressed in SPECIAL by identifying a profile of OWL2, called OWL2-PL (policy language), that is simpler to grasp for users with no background in logic, and can be processed very efficently using specialized reasoners (Bonatti et al. 2020). The description logic correponding to OWL2-PL is called $\mathcal{PL}$. One of the goals of TRAPEZE is extending $\mathcal{PL}$ with additional constructs, so as to support more general use cases. In this paper we focus on extending $\mathcal{PL}$ to support *sticky policies*. They are a sort of license that applies to data transfers, and specifies how direct and indirect recipients can use the data. Before delving into technical details, let us summarize the basic features of the semantic policy framework of SPECIAL and TRAPEZE that we are going to enrich.

The privacy policies of controllers, the consent to processing of data subjects, and the personal data protection regulations themselves are all policies. A policy can be identified with the set of operations authorized by the policy; such operations can be abstracted as tuples of attributes (Bonatti, De Capitani di Vimercati, and Samarati 2002). Accordingly, the semantic policy framework encodes policies in OWL2 as classes of reified tuples. In the personal data protection domain, the characteristic attributes of operations include (not exclusively) the data category being processed, the purpose and the nature of the processing, the third parties with which data are shared, the legal basis of the processing, and other information related to the regulation (Bonatti et al. 2020). For example, a privacy policy stating that email addresses are transferred to third parties for advertising purposes, based on the data subject's consent, is encoded in description logics as follows:

$$\begin{aligned}&\exists \mathsf{has\_data}.\mathsf{Email} \sqcap \\ &\quad \exists \mathsf{has\_processing}.\mathsf{Transfer} \sqcap \\ &\quad \exists \mathsf{has\_purpose}.\mathsf{Advertising} \sqcap \\ &\quad \exists \mathsf{has\_legal\_basis}.\mathsf{Consent}\,.\end{aligned}$$

A (privacy) policy P *complies* with a policy P' (a consent statement or regulation) if, and only if, all the operations authorized by P are also authorized by P', that is, P is a subclass of P'. Thus, compliance checking is naturally reduced to *subsumption checking* in description logic. For example, if Email is a subclass of $\mathsf{Contact}$ (contact data) and $\mathsf{Advertising}$ a subclass of $\mathsf{Marketing}$, then the above policy complies with the consent to transferring contact data to third parties for marketing purposes, which is formalized as:

$$\begin{aligned}&\exists \mathsf{has\_data}.\mathsf{Contact} \sqcap \\ &\quad \exists \mathsf{has\_processing}.\mathsf{Transfer} \sqcap \\ &\quad \exists \mathsf{has\_purpose}.\mathsf{Marketing}\,.\end{aligned}$$

Note that the classical semantics of subsumption checking (based on entailment) treats policies as *closed* policies, that is, only what is explicitly allowed is permitted. For example, a privacy policy whose purpose is not subsumed by $\mathsf{Marketing}$ would not comply with the above consent statement.

[1] https://specialprivacy.ercim.eu/
[2] https://trapeze-project.eu/

Data transfers can be constrained with sticky policies. For instance, the above consent statement can be refined to state that third parties are allowed (only) to directly use the data subject's contact information for marketing purposes (which implicitly forbids further transfers to third parties, if we assume that $\mathsf{Transfer}$ is *not* subsumed by $\mathsf{DirectUse}$):

$$\begin{aligned}&\exists \mathsf{has\_data}.\mathsf{Contact} \sqcap \\ &\quad\exists \mathsf{has\_processing}.\mathsf{Transfer} \sqcap \\ &\quad\exists \mathsf{has\_purpose}.\mathsf{Marketing} \sqcap \\ &\quad\exists \mathsf{sticky}.(\\ &\qquad\exists \mathsf{has\_data}.\mathsf{Contact} \sqcap \\ &\qquad\exists \mathsf{has\_processing}.\mathsf{DirectUse} \sqcap \\ &\qquad\exists \mathsf{has\_purpose}.\mathsf{Marketing}) \,.\end{aligned}$$

Property "sticky" should be functional in order to avoid the ambiguities that may arise from the application of different, overlapping policies.

By nesting sticky policies, one may allow further transfers to third parties, as in the following expression:

$$P_0 \sqcap \exists \mathsf{sticky}.(P_1 \sqcap \exists \mathsf{sticky}.(P_2 \sqcap \ldots \exists \mathsf{sticky}.P_n \ldots)) \,.$$

Here the controller should satisfy P_0, while its direct recipients have to satisfy P_1; in turn, their recipients should satisfy P_2, and so on. Clearly, the above concept regulates only finite disclosure chains of length n. In general, however, disclosure chains can be unbounded. Then sticky policies should be *recursive*, that is, they should identically apply to all (direct and indirect) recipients, and allow each of them to further transfer data to other third parties. Formally, one would like to express something like the infinitary concept

$$P_0 \sqcap \exists \mathsf{sticky}.(P_1 \sqcap \exists \mathsf{sticky}.(P_1 \sqcap \exists \mathsf{sticky}.(P_1 \sqcap \ldots))) \,.$$

Logic provides at least two ways of expressing the above class with a finite expression: greatest fixpoint operators (ν) and transitive role closure (R^+). Since sticky is a functional property, the following are equivalent to the above concept:

$$\begin{aligned}&P_0 \sqcap \nu X.(\exists \mathsf{sticky}.(P_1 \sqcap X)) \,, \\ &P_0 \sqcap (\exists \mathsf{sticky}.P_1) \sqcap \forall \mathsf{sticky}^+.(\exists \mathsf{sticky}.P_1) \,.\end{aligned}$$

Accordingly, in this paper, we are going to investigate the extensions of $\mathcal{PL}$ with greatest fixpoints, universal quantifiers, and transitive role closure. The ultimate goal is supporting sticky policies in a very efficient manner, as some of the use cases of interest to the industrial partners of SPECIAL and TRAPEZE require to complete thousands of compliance checks per second.

Example 1 (Streaming Scenario)**.** Telecom providers, that today are also Internet providers, receive from their base stations about 15000 call records per second, and almost 10000 probing records per second from their wi-fi network. The data contained in the aforementioned records are of great interest for strategic applications and services, such as location-based services and taylored recommendations; however, call and probing records contain personal data, and the European regulation on data protection prohibits the above usage without the consent of the data subjects. Without consent, even storing the data temporarily, waiting for a batch process to discard the records that cannot be processed, is illegal. Then the description of how and why each application processes the data must be checked in real-time for compliance with the consent statements that apply to the records being processed. This scenario is further complicated by the fact that each data subject can withdraw or modify her consent anytime, and that she may selectively decide to opt in or out each processing option (e.g. a customer might accept only location tracking, and not internet tracking). $\square$

More generally, subsumption checks are going to be as frequent as access control checks in our target applications. Therefore, the semantic policy framework must satisfy extreme scalability requirements, that are not common in the knowledge representation area. Thus, an implied necessary requirement is that *subsumption checking must be possible in deterministic polynomial time*, and the degree of the polynomial should be low.

In (Bonatti et al. 2020), it has been proved – both theoretically and experimentally – that $\mathcal{PL}$ satisfies the scalability requirement, and that real policies can be checked for compliance in a few hundreds of μ-seconds using a sequential Java implementation (i.e. a technology that is not intrinsically performant). The challenge now is supporting sticky policies while preserving the performance of $\mathcal{PL}$.

In sections 3 and 4 we prove that the extensions of $\mathcal{PL}$ with ν, and with the combination of $\forall$ and transitive role closure – respectively – are intractable. We prove lower complexity bounds at the first level of the polynomial hierarchy; they suffice to conclude that the above extensions of $\mathcal{PL}$ are not suitable for our purposes. Concerning upper complexity bounds, the complexity of the logics supporting ν and transitive role closure is typically much higher, namely, EXPTIME or harder, if a full set of boolean operators is supported. In our setting, getting a tighter complexity estimate is difficult, due to the limited expressiveness of $\mathcal{PL}$; this aspect is further detailed in section 6.

The intractability results justify a tractable approach tailored to the use cases, based on a restricted language $\mathcal{PL}_0^{\mathsf{sticky}}$ that will be illustrated in section 5. This language preserves the asymptotic complexity of reasoning of $\mathcal{PL}$, so it is a promising approach to sticky policy representation.

The basic notions about description logics and $\mathcal{PL}$ are recalled in the next section. Related and future works are discussed in section 6.

2 Preliminary Definitions

We assume the reader to be familiar with the basic notions of Description Logics (DL) (Baader et al. 2003). Here we recall only the aspects needed for this work. The DL languages of our interest are built from countably infinite sets of concept names ($\mathsf{N_C}$), role names ($\mathsf{N_R}$), and concrete property names ($\mathsf{N_F}$). An *interpretation* $\mathcal{I}$ is a structure $\mathcal{I} = (\Delta^\mathcal{I}, \cdot^\mathcal{I})$ where $\Delta^\mathcal{I}$ is a nonempty set, and the *interpretation function* $\cdot^\mathcal{I}$ is such that (i) $A^\mathcal{I} \subseteq \Delta^\mathcal{I}$ if $A \in \mathsf{N_C}$; (ii) $R^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$ if $R \in \mathsf{N_R}$; (iii)$f^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \mathbb{N}$ if $f \in \mathsf{N_F}$, where $\mathbb{N}$ denotes the set of natural numbers.

Compound concepts and roles are built from concept names, role names, and the logical constructors listed in Ta-

Name	Syntax	Semantics
transitive closure	R^+	$\bigcup_{i\geq 1}(R^i)^{\mathcal{I}} \quad (R \in \mathsf{N_R})$
top	$\top$	$\top^{\mathcal{I}} = \Delta^{\mathcal{I}}$
bottom	$\bot$	$\bot^{\mathcal{I}} = \emptyset$
complement	$\neg C$	$\neg C^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$
intersection	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
union	$C \sqcup D$	$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
existential restriction	$\exists R.C$	$\{d \in \Delta^{\mathcal{I}} \mid \exists (d,e) \in R^{\mathcal{I}} : e \in C^{\mathcal{I}}\}$
universal restriction	$\forall R.C$	$\{d \in \Delta^{\mathcal{I}} \mid \forall (d,e) \in R^{\mathcal{I}} : e \in C^{\mathcal{I}}\}$
interval restrictions	$\exists f.[\ell, u]$	$\{d \in \Delta^{\mathcal{I}} \mid \exists i \in [\ell, u] : (d,i) \in f^{\mathcal{I}}\}$

Table 1: Syntax and semantics of some DL constructs.

$\mathcal{PL}$ axiom α	$\mathcal{I} \models \alpha$ iff:
$A \sqsubseteq B$	$A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$
$A_1 \sqcap \cdots \sqcap A_n \sqsubseteq B$	$A_1^{\mathcal{I}} \cap \cdots \cap A_n^{\mathcal{I}} \subseteq B^{\mathcal{I}}$
$\mathsf{disj}(A, B)$	$A^{\mathcal{I}} \cap B^{\mathcal{I}} = \emptyset$
$\mathsf{range}(R, A)$	$(x, y) \in R^{\mathcal{I}} \to y \in A^{\mathcal{I}}$
$\mathsf{func}(R)$	$(x, y) \in R^{\mathcal{I}} \wedge (x, z) \in R^{\mathcal{I}} \to y = z$

Table 2: Axioms for $\mathcal{PL}$ knowledge bases. Here A, with possible subscripts, and B range over concept names while R is a role name.

ble 1. We use metavariables A, B for concept names, C, D for (possibly compound) concepts, R, S for role names, and f, g for concrete property names. The third column shows how to extend the valuation $\cdot^{\mathcal{I}}$ of an interpretation $\mathcal{I}$ to compound expressions; in the first row, let $(R^1)^{\mathcal{I}} = R^{\mathcal{I}}$ and $(R^i)^{\mathcal{I}} = R^{\mathcal{I}} \circ (R^{i-1})^{\mathcal{I}}$, for all $i > 1$.

In (Bonatti et al. 2020) several extensions of $\mathcal{PL}$ are considered, where knowledge bases can be expressed with fragments of Horn-$\mathcal{SRIQ}$, under suitable restriction related to import-by-query and knowledge compilation techniques, cf. (Bonatti et al. 2020). Here, we consider only the axioms that suffice to prove our lower bounds. These axioms and the corresponding semantics are shown in Table 2.[3] As usual, an interpretation $\mathcal{I}$ is a model of a $\mathcal{PL}$ knowledge base $\mathcal{KB}$ (in symbols, $\mathcal{I} \models \mathcal{KB}$) if and only if $\mathcal{I} \models \alpha$, for all axioms α occurring in $\mathcal{KB}$.

In $\mathcal{PL}$, the query language differs from the knowledge base language, and is equipped with the constructs needed to express policies. Specifically, a *simple $\mathcal{PL}$ concept* is defined by the following grammar, where $A \in \mathsf{N_C}$, $R \in \mathsf{N_R}$, and $f \in \mathsf{N_F}$:

$$C ::= A \mid \bot \mid \exists f.[l, u] \mid \exists R.C \mid C \sqcap C\,.$$

A *full $\mathcal{PL}$ concept* is a union $D_1 \sqcup \ldots \sqcup D_n$ of simple $\mathcal{PL}$ concepts ($n \geq 1$). $\mathcal{PL}$'s *subsumption queries* are inclusions $C \sqsubseteq D$ where C, D are full $\mathcal{PL}$ concepts. A $\mathcal{PL}$ subsumption query $C \sqsubseteq D$ is *simple* if both C and D are simple.

If $\mathcal{KB}$ entails a subsumption query $C \sqsubseteq D$ (in symbols, $\mathcal{KB} \models C \sqsubseteq D$), then we say that C *complies with* D (under $\mathcal{KB}$). Compliance checking is in general coNP-complete, however it downgrades to P whenever the query $C \sqsubseteq D$ is *interval safe*, that is: for all interval constraints $\exists f.[l, u]$ and $\exists f'.[l', u']$ occurring in C and D, respectively, either $[l, u] \subseteq [l', u']$ or $[l, u] \cap [l', u'] = \emptyset$ (Bonatti et al. 2020).

Notably, coming back to the general case where C and D may contain partially overlapping intervals, and $C = C_1 \sqcup \cdots \sqcup C_n$, it is always possible to turn $C \sqsubseteq D$ into an equivalent, interval safe query by splitting the intervals of C in a suitable way. The resulting concept C' has size $O(|C| \cdot |D|^c)$, where $c = \max_{1\leq i\leq n} c_i$ and each c_i is the number of interval constraints occurring in C_i, for $i = 1, \ldots, n$. Fortunately, the exponent c is a fixed constant in our use cases, therefore, C' can be computed in polynomial time and compliance checking is tractable. In particular, policy encoding requires at most one interval constraint per simple concept (such interval is used to specify how long data are kept by the controller). In the following, no more details about these complexity issues will be needed; the interested reader is referred to (Bonatti et al. 2020) for a complete discussion.

Hereafter, in order to keep different sources of complexity cleanly separated in the complexity analysis, we consider (and extend) the restricted logic $\mathcal{PL}_0$ obtained from $\mathcal{PL}$ by disallowing interval constraints – therefore, in $\mathcal{PL}_0$, subsumptions are vacuously interval safe, and subsumption checking is tractable. $\mathcal{PL}$ will be considered again in section 6.

Next, let us define the greatest fixpoint operator ν that will be used in section 3 to augment the query language. To this aim, we consider a supplementary countably infinite set $\mathsf{N_V}$ of *variables*; similarly to an atomic concept, a variable X is interpreted as a set of individuals, $X^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$. Let $\mathcal{E}$ be a subset of $\Delta^{\mathcal{I}}$; by $\mathcal{I}[X \to \mathcal{E}]$ we mean the interpretation such that X is interpreted as $\mathcal{E}$, and all the other symbols are interpreted as in $\mathcal{I}$. Then, the semantics of a fixpoint concept $\nu X.C$ is the following:

$$(\nu X.C)^{\mathcal{I}} = \bigcup\{\mathcal{E} \subseteq \Delta^{\mathcal{I}} \mid \mathcal{E} \subseteq C^{\mathcal{I}[X \to \mathcal{E}]}\}\,.$$

Finally, a pointed interpretation is a pair $(\mathcal{I}, d)$, where $d \in \Delta^{\mathcal{I}}$. We say that $(\mathcal{I}, d)$ satisfies a concept C (in symbols, $\mathcal{I}, d \models C$) iff $d \in C^{\mathcal{I}}$. We also say that $(\mathcal{I}, d)$ is a model of a knowledge base $\mathcal{KB}$ if $\mathcal{I} \models \mathcal{KB}$.

3 $\mathcal{PL}$ with Fixpoints

Let $\mathcal{PL}_0^{\nu}$ be the extension of $\mathcal{PL}_0$ where greatest fixpoints may occur in subsumption queries. At a first glance, $\mathcal{PL}_0^{\nu}$ constitutes a promising way of encoding sticky policies, due to its similarity with $\mathcal{EL}$ with greatest fixpoints, that has been proved to be tractable in (Lutz, Piro, and Wolter 2010).

However, the interplay of greatest fixpoints with functional roles (that are supported only in $\mathcal{PL}$) makes subsumption checking at least coNP-hard. This holds not only for

[3]Interestingly, the axioms listed in Table 2 also suffice to axiomatize the vocabularies that are actually employed in the use cases of SPECIAL and TRAPEZE. Such vocabularies are being developed by the DPVCG,[4] a community group of the W3C devoted to the development of standardized data privacy vocabularies.

$\mathcal{PL}$, but also for all the description logics that support $\sqcap$, $\sqsubseteq$, and func in knowledge base axioms, and $\sqcap$, $\exists$, and ν in subsumption queries.

Theorem 2. *Let $\mathcal{DL}$ be any description logics that supports $\sqcap$, $\sqsubseteq$, and func in knowledge base axioms, and $\sqcap$, $\exists$, and ν in subsumption queries. Subsumption checking in $\mathcal{DL}$ is coNP-hard.*

Proof. The proof is by reduction of the validity problem to subsumption checking. Let $\phi = \bigvee_{i=1}^{m} \ell_{i,1} \wedge \ell_{i,2} \wedge \ell_{i,3}$ be any propositional formula in 3-DNF, and let $x_1, \ldots, x_k$ be the propositional variables occurring in ϕ. Introduce a fresh concept name A_i for each x_i and a fresh concept name $\bar{A}_i$ for each negative literal $\neg x_i$. For each propositional literal ℓ, let $\tilde{\ell}$ denote the corresponding concept name. Informally speaking, the knowledge base $\mathcal{KB}$ encodes ϕ in such a way that if an individual d satisfies ϕ (up to the correspondence between literals and concepts), then d satisfies also a distinguished atomic concept F that represents the truth of ϕ. More formally, let $\mathcal{KB}$ consist of the following axioms, where each concept B_i represents the truth of the i-th disjunct of ϕ:

$$\begin{aligned}&\tilde{\ell}_{i,1} \sqcap \tilde{\ell}_{i,2} \sqcap \tilde{\ell}_{i,3} \sqsubseteq B_i\\ &B_i \sqsubseteq F\\ &\mathsf{func}(R)\end{aligned}$$

(where $i = 1, \ldots, m$). The use of role R will be explained later.

Greatest fixpoints are used to create periodic chains of instance types. To make this more precise, we need some auxiliary definitions. First, for all $i = 1, \ldots, k$ (where k is the number of propositional variables in ϕ), define

$$\begin{aligned}C_i^1 &= A_i \sqcap \exists R.X\\ C_i^{j+1} &= \bar{A}_i \sqcap \exists R.C_i^j\end{aligned}$$

(where X is a concept variable). For example,

$$\begin{aligned}C_i^2 &= \bar{A}_i \sqcap \exists R.(A_i \sqcap \exists R.X)\\ C_i^3 &= \bar{A}_i \sqcap \exists R.(\bar{A}_i \sqcap \exists R.(A_i \sqcap \exists R.X))\\ C_i^4 &= \bar{A}_i \sqcap \exists R.(\bar{A}_i \sqcap \exists R.(\bar{A}_i \sqcap \exists R.(A_i \sqcap \exists R.X)))\\ &\vdots\end{aligned}$$

Note that the instances of any concept $\nu X.C_i^j$ are the first elements of infinite R-chains where A_i is satisfied (at least) every j steps, while the other elements satisfy (at least) $\bar{A}_i$.

Let $p_1 = 2, p_2 = 3, p_3 = 5, \ldots, p_k$ be the first k prime numbers, and define:

$$C = \prod_{i=1}^{k} \nu X.C_i^{p_i} \,. \tag{1}$$

Claim: ϕ is valid iff $\mathcal{KB} \models C \sqsubseteq \nu X.(F \sqcap \exists R.X)$.

First we prove the "only if" part of the claim. If ϕ is valid, then for all models $\mathcal{I}$ of $\mathcal{KB}$ and all individuals $d \in \Delta^{\mathcal{I}}$, the following clearly holds: If d belongs to $A_i^{\mathcal{I}}$ or $\bar{A}_i^{\mathcal{I}}$ for all $i = 1, \ldots, k$, then $d \in F^{\mathcal{I}}$. Note that C forces all the direct and indirect R-successors of its instances to be in $A_i^{\mathcal{I}}$ or $\bar{A}_i^{\mathcal{I}}$, for all $i = 1, \ldots, k$, so all such successors are in $F^{\mathcal{I}}$. It follows, by definition of ν, that all the instances of C belong to $\nu X.(F \sqcap \exists R.X)$, which proves the "only if" part of the claim.

To prove the "if" part of the claim, suppose that ϕ is not valid, and construct a counterexample $\mathcal{I}$ to the subsumption as follows. Let $\Delta^{\mathcal{I}}$ be an infinite set $\{d_i \mid i \in \mathbb{N}\}$. Let $R^{\mathcal{I}} = \{(d_i, d_{i+1}) \mid i \in \mathbb{N}\}$. For all $i \in \mathbb{N}$ and $j = 1, \ldots, k$, let $d_i \in A_j^{\mathcal{I}}$ iff $i \bmod p_j = 0$, and let $\bar{A}_j^{\mathcal{I}} = \Delta^{\mathcal{I}} \setminus A_j^{\mathcal{I}}$. Finally, for all $i = 1, \ldots, m$, let $B_i^{\mathcal{I}} = (\tilde{\ell}_{i,1} \sqcap \tilde{\ell}_{i,2} \sqcap \tilde{\ell}_{i,3})^{\mathcal{I}}$ and $F^{\mathcal{I}} = \bigcup_{i=1}^{m} B_i^{\mathcal{I}}$. By construction, we have both that $\mathcal{I}$ satisfies $\mathcal{KB}$, and $d_1 \in C^{\mathcal{I}}$. Moreover, since the numbers $p_1, \ldots, p_k$ that determine the periodic behavior of C's fixpoints are distinct prime numbers, the individuals $d_1, d_2, \ldots, d_i, \ldots$ collectively satisfy all possible combinations of literal encodings that contain no pair of complementary concepts A_j and $\bar{A}_j$ ($j = 1, \ldots, k$). So the concepts satisfied by $\mathcal{I}$'s elements represent all possible truth assignments to $x_1, \ldots, x_k$.[5] One of this truth assignments falsifies ϕ, by assumption, so – by construction – there exists $d_j \in \Delta^{\mathcal{I}}$ such that $d_j \notin F^{\mathcal{I}}$. As a consequence, $d_1 \notin \nu X.(F \sqcap \exists R.X)$. This proves that $C^{\mathcal{I}} \not\subseteq \nu X.(F \sqcap \exists R.X)^{\mathcal{I}}$, which completes the proof of the claim.

The reduction is correct by the claim; we are only left to show that it can be computed in polynomial time. The size of the knowledge base and the size of the concept $\nu X.(F \sqcap \exists R.X)$ are obviously polynomial in the size of ϕ, so we only have to provide a polynomial bound on the size of C. In order to see this, we use a results by Rosser (Rosser 1941). The k^{th} prime number p_k is bounded by

$$p_k < k(\log k + \log\log k + 2)$$

so, for sufficiently large k, $p_k < 2k^2$. It follows that the length of each concept $\nu X.C_i^{p_i}$ in (1) is $O(k^2)$ and the entire concept C is $O(k^3)$, so the reduction can be computed in polynomial time.[6] □

As a corollary, subsumption checking in $\mathcal{PL}_0^{\nu}$ is coNP-hard, even if neither disjointness axioms nor range axioms are used, and only one functional role is used.

4 $\mathcal{PL}$ with Universal Restrictions and Transitive Role Closure

Transitive role closure provides an alternative way of expressing sticky policies. Transitive closure can be regarded as a restricted form of fixpoint: every concept of the form $\forall R^+.C$ can be expressed with ν as $\nu X.\forall R.(C \sqcap X)$. So, reasoning in $\mathcal{PL}_0$ with transitive role closure might turn out to be less complex than reasoning in $\mathcal{PL}_0$ with greatest fixpoints. Unfortunately, tractability is not preserved, due to the interplay of $\forall$ and $\exists$. We are going to prove that the NP-complete EXACT COVER (XC) problem can be reduced

[5] In particular, each propositional interpretation $\{x_{i_1}, \ldots, x_{i_n}\}$ corresponds to the element d_h with $h = \prod_{l=1}^{n} p_{i_l}$.

[6] It is not hard to see that it can even be computed in logarithmic space.

to subsumption in two extensions of $\mathcal{PL}_0$ that employ the operators $\forall$ and $\cdot^+$ needed to encode sticky policies.

We prove these results by adapting a reduction of XC to concept (un)satisfiability in $\mathcal{ALE}$, extensively illustrated in (Donini 2003, Sec. 3.3.1). Let us first recall the definition of the problem:

Definition 3 (EXACT COVER, XC). *Given a finite set $U = \{u_1, \dots, u_n\}$ and a family $\mathcal{M} = \{M_1, \dots, M_m\}$ of subsets of U, decide whether there exist an* exact cover *of U, that is, a family of mutually disjoint sets $M_{i_1}, \dots, M_{i_q}$ whose union equals U.*

The following lemma introduces the reduction and states its correctness :

Lemma 4. (Donini 2003) *An instance of* XC *has an exact cover if, and only if, the concept $C_\mathcal{M}$ defined below is unsatisfiable:*

$$C_\mathcal{M} = C_1^1 \sqcap \dots \sqcap C_1^m \sqcap D$$

where each C_l^j is inductively defined as

$$C_{2n+1}^j = \top \quad \textit{(base case)}$$

$$C_l^j = \begin{cases} \exists R.C_{l+1}^j & \textit{either } l \le n \wedge u_l \in M_j \textit{ or} \\ & n < l \le 2n \wedge u_{l-n} \in M_j \\ \forall R.C_{l+1}^j & \textit{either } l \le n \wedge u_l \notin M_j \textit{ or} \\ & n < l \le 2n \wedge u_{l-n} \notin M_j \end{cases}$$

and $D = \underbrace{\forall R.\forall R.\dots\forall R}_{2n}.\bot$.

Example 5. Let $U = \{u_1, u_2\}$ and $\mathcal{M} = \{M_1, M_2\}$, where $M_1 = \{u_1\}$ and $M_2 = \{u_2\}$. Concept $C_\mathcal{M}$ is

$$\begin{array}{ll} \exists R.\forall R.\exists R.\forall R.\top \sqcap & (C_1^1) \\ \forall R.\exists R.\forall R.\exists R.\top \sqcap & (C_1^2) \\ \forall R.\forall R.\forall R.\forall R.\bot & (D) \end{array}$$

Note that this instance of XC has an exact cover ($\mathcal{M}$ itself) and $C_\mathcal{M}$ is indeed inconsistent. □

Lemma 4 was proved by showing that an exact cover exists iff the tableaux for $C_\mathcal{M}$ has a clash, caused by a node labelled with both $\top$ and $\bot$, where $\top$ has been introduced by some of the concepts C_1^j and $\bot$ has been introduced by D. Clearly, the same result can be obtained by replacing $\top$ (that is not supported in $\mathcal{PL}$) with a concept name A. So, from Lemma 4, we get:

Corollary 6. *Let $C_\mathcal{M}^A$ be the concept resulting from $C_\mathcal{M}$ by replacing each occurrence of $\top$ with concept name A. An instance of* XC *has an exact cover if, and only if, the concept $C_\mathcal{M}^A$ is unsatisfiable.*

It follows that subsumption is intractable in the extension of $\mathcal{PL}_0$ with $\forall$, that will be denoted by $\mathcal{PL}_0^\forall$.

Corollary 7. *Subsumption checking in $\mathcal{PL}_0^\forall$ is NP-hard, even if the knowledge base is empty.*

Proof. The exact cover problem is reduced to subsumption as follows: let A and B two distinct concept names. For a given instance of XC, the corresponding concept $C_\mathcal{M}^A$ and B have no symbols in common, therefore $C_\mathcal{M}^A \sqsubseteq B$ is valid iff $C_\mathcal{M}^A$ is unsatisfiable. Then this corollary immediately follows from Corollary 6. □

As a last attempt to restore tractability, one may consider another extension of $\mathcal{PL}_0$ where the problematic quantifier $\forall$ can be used only in conjunction with transitive role closure (as required by sticky policy modeling), and viceversa. In other words, expressions like $\forall R^+.C$ and $\exists R.C$ are permitted, while $\forall R.C$ and $\exists R^+.C$ are disallowed. This logic will be denoted with $\mathcal{PL}_0^{\forall+}$. This attempt is motivated by the observation that if $\forall R$ were replaced with $\forall R^+$ in $C_\mathcal{M}^A$, then the resulting concept would not capture exact covers anymore.

Example 8. Let $U = \{u_1, u_2, u_3\}$ and $\mathcal{M} = \{M_1, M_2\}$, where $M_1 = \{u_1\}$ and $M_2 = \{u_2\}$. The replacement of $\forall R$ with $\forall R^+$ in $C_\mathcal{M}^A$ yields:

$$\begin{array}{l} \exists R.\forall R^+.\forall R^+.\exists R.\forall R^+.\forall R^+.A \sqcap \\ \forall R^+.\exists R.\forall R^+.\forall R^+.\exists R.\forall R^+.A \sqcap \\ \forall R^+.\forall R^+.\forall R^+.\forall R^+.\forall R^+.\forall R^+.\bot \end{array}$$

The concepts in the first two lines create an infinite sequence of R-successors that clashes with the concept in the third line. So the above concept is inconsistent although this instance of XC has no exact covers. □

Unfortunately, the restriction on $\forall$ and $\cdot^+$ does not yield a tractable logic, either. First note that the reduction reported in Lemma 4 works equally well if different roles are used at each level, as in the following example.

Example 9. The concept $C_\mathcal{M}$ illustrated in Example 5 could be equivalently replaced with

$$\begin{array}{l} \exists R_1.\forall R_2.\exists R_3.\forall R_4.\top \sqcap \\ \forall R_1.\exists R_2.\forall R_3.\exists R_4.\top \sqcap \\ \forall R_1.\forall R_2.\forall R_3.\forall R_4.\bot\,. \end{array}$$

□

This version preserves the correspondence with XC stated in Lemma 4, because the tableaux produced by the two reductions have the same structure and differ only in the role names. Now each $\forall R_i$ can be equivalently replaced with $\forall R_i^+$; more precisely, it is easy to verify that also this second change preserves the structure of the tableaux for $C_\mathcal{M}$, and that the only difference is that $\forall R_i$ is replaced by $\forall R_i^+$ in node labels.

As a consequence of the above discussion, subsumption checking in $\mathcal{PL}_0^{\forall+}$ is intractable due to the following reduction from XC:

Lemma 10. *An instance of* XC *has an exact cover if, and only if, the concept $C'^A_\mathcal{M}$ defined below is unsatisfiable:*

$$C'^A_\mathcal{M} = C'^1_1 \sqcap \dots \sqcap C'^m_1 \sqcap D'$$

where each C'^j_l is inductively defined as

$$C'^j_{2n+1} = A \quad \textit{(base case)}$$

$$C'^j_l = \begin{cases} \exists R_l.C'^j_{l+1} & \textit{either } l \le n \wedge u_l \in M_j \textit{ or} \\ & n < l \le 2n \wedge u_{l-n} \in M_j \\ \forall R_l^+.C'^j_{l+1} & \textit{either } l \le n \wedge u_l \notin M_j \textit{ or} \\ & n < l \le 2n \wedge u_{l-n} \notin M_j \end{cases}$$

and $D' = \forall R_1^+.\forall R_2^+.\dots.\forall R_{2n}^+.\bot$.

The lower complexity bound (next theorem) can be proved by analogy with the proof of Corollary 7, using $C'^{A}_{\mathcal{M}}$ and Lemma 10 in place of $C^{A}_{\mathcal{M}}$ and Corollary 6.

Theorem 11. *Subsumption checking in $\mathcal{PL}_0^{\forall+}$ is NP-hard.*

5 Towards A Tractable Approach

Here we introduce a language tailored to the encoding of sticky policies. Such a specialized approach is motivated by the intractability results proved in the previous sections. The complexity caused by the interactions between ν and $\exists$ identified in section 3 will be avoided by restricting the language so as to allow only simple, linear recursions whose cyclic behavior has period 1.

Definition 12 (Pol,$\mathcal{PL}_0^{\mathsf{sticky}}$)**.** *Let* Pol *be the least language containing:*

- *all the simple concepts of $\mathcal{PL}_0$ with no occurrence of* sticky *(called* sticky-free *concepts);*
- *all the concepts $C \sqcap \exists\mathsf{sticky}.D$ such that C is a* sticky*-free $\mathcal{PL}_0$ concept and D a ν-free concept in* Pol*;*
- *all the concepts $C \sqcap \nu X.\exists(\mathsf{sticky}.D \sqcap X)$ such that C and D are* sticky*-free $\mathcal{PL}_0$ concepts.*

$\mathcal{PL}_0^{\mathsf{sticky}}$ *knowledge bases are $\mathcal{PL}$ knowledge bases containing only axioms from Table 2, and at least the axiom* func(sticky)*, that must also be the unique axiom where* sticky *occurs. $\mathcal{PL}_0^{\mathsf{sticky}}$ subsumption queries are expressions $C \sqsubseteq D$ where C and D are in* Pol*.*

Example 13. The following policies are in Pol and can be used in $\mathcal{PL}_0^{\mathsf{sticky}}$ subsumption queries:

$$
\begin{array}{l}
\exists\mathsf{has\_data}.\mathsf{Contact} \sqcap \\
\exists\mathsf{has\_processing}.\mathsf{Transfer} \sqcap \\
\exists\mathsf{has\_purpose}.\mathsf{Marketing} \sqcap \\
\exists\mathsf{sticky}.(\\
\quad \exists\mathsf{has\_data}.\mathsf{Contact} \sqcap \\
\quad \exists\mathsf{has\_processing}.\mathsf{Transfer} \sqcap \\
\quad \exists\mathsf{has\_purpose}.\mathsf{Marketing} \sqcap \\
\quad \exists\mathsf{sticky}.(\\
\qquad \exists\mathsf{has\_data}.\mathsf{Contact} \sqcap \\
\qquad \exists\mathsf{has\_processing}.\mathsf{DirectUse} \sqcap \\
\qquad \exists\mathsf{has\_purpose}.\mathsf{Marketing}))\,.
\end{array}
$$

$$
\begin{array}{l}
\exists\mathsf{has\_data}.\mathsf{Contact} \sqcap \\
\exists\mathsf{has\_processing}.\mathsf{Transfer} \sqcap \\
\exists\mathsf{has\_purpose}.\mathsf{Marketing} \sqcap \\
\nu X.\exists\mathsf{sticky}.(\\
\quad \exists\mathsf{has\_data}.\mathsf{Contact} \sqcap \\
\quad \exists\mathsf{has\_processing}.\mathsf{SomeProcessing} \sqcap \\
\quad \exists\mathsf{has\_purpose}.\mathsf{Marketing} \sqcap X)\,.
\end{array}
$$
□

$\mathcal{PL}_0^{\mathsf{sticky}}$ subsumption checking can be reduced to the same problem in $\mathcal{PL}_0$. To see this, we first prove a lemma that considers the four possible cases in which the given $\mathcal{PL}_0^{\mathsf{sticky}}$ subsumption contains the fixpoint operator ν (the other $\mathcal{PL}_0^{\mathsf{sticky}}$ subsumptions are just classic $\mathcal{PL}_0$ subsumptions). Each case is reduced to a small number of $\mathcal{PL}_0$ subsumptions.

Lemma 14. *Let $\mathcal{KB}$ be a $\mathcal{PL}_0^{\mathsf{sticky}}$ knowledge base, and let C, D, E, and F be* sticky*-free* Pol *concepts.*

1. *$\mathcal{KB} \models C \sqcap \nu X.\exists\mathsf{sticky}.(D \sqcap X) \sqsubseteq E \sqcap \nu X.\exists\mathsf{sticky}.(F \sqcap X)$ iff some of the following conditions hold:*
   - *$\mathcal{KB} \models C \sqsubseteq \bot$,*
   - *$\mathcal{KB} \models D \sqsubseteq \bot$,*
   - *both $\mathcal{KB} \models C \sqsubseteq E$ and $\mathcal{KB} \models D \sqsubseteq F$ hold;*
2. *if G is ν-free, then $\mathcal{KB} \models G \sqsubseteq C \sqcap \nu X.\exists\mathsf{sticky}.(D \sqcap X)$ iff $\mathcal{KB} \models G \sqsubseteq \bot$;*
3. *if E is both ν-free and* sticky*-free, then $\mathcal{KB} \models C \sqcap \nu X.\exists\mathsf{sticky}.(D \sqcap X) \sqsubseteq E$ iff either $\mathcal{KB} \models C \sqsubseteq E$ or $\mathcal{KB} \models D \sqsubseteq \bot$;*
4. *if G is ν-free but not* sticky*-free, that is,*
   $$G = G_0 \sqcap \exists\mathsf{sticky}.(G_1 \sqcap \exists\mathsf{sticky}.(\ldots\exists\mathsf{sticky}.G_n\ldots)) \quad (2)$$
   (where $G_0, \ldots, G_n$ are sticky*-free $\mathcal{PL}_0$ concepts), then $\mathcal{KB} \models C \sqcap \nu X.\exists\mathsf{sticky}.(D \sqcap X) \sqsubseteq G$ iff some of the following conditions hold:*
   - *$\mathcal{KB} \models C \sqsubseteq \bot$,*
   - *$\mathcal{KB} \models D \sqsubseteq \bot$,*
   - *both $\mathcal{KB} \models C \sqsubseteq G_0$ and $\mathcal{KB} \models D \sqsubseteq G_1 \sqcap \ldots \sqcap G_n$ hold;*

Proof. We start by proving statement 2. Since G is ν-free, it belongs to $\mathcal{PL}_0$, therefore – if consistent w.r.t. $\mathcal{KB}$– it has a finite tree-shaped model that satisfies $\mathcal{KB}$ (Bonatti et al. 2020). In such models, the concept on the right-hand side (that has only infinite models) is empty, so the subsumption is false. It follows easily that the subsumption holds iff $\mathcal{KB} \models G \sqsubseteq \bot$. This proves 2.

Proof of 3, "if". If $\mathcal{KB} \models C \sqsubseteq E$, then the subsumption holds because its left-hand side is subsumed by C and $\sqsubseteq$ is transitive. If $\mathcal{KB} \models D \sqsubseteq \bot$, then also $\mathcal{KB} \models \nu X.\exists\mathsf{sticky}.(D \sqcap X) \sqsubseteq \bot$, so the subsumption holds because its left-hand side is subsumed by $\bot$.

Proof of 3, "only if". By contraposition, assume that $\mathcal{KB} \not\models C \sqsubseteq E$ and $\mathcal{KB} \not\models D \sqsubseteq \bot$. Then there exist two pointed interpretations $(\mathcal{I}_1, d_1)$ and $(\mathcal{I}_2, d_2)$, that are models of $\mathcal{KB}$, and satisfy the sticky-free concepts $C \sqcap \neg E$ and D, respectively. We may assume w.l.o.g. that $\mathcal{I}_1$ and $\mathcal{I}_2$ are disjoint. Define an interpretation $\mathcal{J}$ as the union of $\mathcal{I}_1$ and $\mathcal{I}_2$ extended with the following definition of sticky: $\mathsf{sticky}^{\mathcal{J}} = \{(d_1, d_2), (d_2, d_2)\}$. By construction, $\mathcal{J}$ satisfies $\mathcal{KB}$: indeed, the union of $\mathcal{I}_1$ and $\mathcal{I}_2$ satisfies $\mathcal{KB}$ by the disjoint model union property (Bonatti et al. 2020), and the definition of sticky satisfies func(sticky), that is the unique axiom involving sticky in $\mathcal{KB}$. Moreover, by construction, $(\mathcal{J}, d_1)$ satisfies the left-hand side of the subsumption but not E. Then $\mathcal{J}$ witnesses that the subsumption does not hold.

Proof of 1. The "if" part is straightforward and left to the reader. The "only if" part is proved similarly to the previous case. More precisely, by contraposition, assume that $\mathcal{KB} \not\models C \sqsubseteq \bot$, $\mathcal{KB} \not\models D \sqsubseteq \bot$, and either $\mathcal{KB} \not\models C \sqsubseteq E$ or $\mathcal{KB} \not\models D \sqsubseteq F$ holds. In the former case, construct a counterexample $(\mathcal{J}, d_1)$ to the subsumption by composing two models of $C \sqcap \neg E$ and D, respectively, as shown in the proof of statement 3. Similarly, in the latter case, construct a counterexample $(\mathcal{J}, d_1)$ to the subsumption by composing two models of C and $D \sqcap \neg F$, respectively.

Proof of 4, "if". Let FP denote the fixpoint expression in the left-hand side of the subsumption, and note that the left-hand side is equivalent to:

$$C \sqcap \underbrace{\exists \mathsf{sticky}.(D \sqcap \exists \mathsf{sticky}.(\ldots \exists \mathsf{sticky}.(D}_{n \text{times}} \sqcap FP) \ldots))$$

This makes the left-hand side directly comparable with (2). Now the "if" part is straightforward and left to the reader.

Proof of 4, "only if". By contraposition, assume that $\mathcal{KB} \not\models C \sqsubseteq \bot$, $\mathcal{KB} \not\models D \sqsubseteq \bot$, and either $\mathcal{KB} \not\models C \sqsubseteq G_0$ or $\mathcal{KB} \not\models D \sqsubseteq G_1 \sqcap \ldots \sqcap G_n$ holds. In the former case, obtain a counterexample to the subsumption by composing two models of $C \sqcap \neg G_0$ and D, respectively, as shown in the proof of 3. In the latter case, there exists i $(1 \leq i \leq n)$ such that $\mathcal{KB} \not\models D \sqsubseteq D_i$. Then obtain a counterexample to the subsumption by composing two models of C and $D \sqcap \neg G_i$, respectively, using the same approach as in 3. □

As a consequence of the above lemma, we get:

Theorem 15. *Subsumption checking in $\mathcal{PL}_0^{\mathsf{sticky}}$ can be done in polynomial time.*

Proof. $\mathcal{PL}_0^{\mathsf{sticky}}$ subsumption checking can always be reduced to subsumption checking in $\mathcal{PL}_0$. In particular, for all given $\mathcal{PL}_0^{\mathsf{sticky}}$ subsumptions $C \sqsubseteq D$, either C and D are ν-free (so $C \sqsubseteq D$ is a $\mathcal{PL}_0$ subsumption), or $C \sqsubseteq D$ falls in one of the four cases covered by Theorem 14. For each of these cases, Theorem 14 provides an equivalent set of subsumption checks that involve $\mathcal{PL}_0$ concepts only (as ν-free $\mathcal{PL}_0^{\mathsf{sticky}}$ concepts are $\mathcal{PL}_0$ concepts). Since subsumption checking in $\mathcal{PL}_0$ is in PTIME, the same holds for $\mathcal{PL}_0^{\mathsf{sticky}}$. □

Note that, in the worst case, a subsumption check in $\mathcal{PL}_0^{\mathsf{sticky}}$ is reduced to four subsumption checks in $\mathcal{PL}_0$, therefore $\mathcal{PL}_0^{\mathsf{sticky}}$ preserves the asymptotic complexity of compliance checking in $\mathcal{PL}_0$.

6 Discussion and Related Work

The approach to sticky policies outlined in the previous section is only a first step towards a complete tractable solution. The language $\mathcal{PL}_0^{\mathsf{sticky}}$ does not yet support some of the features of $\mathcal{PL}$, such as intervals and unions. These two features are related, because the normalization of intervals that makes $\mathcal{PL}$ subsumption queries interval-safe (hence tractable) introduces unions. Some of these unions may occur within the range of ν, and it is not possible – in general – to move them to the top level to reduce general policies to mere unions of Pol concepts. Therefore, it is necessary to investigate the interplay of fixpoints and unions, and its potential impact on complexity. We conjecture that – with a careful definition of $\mathcal{PL}^{\mathsf{sticky}}$ – interval safety can still be obtained efficiently (given the natural restrictions satified by policies), and that the proof of Theorem 14 can be generalized to prove that tractability is preserved also in the presence of intervals and unions. Of course, the theoretical complexity analysis shall be complemented by an experimental performance evaluation to see if the ad hoc framework meets the scalability requirements of use cases.

Another limitation of the current version of $\mathcal{PL}_0^{\mathsf{sticky}}$ is that a single consent policy cannot permit both finite and infinite data transfer chains at the same time (cf. Theorem 14, point 2), while in some cases it may be useful to state that any number of transfers is permitted. Thus, $\mathcal{PL}_0^{\mathsf{sticky}}$ should be extended to support classes that may contain both finite and infinite chains, such as:

$$\nu X.\big((\exists \mathsf{sticky}.(C \sqcap X)) \sqcup D\big)\,,$$

where C is a policy that admits transfers and D a policy that does not admit transfers. This provides an independent motivation for introducing unions within the scope of ν.

It is not easy to turn the lower complexity bounds presented in this paper into exact characterizations. On the one hand, the extensions of $\mathcal{PL}$ that we considered make it possible to axiomatize complex, exponentially large structures, like those used in some proofs of PSPACE-hardness; on the other hand, the limited expressiveness of $\mathcal{PL}$ (and in particular the restrictions on union and negation) makes it difficult to use those structures to reconstruct the complexity results that have been proved for DLs that (unlike $\mathcal{PL}$) support a full set of boolean concept operators. An additional difficulty stems from the fact that, in $\mathcal{PL}$'s extensions, fixpoints and transitive role closures may occur only in the queries and cannot be used inside axioms.

The logic $\mathcal{PL}_0^{\forall}$ investigated in Section 4 is similar to $\mathcal{FL}_0$, in some respects. It has been proved in (Nebel 1990) that subsumption checking in $\mathcal{FL}_0$ is coNP-complete for acyclic TBoxes.

Fixpoints have been extensively investigated in the context of description logics, see for example (Calvanese, Giacomo, and Lenzerini 1999; Bonatti and Peron 2004; Bonatti et al. 2008; Lutz, Piro, and Wolter 2010; Franconi and Toman 2011). Most of these papers deal with expressive logics whose reasoning tasks are at least EXPTIME-hard (sometimes even undecidable). The only exception is (Lutz, Piro, and Wolter 2010), that proves the tractability of $\mathcal{EL}$ with greatest fixpoints. This logic and $\mathcal{PL}_0^{\nu}$ have several traits in common. The intractability of $\mathcal{PL}_0^{\nu}$ is due to the interplay of functionality axioms (not supported by $\mathcal{EL}$) with the fixpoints occurring in the queries.

The seminal work that started the investigation of transitive role closures is (Sattler 1996). One of its results is the proof that extending $\mathcal{ALC}$ with transitive role closure makes concept satisfiability EXPTIME-complete. Subsumption checking has the same complexity because, in $\mathcal{ALC}$ and its extensions, concept satisfiability and subsumption checking are mutually reducible to each other. The complexity of transitive role closure in $\mathcal{EL}$ has been studied in (Haase and Lutz 2008). It is proved that subsumption checking is complete for coNP, PSPACE, and EXPTIME if TBoxes are empty, acyclic, and cyclic, respectively.

It may be interesting to investigate whether regular path queries can be used to model sticky policies. A rich set of results on nested regular path queries and low-complexity DLs can be found in (Bienvenu et al. 2014). However, this paper deals with query evaluation, while the counterpart of compliance checking (i.e. subsumption checking) is query containment.

We conclude by pointing out that the same approach adopted for data usage policies works out of the box for *licenses in financial data markets*, an area that has similar requirements (i.e. support to data transfers, and need for massive and reliable compliance checking, in order to reduce the risks of sanctions due to license violations). Moreover, the "Share Alike" Creative Commons licence has a transitive nature, much like sticky policies. These observations open up a brand new range of applications for $\mathcal{PL}$ and its extensions.

Acknowledgments

This work is funded by the European Union with grant n. 883464. The autors are grateful to the anonymous reviewers for their insightful and constructive comments.

References

Baader, F.; Calvanese, D.; McGuinness, D. L.; Nardi, D.; and Patel-Schneider, P. F., eds. 2003. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press.

Bienvenu, M.; Calvanese, D.; Ortiz, M.; and Simkus, M. 2014. Nested regular path queries in description logics. In Baral, C.; Giacomo, G. D.; and Eiter, T., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourteenth International Conference, KR 2014, Vienna, Austria, July 20-24, 2014*. AAAI Press.

Bonatti, P. A., and Peron, A. 2004. On the undecidability of logics with converse, nominals, recursion and counting. *Artif. Intell.* 158(1):75–96.

Bonatti, P. A.; Lutz, C.; Murano, A.; and Vardi, M. Y. 2008. The complexity of enriched mu-calculi. *Log. Methods Comput. Sci.* 4(3).

Bonatti, P. A.; Ioffredo, L.; Petrova, I. M.; Sauro, L.; and Siahaan, I. S. R. 2020. Real-time reasoning in OWL2 for GDPR compliance. *Artif. Intell.* 289:103389.

Bonatti, P. A.; De Capitani di Vimercati, S.; and Samarati, P. 2002. An algebra for composing access control policies. *ACM Trans. Inf. Syst. Secur.* 5(1):1–35.

Bonatti, P. A.; Sauro, L.; and Langens, J. 2021. Representing consent and policies for compliance. In *IEEE European Symposium on Security and Privacy Workshops, EuroS&P 2021, Vienna, Austria, September 6-10, 2021*, 283–291. IEEE.

Calvanese, D.; Giacomo, G. D.; and Lenzerini, M. 1999. Reasoning in expressive description logics with fixpoints based on automata on infinite trees. In Dean, T., ed., *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI 99, Stockholm, Sweden, July 31 - August 6, 1999. 2 Volumes, 1450 pages*, 84–89. Morgan Kaufmann.

Donini, F. M. 2003. Complexity of reasoning. In Baader et al. (2003). 96–136.

Franconi, E., and Toman, D. 2011. Fixpoints in temporal description logics. In Walsh, T., ed., *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, 875–880. IJCAI/AAAI.

Haase, C., and Lutz, C. 2008. Complexity of subsumption in the family of description logics: Acyclic and cyclic tboxes. In Ghallab, M.; Spyropoulos, C. D.; Fakotakis, N.; and Avouris, N. M., eds., *ECAI 2008 - 18th European Conference on Artificial Intelligence, Patras, Greece, July 21-25, 2008, Proceedings*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, 25–29. IOS Press.

Lutz, C.; Piro, R.; and Wolter, F. 2010. Enriching $\mathcal{EL}$-concepts with greatest fixpoints. In Coelho, H.; Studer, R.; and Wooldridge, M. J., eds., *ECAI 2010 - 19th European Conference on Artificial Intelligence, Lisbon, Portugal, August 16-20, 2010, Proceedings*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, 41–46. IOS Press.

Nebel, B. 1990. Terminological reasoning is inherently intractable. *Artif. Intell.* 43(2):235–249.

Rosser, B. 1941. Explicit bounds for some functions of prime numbers. *American Journal of Mathematics* 63(1):211–232.

Sattler, U. 1996. A concept language extended with different kinds of transitive roles. In Görz, G., and Hölldobler, S., eds., *KI-96: Advances in Artificial Intelligence, 20th Annual German Conference on Artificial Intelligence, Dresden, Germany, September 17-19, 1996, Proceedings*, volume 1137 of *Lecture Notes in Computer Science*, 333–345. Springer.