Tractable Compliance Checking with Negation[⋆]

Piero A. Bonatti[1][0000−0003−1436−5660] and Luigi Sauro[1][0000−0001−6056−0868]

Università degli Studi di Napoli "Federico II"
{pab,luigi.sauro}@unina.it

Abstract. $\mathcal{PL}$ is a low-complexity fragment of OWL2, expressly designed to automatically check the compliance of privacy policies with respect to the consent of data subjects and the regulations on personal data processing, such as the GDPR. Privacy policies, consent and regulations can all be expressed uniformly with $\mathcal{PL}$ classes, and compliance checking is reduced to subsumption checking. The latter is tractable and highly scalable to meet the performance requirements of some applications. Several use cases need negation in the queries; however, subsumption checking becomes coNP-hard, even if negation is restricted to atomic concepts. In this paper we show that an alternative encoding based on histories of positive and negative policies (where negation is applied to compound concepts) can be processed in polynomial time by a correct and complete algorithm.

Keywords: GDPR compliance checking · Tractable DL · Negation.

1 Introduction

$\mathcal{PL}$ is a recently introduced fragment of $\mathcal{SROIQ}(D)$ that is being successfully used to encode privacy policies, the consent of data subjects, and objective parts of the GDPR [4, 5]. It is also being used to encode licenses formulated with a profile of ODRL[1] and with JSON; well-defined translations from these formats to $\mathcal{PL}$ provide the formal semantics of licenses. The main goal of $\mathcal{PL}$ is enabling automated compliance checks for data economies. Many companies are interested in such tools, that help in preventing sanctions and reputation loss.

$\mathcal{PL}$ supports simple ontologies that define privacy-related and licensing-related vocabularies.[2] Policies, consent, regulations, and licenses are encoded as classes; compliance checking is reduced to subsumption checking over such classes.

$\mathcal{PL}$ balances expressiveness with usability and scalability. In some use cases, it should be possible to execute a few thousands compliance checks per second;

⋆ This work is funded by the European Union with grant n. 883464. Copyright ©2021 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

1 https://w3c.github.io/market-data-odrl-profile/md-odrl-profile.html

2 Currently, the reference vocabularies for privacy are those defined by the DPVCG group of the W3C, see https://www.w3.org/community/dpvcg/.

experiments show that a specialized engine implemented in Java can check the compliance of typical privacy policies with respect to the available consent in about 400-500 μ-seconds [5]. Concerning usability, SPECIAL's use case partners have successfully verified that their employees can write correct privacy policies.

The use cases of the H2020 project SPECIAL[3] – that developed the first version of $\mathcal{PL}$ – and its ongoing follow-up TRAPEZE[4] call for extensions of $\mathcal{PL}$. Here we focus on negation, that is also needed in the licensing domain. Negation has three main applications in these contexts:

– expressing exceptions to a general policy/license; the resulting policies are more compact and readable;
– supporting the right of the data subjects to withdraw or restrict their consent to data processing;
– supporting *dynamic consent* [8].

In dynamic consent, users are asked for consent immediately before data are collected. Users may reply within a specified timeout, after which consent is implicitly assumed. Users may deny consent either within the timeout or later, by revising such implicit consent statements. In all cases, it is useful to formalize statements such as "*location can be tracked in Naples, but* not *in Via Roma*".

This paper investigates how to introduce $\mathcal{SROIQ}$'s complement operator '$\neg$' in $\mathcal{PL}$ without affecting its tractability and scalability. The main contributions are the following:

1. we prove that the simplest way of introducing negation in $\mathcal{PL}$ makes reasoning coNP-hard;
2. then we introduce a notion of *history* that addresses the above use-case requirements and can be processed in polynomial time.

The paper is organized as follows: In the next section, we recall the basic definitions and properties of $\mathcal{PL}$, illustrate it with some examples, and list related complexity results. Section 3 proves that $\mathcal{PL}$ with atomic negation in the queries is intractable. Section 4 introduces histories and proves that compliance checking based on histories is tractable. Section 5 reports our conclusions.

2 Preliminaries and Related Work

We assume the reader to be familiar with the syntax and the model-theoretic semantics of description logics, and refer to [3] for all definitions. As usual, if $\mathcal{I}$ is an interpretation, then we denote its domain and its interpretation function with $\Delta^{\mathcal{I}}$ and $\cdot^{\mathcal{I}}$, respectively.

The axioms supported by $\mathcal{PL}$ *knowledge bases* are those illustrated in Table 1. $\mathcal{PL}$ *queries* are inclusions $C \sqsubseteq D$ where C and D are *full* $\mathcal{PL}$ *concepts*, that is, unions of the form:

$$C_1 \sqcup \ldots \sqcup C_n$$

[3] https://specialprivacy.ercim.eu/
[4] https://trapeze-project.eu/

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

Table 1. Axioms for $\mathcal{PL}$ knowledge bases. Here A and B range over concept names while R is a role name.

where each C_i is a *simple* $\mathcal{PL}$ *concept*. Such concepts are defined by the following grammar:

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

where A is a concept name, R is a role name, F is an integer-valued concrete feature (i.e. a *data property*, in OWL2 terminology), and l, u are integers. The meaning of $\exists F.[l, u]$ is:

$$(\exists F.[l, u])^{\mathcal{I}} = \left\{ d \in \Delta^{\mathcal{I}} \mid \exists i \in \mathbb{Z}.(d, i) \in F^{\mathcal{I}} \wedge l \leq i \leq u \right\}$$

(recall that for all integer-valued concrete features F and all interpretations $\mathcal{I}$, $F^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \mathbb{Z}$).

2.1 Examples

A company – call it BeFit – sells a wearable fitness appliance and wants to: (i) process biometric data (stored in the EU) for sending health-related advice to its customers, and (ii) share the customer's location data with their friends. Location data are kept for a minimum of one year but no longer than 5; biometric data are kept for an unspecified amount of time. In order to carry out these two data processing activities legally, BeFit needs consent from its European customers. Formally, the encoding of BeFit's privacy policy P in $\mathcal{PL}$ looks as follows:

$$\begin{aligned}
&(\exists \mathsf{has\_purpose.FitnessRecommendation} \sqcap \exists \mathsf{has\_data.BiometricData} \sqcap \\
&\quad \exists \mathsf{has\_processing.Analytics} \sqcap \exists \mathsf{has\_recipient.BeFit} \sqcap \\
&\quad \exists \mathsf{has\_storage}.\exists \mathsf{has\_location.EU}) \\
&\sqcup \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad (1) \\
&(\exists \mathsf{has\_purpose.SocialNetworking} \sqcap \exists \mathsf{has\_data.LocationData} \sqcap \\
&\quad \exists \mathsf{has\_processing.Transfer} \sqcap \exists \mathsf{has\_recipient.DataSubjFriends} \sqcap \\
&\quad \exists \mathsf{has\_storage}.\exists \mathsf{has\_duration}.[y_1, y_5])\,.
\end{aligned}$$

Note that P is the union of two sub-concepts that formalize operations (i) and (ii). Here y_1 and y_5 stand for the integer representation of one year and five years, respectively.

A consent policy C, that specifies the class of operations that a data subject *permits*, is encoded in a similar fashion. Then, P complies with C iff $\mathcal{K} \models P \sqsubseteq C$, where $\mathcal{K}$ is the knowledge base that defines the terms used in the policies. For example, the consent policy of a user that opts out (ii), and opts in (i) only for anonimized data, can be encoded as follows:

$$\begin{aligned}(&\exists \mathsf{has\_purpose}.\mathsf{FitnessRecommendation} \sqcap \exists \mathsf{has\_data}.\mathsf{BiometricData} \sqcap \\ &\exists \mathsf{has\_processing}.\mathsf{Analytics} \sqcap \exists \mathsf{has\_recipient}.\mathsf{BeFit} \sqcap \\ &\exists \mathsf{has\_storage}.\exists \mathsf{has\_location}.\mathsf{EU})\,.\end{aligned} \tag{2}$$

Clearly, BeFit's privacy policy P complies with this consent policy. As a second example, suppose that the data subject allows analytics on anonimized biometric data. This policy can be formalized as follows:

$$\begin{aligned}(&\exists \mathsf{has\_processing}.\mathsf{Analytics} \sqcap \\ &\exists \mathsf{has\_data}.(\mathsf{BiometricData} \sqcap \mathsf{Anonymized}))\,.\end{aligned} \tag{3}$$

In this case, P does not comply with the consent policy.[5]

The knowledge base $\mathcal{K}$ typically contains both domain-specific axioms and domain-independent axioms, that represent general properties of the business policies. The latter axioms specify, for instance, that the primary roles $\mathsf{has\_purpose}$, $\mathsf{has\_data}$, $\mathsf{has\_processing}$, and $\mathsf{has\_storage}$ are functional, and range over suitable, pairwise-disjoint concepts:

$$\begin{aligned}&\mathsf{range}(\mathsf{has\_purpose}, \mathsf{AnyPurpose})\\ &\mathsf{range}(\mathsf{has\_data}, \mathsf{AnyData})\\ &\mathsf{disj}(\mathsf{AnyPurpose}, \mathsf{AnyData})\\ &\dots\end{aligned}$$

Domain-specific axioms can be used to instantiate the compliance framework in a particular application domain. For example, in our scenario, domain-specific data categories and processing can be introduced with inclusions like:

$$\begin{aligned}\mathsf{HeartRate} &\sqsubseteq \mathsf{BiometricData}\,,\\ \mathsf{ComputeAvg} &\sqsubseteq \mathsf{Analytics}\,.\end{aligned}$$

These two axioms, together with the above consent, allow BeFit to compute the average heart rate of the data subject in order to send her fitness recommendations. Other typical axioms specify which classes are mutually disjoint, as in:

$$\mathsf{disj}(\mathsf{BiometricData}, \mathsf{LocationData})\,.$$

Note that negation is not supported in this language, so there is no explicit way of denying consent, nor is it possible to introduce exceptions to a general consent. Two alternative ways of introducing negation in $\mathcal{PL}$'s representation of policies will be discussed in sections 3 and 4.

[5] We are assuming that $\mathcal{K}$ does not entail $\mathsf{BiometricData} \sqsubseteq \mathsf{Anonymized}$.

2.2 Interval Safety and Convexity

In this section we report some definitions and results from [4] that will be useful in Section 4.

Definition 1. *A $\mathcal{PL}$ query $C \sqsubseteq D$ is* interval safe *if, for all constraints $\exists f.[l,u]$ occurring in C and all $\exists f.[l',u']$ occurring in D, either $[l,u] \subseteq [l',u']$ or $[l,u] \cap [l',u'] = \emptyset$.*

Every $\mathcal{PL}$ query $C \sqsubseteq D$ is equivalent to an interval safe inclusion $C^* \sqsubseteq D$, where $\models C^* \equiv C$ [4, Prop. 5]. Informally speaking, when subsumptions are interval safe, then they behave as in a convex logic. Formally, interval safe queries enjoy the following property, that follows from lemmas 1 and 2 of [4], combined with Propositions 1 of the same paper:

Lemma 1. *Let $\mathcal{K}$ be a $\mathcal{PL}$ knowledge base and let C be a simple $\mathcal{PL}$ concept such that $\mathcal{K} \not\models C \sqsubseteq \bot$. There exist an interpretation $\mathcal{I}$ and an individual $d \in \Delta^{\mathcal{I}}$ such that:*

a) $\mathcal{I} \models \mathcal{K}$;
b) $d \in C^{\mathcal{I}}$;
c) for all interval safe $\mathcal{PL}$ queries $C \sqsubseteq D$, $d \in D^{\mathcal{I}}$ iff $\mathcal{K} \models C \sqsubseteq D$.

We will call such pairs $(\mathcal{I}, d)$ *canonical models of $\mathcal{K}$ and C*. The above lemma implies the following *convexity property*:

Proposition 1. *for all $\mathcal{PL}$ knowledge bases $\mathcal{K}$, and all interval safe $\mathcal{PL}$ queries $q = (C \sqsubseteq D_1 \sqcup \ldots \sqcup D_n)$ where C is simple, $\mathcal{K} \models q$ iff there exists i such that $\mathcal{K} \models C \sqsubseteq D_i$ $(1 \le i \le n)$.*

2.3 Complexity Results and Related Work

Deciding whether a full $\mathcal{PL}$ concept is inconsistent with respect to a $\mathcal{PL}$ knowledge base is in P. Deciding whether a $\mathcal{PL}$ query is a logical consequence of a $\mathcal{PL}$ knowledge base is in general coNP-complete. It is in P if, for all simple $\mathcal{PL}$ concepts C_i occurring on the left-hand side of the subsumption query, the number of concrete features occurring in C_i is bounded by a constant c. In SPECIAL and TRAPEZE's privacy policies, $c = 1$. With this bound, a sequential Java implementation can reach over 2000 compliance checks per second, thereby addressing the scalability requirements posed by the use case partners of SPECIAL and TRAPEZE [5].

The above complexity results – that have been proved in [4] – have been extended in [5] to knowledge bases $\mathcal{K} \cup \mathcal{O}$ where $\mathcal{K}$ is a $\mathcal{PL}$ knowledge base and $\mathcal{O}$ is a Horn-$\mathcal{SRIQ}$ knowledge base, under the following assumptions:

1. $\mathcal{O}$ belongs to a tractable fragment of Horn-$\mathcal{SRIQ}$;
2. none of the roles occurring in $\mathcal{O}$ occurs in either $\mathcal{K}$ or the given $\mathcal{PL}$ query.[6]

[6] With suitable syntactic sugar, the roles defined in $\mathcal{O}$ may be used in the queries, provided that the roles defined in $\mathcal{K}$ do not occur within the scope of those defined in $\mathcal{O}$ [5, Remark 5.1].

Note that $\mathcal{PL}$ knowledge bases are in OWL2-RL, a Horn fragment of OWL2; they are also in the extensions of $\mathcal{EL}$ and $DL\text{-}lite^{\mathcal{H}}_{horn}$ with functional roles. Answering $\mathcal{PL}$ subsumption queries is equivalent to solving *query containment problems* with respect to $\mathcal{PL}$ knowledge bases, where the queries are the translation of full $\mathcal{PL}$ concepts into formulae of first-order logic. Such formulae are instances of the class of queries investigated in [9], called *extended faceted queries*. More precisely, the queries derived from full $\mathcal{PL}$ concepts are *unions of conjunctive queries with comparison operators* (in our case only '$\leq$', that is needed to express interval membership). In the following sections, such queries will be extended with negation.[7] The comparison operator makes the usual reduction of query containment to query answering impossible, so the literature on answering unions of conjunctive faceted queries is not directly applicable to our case. For more details – and further tractability and intractability results concerning faceted queries – see [5, Sec. 5.3].

In the extension of $\mathcal{EL}$ with functional roles, subsumption checking is complete for EXPTIME, in general [2]. A tractability result for empty TBoxes is reported in [6, Fig. 4]; however, in the same paper, it is proved that even with acyclic TBoxes, subsumption is coNP-complete. The tractability of an extension of $\mathcal{EL}$ with non-convex concrete domains (like intervals) has been proved in [6], under the assumption that the TBox is a set of *definitions* of the form $A \equiv C$, where each A is a concept name and appears in the left-hand side of at most one definition.

If $DL\text{-}lite^{\mathcal{H}}_{horn}$ is extended with functional roles, then the data complexity of query answering raises at the first level of the polynomial hierarchy and knowledge base satisfiability becomes EXPTIME-complete [1].

Finally, a result relevant to the extension of queries with negation discussed in the following sections. Rosati [7] proved that answering conjunctive queries with safe negation in *DL-lite* is coNP-hard. The proof makes use of a knowledge base that contains negation and inverse roles. In our setting, queries are intractable even when the knowledge base is empty.

3 Intractability of $\mathcal{PL}$ with Atomic Negation

With reference to the example illustrated in Section 2.1, assume that a BeFit customer wants to restrict her consent, by precluding that locations in her homeplace (say Rome) are shared with friends. A simple way of introducing this restriction in (1) consists in replacing the expression $\exists \mathsf{has\_data}.\mathsf{LocationData}$ with

$$\exists \mathsf{has\_data}.(\mathsf{LocationData} \sqcap \neg \mathsf{RomeLocationData})\,,$$

where the term $\mathsf{RomeLocationData}$ is axiomatized in the knowledge base $\mathcal{K}$ with the inclusion:

$$\mathsf{RomeLocationData} \sqsubseteq \mathsf{LocationData}\,.$$

[7] We will actually work on the subsumption problems that correspond to containment problems involving unions of conjunctive queries with '$\leq$' and negation.

This encoding extends $\mathcal{PL}$ queries with *atomic negation* (that is, negation applied to concept names). Our use cases do not need negation in $\mathcal{PL}$ knowledge bases.

Subsumption checking over full $\mathcal{PL}$ concepts with atomic negation can be reformulated as query containment over unions of conjunctive queries with safe negation. Containment of conjunctive queries with negation is Π_2^p-complete [10], so the intractability of $\mathcal{PL}$ with atomic negation in the queries does not come as a surprise.

Here we show that subsumption checking remains intractable (coNP-hard) even if atomic negation is used as in the above example verbatim, that is, in existential restrictions of the form

$$\exists R.(A \sqcap \neg B) \tag{4}$$

where A, B are concept names. The reduction is from the validity of 3-DNF formulae. Given such a propositional formula $\phi = \bigvee_{i=1}^{n}(L_{i,1} \wedge L_{i,2} \wedge L_{i,3})$, its reduction is:

$$\exists R.A \sqsubseteq \bigsqcup_{i=1}^{n} \big(\exists R.(A \sqcap L_{i,1}) \sqcap \exists R.(A \sqcap L_{i,2}) \sqcap \exists R.(A \sqcap L_{i,3})\big) \tag{5}$$

where R is any role name and A is a concept name that does not occur in any $L_{i,j}$. It is not hard to see that ϕ is valid iff the above inclusion is valid, and that the reduction can be computed in polynomial time (actually, logarithmic space), so the following theorem holds:

Theorem 1. *Subsumption checking in $\mathcal{PL}$ with atomic negation is* coNP*-hard even if the knowledge base is empty and negation is restricted to concepts of the form* (4)*.*

4 Histories

Our use cases can also be addressed by modeling the sequence of consent and denial statements issued by the data subject (either through the dynamic consent system or through the dashboards for personal data control). In formal terms, each such statement is a simple policy with a sign, that specifies whether the operations described by the policy are permitted or forbidden. In case of conflicts, older statements are overridden by more recent statements.

Definition 2 (History). *A* history *is a finite sequence $+C_1 \pm C_2 \ldots \pm C_n$ where each C_i is a simple $\mathcal{PL}$ concept.*

Histories are just a handy way of representing compound concepts with negation, as shown by the following inductive translation tr, where H is a history:

history	$tr(history)$
$+C$	C
$H+C$	$tr(H) \sqcup C$
$H-C$	$tr(H) \sqcap \neg C$

Hereafter, we will identify each history H with the corresponding concept $tr(H)$.

Example 1. Consider the example in Section 3 and let H be the history $+D_1 + D_2 - D_3$, where D_1 and D_2 are the two arguments of the union in (1) and D_3 is the concept:

$$\exists \mathsf{has\_data}.(\mathsf{RomeLocationData})\,.$$

By the distributivity of $\sqcap$ on $\sqcup$, H is equivalent to $(D_1 \sqcap \neg D_3) \sqcup (D_2 \sqcap \neg D_3)$. Note that $D_1 \sqcap \neg D_3$ is equivalent to D_1, because D_1 and D_3 are disjoint. This follows from three facts: (i) $\mathsf{LocationData}$ and $\mathsf{BiometricData}$ are disjoint, (ii) $\mathsf{RomeLocationData}$ is a subclass of $\mathsf{LocationData}$, and (iii) $\mathsf{has\_data}$ is functional. Concerning $D_2 \sqcap \neg D_3$, by (iii) we have that the concept

$$\exists \mathsf{has\_data}.(\mathsf{LocationData}) \sqcap \neg \exists \mathsf{has\_data}.(\mathsf{RomeLocationData})$$

is equivalent to

$$\exists \mathsf{has\_data}.(\mathsf{LocationData} \sqcap \neg \mathsf{RomeLocationData})\,. \tag{6}$$

Therefore, $D_2 \sqcap \neg D_3$ is equivalent to the concept D_2' obtained from D_2 by replacing $\exists \mathsf{has\_data}.(\mathsf{LocationData})$ with (6). Summarizing, H is equivalent to $D_1 \sqcup D_2'$, therefore it represents the restricted consent of the customer correctly. $\square$

We are going to prove that the entailments of the form

$$\mathcal{K} \models C \sqsubseteq H \tag{7}$$

where $\mathcal{K}$ is a $\mathcal{PL}$ knowledge base, C is a full $\mathcal{PL}$ concept, and H is a history, can be decided in polynomial time. For this purpose, we need a few preliminary definitions and lemmas. First, we normalize H with the following procedure[8]:

Definition 3 (Normalization $norm(H)$**).** *Let $H = +C_1 \pm C_2 \ldots \pm C_n$ be a history, $\mathcal{K}$ be a knowledge base, and C be a concept. The* normalization of H w.r.t. $\mathcal{K}$ and C, *denoted by $norm_{\mathcal{K},C}(H)$, is obtained with the following three steps:*

a) replace each $-C_i$ in H with $-(C_i \sqcap C)$; the result of this transformation will be denoted by $H \downarrow C$;

b) remove from $H \downarrow C$ all elements $\pm D$ such that $\mathcal{K} \models D \sqsubseteq \bot$;

c) remove from the resulting history all the elements $\pm D$ followed by some element $\pm D'$ such that $\mathcal{K} \models D \sqsubseteq D'$.

The subscripts $\mathcal{K}$ and C in $norm_{\mathcal{K},C}(H)$ will be omitted when clear from context.

The above normalization preserves the entailments of the form (7), thus it can be exploited to answer subsumption queries involving histories:

[8] Note that the normalization procedure in Definition 3 has been introduced only for the purpose of simplifying proofs. Hence, it will be not part of the actual entailment procedure described in Algorithm 1.

Lemma 2. *For all $\mathcal{PL}$ knowledge bases $\mathcal{K}$, full $\mathcal{PL}$ concepts C, and histories H,*

$$\mathcal{K} \models C \sqsubseteq H \textit{ iff } \mathcal{K} \models C \sqsubseteq \mathit{norm}(H)\,.$$

The proof of the above lemma is based on the next lemma:

Lemma 3. *Let $\mathcal{K}$ be a $\mathcal{PL}$ knowledge base, let C be a full $\mathcal{PL}$ concept, and let $H = +C_1 \pm \ldots \pm C_n$ be a history. Then the following propositions hold:*

a) $\models H \sqcap C \equiv (H \downarrow C) \sqcap C$ *;*

b) if $\mathcal{K} \models D \sqsubseteq \bot$ *then* $\mathcal{K} \models H{\pm}D \equiv H$ *;*

c) if $H = H_1{\pm}C_i H_2{\pm}C_n$ *and* $\mathcal{K} \models C_i \sqsubseteq C_n$*, then* $\mathcal{K} \models H \equiv H_1 H_2{\pm}C_n$*.*

Proof. (Sketch) We prove Lemma 3.a) by induction on the length of H. In the base case, $H = +D$, so $H \sqcap C$ and $(H \downarrow C) \sqcap C$ are syntactically the same.

Let $H = H'{\pm}D$. If D occurs positively, we have by distributivity that $H{\sqcap}C$ is equivalent to $(H'{\sqcap}C){+}(D{\sqcap}C)$. Moreover, by construction $H \downarrow C = (H' \downarrow C){+}D$ and hence $(H \downarrow C) \sqcap C$ is equivalent to $((H' \downarrow C) \sqcap C) + (D \sqcap C)$. Then, the assertion follows from the fact that, by (IH), $H' \sqcap C$ and $(H' \downarrow C) \sqcap C$ are equivalent. An analogous proof for the negative case is left to reader.

Lemma 3.b) is trivial, so we focus on point c). If C_i and C_n occur positively in Lemma 3.c) then $H_1 H_2{+}C_n$ is always subsumed by $H_1{+}C_i H_2{+}C_n$. The other way round directly follows from the fact that $H_1{+}C_i H_2$ is subsumed by $H_1 H_2{+}C_i$ and $\mathcal{K} \models C_i \sqsubseteq C_n$. Analogously, we can prove the negative case by inverting the directions of subsumptions. Next, assume that C_i is negative and C_n positive. Since $\mathcal{K} \models C_i \sqsubseteq C_n$, the difference between the extensions of $H_1 - C_i H_2$ and $H_1 H_2$ is always contained in the extension of C_n. It follows that $H_1 - C_i H_2 + C_n \equiv H_1 H_2 + C_n$, so point c) holds. The case in which C_i is positive and C_n negative is similar. There are no further cases, so the proof is complete. □

Proof of Lemma 2. (Sketch) It suffices to show that each of the three normalization steps preserves (7). First observe that:

$$\begin{aligned} \mathcal{K} \models C \sqsubseteq H \text{ iff } & \mathcal{K} \models C \sqsubseteq H \sqcap C \\ \text{iff } & \mathcal{K} \models C \sqsubseteq (H \downarrow C) \sqcap C \text{ (by Lemma 3.a)} \\ \text{iff } & \mathcal{K} \models C \sqsubseteq H \downarrow C\,, \end{aligned}$$

therefore step a) of Definition 3 preserves (7). Next, let H' be the result of applying normalization step b) to $H \downarrow C$. By a simple induction on the length of $H \downarrow C$, based on Lemma 3.b, it can be shown that $\mathcal{K} \models H \downarrow C \equiv H'$, therefore

$$\mathcal{K} \models C \sqsubseteq H \text{ iff } \mathcal{K} \models C \sqsubseteq H'\,.$$

Similarly, it can be proved that the last step preserves (7) by a simple induction on the length of H' based on Lemma 3.c. The details are left to the reader. □

The next two lemmas need the following definitions:

$$pos(H) = \{C_i \mid +C_i \text{ occurs in } H\}\,,$$
$$neg(H) = \{C_i \mid -C_i \text{ occurs in } H\}\,.$$

Lemma 4. *Let $C \sqsubseteq H$ be any interval safe inclusion where C is a simple $\mathcal{PL}$ concept. If $neg(norm_{\mathcal{K},C}(H)) \neq \emptyset$ then $\mathcal{K} \not\models C \sqsubseteq norm_{\mathcal{K},C}(H)$.*

Proof. Assume that $neg(norm(H)) \neq \emptyset$. Then $norm(H) = H'{-}(C_i \sqcap C)H''$, where $\mathcal{K} \not\models C_i \sqcap C \sqsubseteq \bot$ (otherwise the negative element would have been eliminated in phase b) of the normalization). By Lemma 1, there exists a canonical model $(\mathcal{I}, d)$ of $\mathcal{K}$ and $C_i \sqcap C$. Points a) and b) of Lemma 1 imply that $\mathcal{I} \models \mathcal{K}$ and

$$d \in (C_i \sqcap C)^{\mathcal{I}} \tag{8}$$

therefore $d \notin (H'{-}(C_i \sqcap C))^{\mathcal{I}}$. Moreover, for all elements $+C_j$ in H'', we have

$$\mathcal{K} \not\models C_i \sqcap C \sqsubseteq C_j$$

otherwise $-(C_i \sqcap C)$ would have been eliminated in phase c) of the normalization. Then Lemma 1.c implies that for all $+C_j$ in H'',

$$d \notin C_j^{\mathcal{I}}\,.$$

It follows that $d \notin norm(H)^{\mathcal{I}}$. Moreover, $d \in C^{\mathcal{I}}$, by (8), so $\mathcal{K} \not\models C \sqsubseteq norm(H)$. □

Proposition 2. *For all histories H, if $\mathcal{K} \models C \sqsubseteq H$ then $\mathcal{K} \models C \sqsubseteq \bigsqcup pos(H)$.*

Proof. It can be proved, by a straightforward induction on the length of H, that $\models H \sqsubseteq \bigsqcup pos(H)$. The proposition immediately follows. □

We are finally ready for the main result:

Theorem 2. *Let $\mathcal{K}$ be a $\mathcal{PL}$ knowledge base, C be a simple $\mathcal{PL}$ concept, and H be a history such that $C \sqsubseteq H$ is interval safe. Then $\mathcal{K} \models C \sqsubseteq norm(H)$ iff the following two conditions hold:*

a) $neg(norm(H)) = \emptyset$, and
b) there exists $D \in pos(H) = pos(norm(H))$ such that $\mathcal{K} \models C \sqsubseteq D$.

Proof. ($\Leftarrow$) Suppose that a) and b) hold. By b), $\mathcal{K} \models C \sqsubseteq D$, and clearly $\models D \sqsubseteq \bigsqcup pos(H)$, because D is one of the arguments of the union. Moreover, by a), we have that $H = \bigsqcup pos(H)$. It follows immediately that $\mathcal{K} \models C \sqsubseteq H$.

($\Rightarrow$) We prove the contrapositive. Accordingly, suppose that either a) or b) is false. In the former case, by Lemma 4, $\mathcal{K} \not\models C \sqsubseteq norm(H)$ and the theorem holds. If b) is false, then by convexity $\mathcal{K} \not\models C \sqsubseteq \bigsqcup pos(H)$, and by the contrapositive of Proposition 2, $\mathcal{K} \not\models C \sqsubseteq norm(H)$. □

As a corollary of the above theorem, we obtain the desired tractability result.

Corollary 1. *Let c be an arbitrary but fixed non-negative integer. For all $\mathcal{PL}$ knowledge bases $\mathcal{K}$ and all subsumption queries $q = (C_1 \sqcup \ldots \sqcup C_n \sqsubseteq H)$ such that*

1. *H is a history,*
2. *each C_i is a simple $\mathcal{PL}$ concept (i.e. their union is a full $\mathcal{PL}$ concept),*
3. *for all $i = 1, \ldots, n$, the number of subconcepts of the form $\exists f.[l, u]$ occurring in C_i is bounded by c,*

the entailment $\mathcal{K} \models q$ can be decided in polynomial time.

Proof. Clearly, $\mathcal{K} \models q$ holds iff for all $i = 1, \ldots, n$, $\mathcal{K} \models C_i \sqsubseteq H$ holds. Thus, it suffices to prove that each of these n entailments can be decided in polynomial time. Due to the bound c on the number of intervals, each of these subsumption can be transformed in polynomial time into an equivalent, interval safe inclusion $C_i^* \sqsubseteq H$, as shown in [4, Proposition 5]. By Lemma 2, the resulting subsumption tests can equivalently be replaced by

$$\mathcal{K} \models C_i^* \sqsubseteq \mathit{norm}(H)\,.$$

Now Theorem 2 can be applied to verify the above subsumptions.

The test a) in Theorem 2 takes time $O(|\mathit{norm}(H)|) = O(|H|)$. Test b) requires answering at most $|H|$ $\mathcal{PL}$ queries, each of which takes polynomial time [4, Proposition 6]. So the overall cost of testing the conditions of Theorem 2 is polynomial. We are only left to show that computing $\mathit{norm}(H)$ takes polynomial time. Phase a) takes time $|C| \cdot |H|$. Phase b) requires at most $|H \downarrow C|$ concept consistency tests, each of which takes polynomial time [4, Proposition 8]. Finally, phase c) requires answering at most $|H \downarrow C|$ $\mathcal{PL}$ subsumption queries, each of which takes polynomial time. Thus, the overall computation of $\mathit{norm}(H)$ takes polynomial time. □

Remark 1. Note that the proof of tractability relies only on the following properties of $\mathcal{PL}$:

1. the *convexity* of interval safe $\mathcal{PL}$ (Lemma 1 and Proposition 1), that is needed in Theorem 2;
2. the *tractability of subsumption query answering* in $\mathcal{PL}$ with bounded occurrences of intervals.

Therefore, our results apply to all tasks of the form (7) such that $\mathcal{K}$, C, and the concepts occurring in the history belong to a convex, tractable logic (say, any Horn description logic with a tractable subsumption problem).

In practice, it is not necessary to compute $\mathit{norm}(H)$ explicitly. Consider Algorithm 1, for example. Lines 2–4 check condition b) of Theorem 2. In particular, if the computation reaches line 5, then C_k is the rightmost concept D in H satisfying condition b). The "for" loop at line 5 looks for the elements $-(C_j \sqcap C)$ of $H \downarrow C$ that survive phase b) of the normalization. The loop in lines 7–9 checks whether $-(C_j \sqcap C)$ survives phase c), too. Note that $-(C_j \sqcap C)$ is looked for after $+C_k$, because all the negative elements of $H \downarrow C$ on the left of $+C_k$ are eliminated during normalization phase c), due to $+C_k$ itself. Thus, Algorithm 1 returns *true* iff conditions a) and b) of Theorem 2 hold.

Algorithm 1: $\mathcal{PL}$ Subsumption with histories

```
Input: $\mathcal{K}$, $C$, $H$
  where $C$ is a simple$\mathcal{PL}$ concept, $H = \pm C_1 \dots \pm C_n$, and $C \sqsubseteq H$ is interval safe
Output: true if $\mathcal{K} \models C \sqsubseteq H$, false otherwise
1  if $\mathcal{K} \models C \sqsubseteq \bot$ then return true
   // Condition b)
2  $k := n$
3  while $k > 0$ and $(sign(\pm C_k) \neq '+'$ or $\mathcal{K} \not\models C \sqsubseteq C_k)$ do  $k := k - 1$
4  if $k = 0$ then return false  // else $C_k \in pos(H)$ and $\mathcal{K} \models C \sqsubseteq C_k$
   // Condition a)
5  for $i := k + 1$ to $n$ do
6      if $sign(\pm C_i) = '-'$ and $\mathcal{K} \not\models C_i \sqcap C \sqsubseteq \bot$ then
7          $j := i + 1$
8          while $j \leq n$ and $(sign(\pm C_j) \neq '+'$ or $\mathcal{K} \not\models C_i \sqcap C \sqsubseteq C_j)$ do
9              $j := j + 1$
10         if $j > n$ then return false
11 return true
```

5 Conclusions

Summarizing, $\mathcal{PL}$ queries need to be extended with negation, in order to address the use cases related to automated compliance with the GDPR. Atomic negation raises the complexity of subsumption checking (on which compliance checking is based) at least to the first level of the polynomial hierarchy. This holds even if negation is further restricted to the specific concept expressions that arise in our reference examples. We have tackled this issue by introducing *histories* of positive and negated $\mathcal{PL}$ concepts. Histories represent in a natural way the consent of data subjects and can be processed in polynomial time with Algorithm 1.

The running time of this algorithm grows quadratically with the length of the history, which is likely to be incompatible with the scalability requirements for $\mathcal{PL}$. Thus, in a forthcoming work, we are going to introduce suitable optimizations. Points b) and c) of Lemma 3 already show how to eliminate redundant elements from a history. Further optimizations may partition histories into a set of disjoint, shorter histories. We are also going to explore compact policy representations, where the common parts of history elements are factorized.

References

1. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The DL-Lite family and relations. *J. Artif. Intell. Res.*, 36:1–69, 2009.
2. F. Baader, S. Brandt, and C. Lutz. Pushing the EL envelope. In *IJCAI-05*, pages 364–369. Professional Book Center, 2005.
3. F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications.* Cambridge University Press, 2003.

4. P. A. Bonatti. Fast compliance checking in an OWL2 fragment. In J. Lang, editor, *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden*, pages 1746–1752. ijcai.org, 2018.
5. P. A. Bonatti, L. Ioffredo, I. M. Petrova, L. Sauro, and I. S. R. Siahaan. Real-time reasoning in OWL2 for GDPR compliance. *Artif. Intell.*, 289:103389, 2020.
6. C. Haase and C. Lutz. Complexity of subsumption in the EL family of description logics: Acyclic and cyclic tboxes. In *ECAI 2008 - 18th European Conference on Artificial Intelligence, Proceedings*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 25–29. IOS Press, 2008.
7. R. Rosati. The limits of querying ontologies. In T. Schwentick and D. Suciu, editors, *Database Theory - ICDT 2007, 11th International Conference, Barcelona, Spain, January 10-12, 2007, Proceedings*, volume 4353 of *Lecture Notes in Computer Science*, pages 164–178. Springer, 2007.
8. E. Schlehahn and R. Wenning. Legal requirements for a privacy-enhancing big data V2. SPECIAL deliverable D1.6, April 2018. Available through https://specialprivacy.ercim.eu/.
9. E. Sherkhonov, B. Cuenca Grau, E. Kharlamov, and E. V. Kostylev. Semantic faceted search with aggregation and recursion. In *The Semantic Web - ISWC 2017 - 16th International Semantic Web Conference, Vienna, Austria, October 21-25, 2017, Proceedings, Part I*, pages 594–610, 2017.
10. J. D. Ullman. Information integration using logical views. In F. N. Afrati and P. G. Kolaitis, editors, *Database Theory - ICDT '97, 6th International Conference, Delphi, Greece, January 8-10, 1997, Proceedings*, volume 1186 of *Lecture Notes in Computer Science*, pages 19–40. Springer, 1997.