A richer policy language for GDPR compliance

Piero A. Bonatti[1], Iliana M. Petrova[2], and Luigi Sauro[1]

[1] Università degli studi di Napoli Federico II
[2] CeRICT

Abstract. We illustrate an OWL2 fragment properly tailored for the specification of both companies' data protection policies and data subjects' consent to data processing. Assessing a company's policy for compliance with data subjects' consent is then reduced to a subsumption checking. We provide a complete subsumption algorithm which combines tableaux-based and structural-subsumption techniques and show that, under some conditions, it performs in polynomial time.

1 Introduction

The European General Data Protection Regulation[3] (GDPR) places tight restrictions on the processing of personal data and establishes substantial administrative fines to discourage infringements. Since collecting and processing personal data is a paramount source of innovation and revenue, *data controllers* (i.e. the personal and legal entities that process personal data) are interested in maximizing personal data usage within the limits posed by the GDPR.

The European H2020 project SPECIAL[4] is aimed at supporting controllers with methodological and technological means to comply with the GDPR requirements efficiently and safely. Compliance checking plays a crucial role in this picture: first, the data usage policies of the industrial partners must be compared with a (partial) formalization of the GDPR itself. Second, in the absence of a legitimate legal basis[5], personal data cannot be stored or processed, even temporarily, without the explicit consent of the data subjects. As the number of data subjects (and their policies) can be as large as the number of customers of a major communication service provider, some of the project's use cases consist in checking storage permissions against a stream of incoming data points, at the rate of hundreds of thousands per minute. Thus, a crucial tasks is the development of scalable reasoning procedures for compliance checking.

Here, we focus on $\mathcal{PL}$, that is, the SPECIAL's approach to the representation of data usage activities, consent to data processing, and selected parts of the GDPR. $\mathcal{PL}$ is a DL

[3] http://data.consilium.europa.eu/doc/document/ST-5419-2016-INIT/en/pdf

[4] https://www.specialprivacy.eu/

[5] Such legal bases include public interest, the vital interests of the data subject, signed contracts, and the legitimate interests of the data controller, just to name a few (cf. GDPR Art. 6). However, the regulation constrains the applicability of these legal bases with a number of provisos and caveats as the data minimization principle introduced in Art. 5, and the limitations to the legitimate interests of the controller rooted in Art. 6.1(f).

fragment specifically designed to balance expressiveness requirements to encode data usage policies and scalability of reasoning. A first formalization has been presented in [5]. In this paper, we describe an extention, $\mathcal{PL}^+$, that overcomes some limitations of expressiveness that arose from a further in-depth analysis of the GDPR on one side, and the policy specifications of the industrial partners involved in the project on the other.

More precisely, $\mathcal{PL}^+$ enhances $\mathcal{PL}$ in the following aspects:

- it extends $\mathcal{PL}$ by including role domains in the ontology language and role chain agreements ($R_1 \circ \cdots \circ R_n = S_1 \circ \cdots \circ S_m$) in the query language;
- it relies on a new subsumption checking algorithm that combines tableaux-based and structural subsumption reasoning techniques. This algorithm is notably simpler than the one presented in [5] and hence we expect it to perform faster.

Thereafter, we focus on the technical aspects of $\mathcal{PL}^+$ referring to [5] for further motivations and examples.

2 Preliminaries

$\mathcal{PL}^+$ is tailored to formalize policies regulating the usage of personal data. It can express (i) the policies of *data controllers* (i.e. the organizations that collect and process personal data), (ii) the selective consent to data usage issued by data subjects, and (iii) parts of the GDPR related to consent management, user rights, and data transfer.

The aspects of data usage that have legal relevance are clearly indicated in several articles of the GDPR. In particular, the main properties of a *usage policy* that need to be encoded and archived include:

- reasons for data processing (purpose);
- which data categories are involved;
- what kind of processing is applied to the data;
- which third parties data are distributed to (recipients);
- countries in which the data will be stored (location);
- time limits for data erasure (duration).

Here we focus only on the technical aspects needed in this work and refer the reader to [3] for further details. The $\mathcal{PL}^+$ language is built from countably infinite and mutually disjoint sets of concept names ($\mathsf{N_C}$), role names ($\mathsf{N_R}$) and concrete feature 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* $\cdot^{\mathcal{I}}$, defined over the signature $\Sigma = \mathsf{N_C} \cup \mathsf{N_R} \cup \mathsf{N_F}$, is s.t. (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 $\mathbb{N}$ is the set of natural numbers. A pointed interpretation is a pair $\mathcal{I}, d$ where $\mathcal{I}$ is an interpretation and d is an element of the domain $\Delta^{\mathcal{I}}$.

A $\mathcal{PL}^+$ knowledge base KB is a finite set of axioms of the form $\mathsf{func}(f)$, $\mathsf{func}(R)$, $\mathsf{dom}(R, A)$, $\mathsf{ran}(R, A)$, $\mathsf{disj}(A, B)$, and $A \sqsubseteq B$, where A and B are concept names, R is a role name, and f denotes a concrete feature. An interpretation $\mathcal{I}$ *satisfies* an axiom α if $\mathcal{I}$ satisfies the corresponding semantic condition in Figure 1.

Figure 2 shows the syntax and semantics of $\mathcal{PL}^+$ operators which allow to define query concepts. We say that a concept is *simple* if no disjunction occurs in it. By applying the logical equivalences $C_1 \sqcap (C_2 \sqcup C_3) \equiv (C_1 \sqcap C_2) \sqcup (C_1 \sqcap C_3)$ and

Name	Syntax	Semantics
GCI	$A \sqsubseteq B$	$A^{\mathcal{I}} \subseteq B^{\mathcal{I}}$
disj	$\mathsf{disj}(A, B)$	$A^{\mathcal{I}} \cap B^{\mathcal{I}} = \emptyset$
role functionality	$\mathsf{func}(R)$	$R^{\mathcal{I}}$ is a partial function
concr. feature functionality	$\mathsf{func}(f)$	$f^{\mathcal{I}}$ is a partial function
domain	$\mathsf{dom}(R, A)$	$R^{\mathcal{I}} \subseteq A^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
range	$\mathsf{ran}(R, A)$	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times A^{\mathcal{I}}$

Fig. 1. Syntax and semantics of $\mathcal{PL}^+$ axioms.

$\exists R.(C_1 \sqcup C_2) \equiv \exists R.C_1 \sqcup \exists R.C_2$ any concept C can be reduced in a normal form $C_1 \sqcup \cdots \sqcup C_n$ where each C_i is simple (clearly, the normal form can be exponentially longer than C). A pointed interpretation satisfies a concept C, formally $\mathcal{I}, d \models C$, iff $d \in C^{\mathcal{I}}$. Then, $\mathcal{I}$ satisfies C in case $\mathcal{I}, d \models C$ for some $d \in \Delta^{\mathcal{I}}$. As usual, $KB \models C \sqsubseteq D$ means that $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, for all models $\mathcal{I}$ of KB.

Example 1. A company – call it BeFit – sells a wearable fitness appliance and wants to process biometric data (stored in the EU) for sending health-related advice to a customer. Furthermore, BeFit wants to share the customer's location data with a group of enterprises (located in the USA) engaged in a joint economic activity. According to Art. 47 of the GDPR, the transfer of personal data outside of EU is allowed towards recipients that enforce, so-called, binding corporate rules. 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 do all this legally, BeFit needs consent from its customers. The consent BeFit acquires from a customer may look as follows:

$$
\begin{aligned}
&(\exists purpose.FitnessRecommendation \sqcap \exists data.BiometricData \sqcap \\
&\ \exists processing.Analytics \sqcap \exists recipient.BeFit \sqcap \exists store.\exists location.EU) \\
&\sqcup \\
&(\exists purpose.Monetize \sqcap \exists data.LocationData \sqcap \exists processing.Transfer \sqcap \\
&\ \exists recipient.USEnterpriseGroup \sqcap \exists store.(\exists location.USA \sqcap duration : [1,5]) \sqcap \\
&\ (controller \circ\ applyBindingCorporateRulesTo = recipient) \sqcap \\
&\ \exists controller.BeFit)\,,
\end{aligned}
$$

where integers represent years. Moreover, KB can include the following axioms:

$$HeartRate \sqsubseteq BiometricData \qquad ComputeAvg \sqsubseteq Analytics$$

$$func(recipient)$$

Given the knowledge base, the first disjunct of the above consent allows BeFit to compute the average heart rate of the data subject in order to send her fitness recommendations. The purpose could be further detailed by saying how the data subject is contacted, e.g. "recommendation via SMS". Then, in the customer's consent

Name	Syntax	Semantics
bottom	$\bot$	$\bot^{\mathcal{I}} = \emptyset$
conjunction	$C \sqcap D$	$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$
disjunction	$C \sqcup D$	$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
existential restriction	$\exists R.C$	$\{d \in \Delta^{\mathcal{I}} \mid \exists (d,e) \in R^{\mathcal{I}} : e \in C^{\mathcal{I}}\}$
concrete constraint	$f : [l,u]$	$\{d \in \Delta^{\mathcal{I}} \mid \exists w \in \mathbb{N}\, s.t.\, (d,w) \in f^{\mathcal{I}}$ and $l \leq w \leq u)$
existential chain restriction	$(R_1 \circ \cdots \circ R_n = S_1 \circ \cdots \circ S_m)$	$\{d \in \Delta^{\mathcal{I}} \mid \exists \bar{d}$ s.t. $(d,\bar{d}) \in (R_1 \circ \cdots \circ R_n)^{\mathcal{I}} \cap (S_1 \circ \cdots \circ S_m)^{\mathcal{I}}\}$

Fig. 2. Syntax and semantics of $\mathcal{PL}^+$ (query) concepts.

$\exists purpose.FitnessRecommendation$ should be replaced with something like

$$\exists purpose.(FitnessRecommendation \sqcap \exists contact.SMS)\,.$$

Note also that, since $recipient$ is a functional role, the chain agreement in the second disjunct imposes to the recipient the application of the controller's binding corporate rules. □

An Abox $\mathcal{A}$ is a finite set of assertions of the form $C(x)$ and $R(x,y)$, where R is a role name, C is a simple query concept, and x and y belong to a countably infinite set of variable names VR. In particular, we assume that VR contains a distinguished variable x_0. An assignment $\pi : \mathsf{VR} \to \Delta^{\mathcal{I}}$ interprets all variables in VR as elements of an interpretation $\mathcal{I}$. Let $\mathsf{VR}_{\mathcal{A}} \subseteq \mathsf{VR}$ be the set of variable names occurring in $\mathcal{A}$, we say that π and π' agree in $\mathcal{A}$ if $\pi(x) = \pi'(x)$, for all $x \in \mathsf{VR}_{\mathcal{A}}$. Then, a pair $\mathcal{I}, \pi$ satisfies an Abox $\mathcal{A}$ (formally, $\mathcal{I}, \pi \models \mathcal{A}$) iff (i) $\pi(x) \in C^{\mathcal{I}}$, for all $C(x) \in \mathcal{A}$, and (ii) $(\pi(x), \pi(y)) \in R^{\mathcal{I}}$, for all $R(x,y) \in \mathcal{A}$.

A pointed interpretation $\mathcal{I}, d$ satisfies $\mathcal{A}$ (formally, $\mathcal{I}, d \models \mathcal{A}$) if $\mathcal{I}, \pi \models \mathcal{A}$ for some assignment π with $\pi(x_0) = d$. We write $KB, \mathcal{A} \models C(x)$ meaning that $\mathcal{I}, \pi \models \mathcal{A}$ implies $\mathcal{I}, \pi \models C(x)$, for all models $\mathcal{I} \models KB$ and assignments π.

Finally, we say that C (resp. $\mathcal{A}$) and D are *interval safe* iff for all constraints $f : [l,u]$ occurring in C (resp. for all $f : [l,u]$ s.t. $f : [l,u](x) \in \mathcal{A}$, for some variable x) and $f : [l',u']$ occurring in D, either $[l,u] \subseteq [l',u']$, or $[l,u] \cap [l',u'] = \emptyset$.

3 The subsumption algorithm

Checking whether $KB \models C \sqsubseteq D$ is performed in three phases. The first phase pre-processes the concept C. First of all, for each $f : [l,u]$ in C, let $x_1 < x_2 < \cdots < x_r$ be the integers that occur as interval endpoints in sub-concept $f : [l',u']$ of D and belong to $[l,u]$. Let $x_0 = \ell$ and $x_{r+1} = u$ and replace $f[l,u]$ with the equivalent concept

$$\bigsqcup_{i=0}^{r} \big(f : [x_i, x_i] \sqcup f : [x_i + 1, x_{i+1} - 1]\big) \sqcup f : [x_{r+1}, x_{r+1}]. \tag{1}$$

Doing so, we ensure that C and D are interval safe. Then, C is translated in the corresponding normal form $C_1 \sqcup \cdots \sqcup C_n$. As a consequence, a knowledge base KB entails

Rule $\rightarrow_{\sqcap}$	if 1. $C_1 \sqcap C_2(x) \in \mathcal{A}$ and 2. $\{C_1(x), C_2(x)\} \not\subseteq \mathcal{A}$, then set $\mathcal{A}' \leftarrow \mathcal{A} \cup \{C_1(x), C_2(x)\}$
Rule $\rightarrow_{\sqsubseteq}$	if 1. $A(x) \in \mathcal{A}$ and 2. $A \sqsubseteq B \in KB$, and 3. $B(x) \notin \mathcal{A}$ then set $\mathcal{A}' \leftarrow \mathcal{A} \cup \{B(x)\}$
Rule $\rightarrow_{\exists}$	if 1. $\exists R.C(x) \in \mathcal{A}$ and 2. for no y $\{R(x,y), C(y)\} \subseteq \mathcal{A}$ then create a new variable y_{new} and set $\mathcal{A}' \leftarrow \mathcal{A} \cup \{R(x, y_{new}), C(y_{new})\}$
Rule $\rightarrow_{\mathsf{dom}}$	if 1. $R(x,y) \in \mathcal{A}$ and 2. $\mathsf{dom}(R, A) \in KB$, and 3. $A(x) \notin \mathcal{A}$, then set $\mathcal{A}' \leftarrow \mathcal{A} \cup \{A(x)\}$
Rule $\rightarrow_{\mathsf{ran}}$	if 1. $R(x,y) \in \mathcal{A}$ and 2. $\mathsf{ran}(R, A) \in KB$, and 3. $A(y) \notin \mathcal{A}$ then set $\mathcal{A}' \leftarrow \mathcal{A} \cup \{A(y)\}$
Rule $\rightarrow_{\mathsf{func}(R)}$	if 1. $\{R(x,y), R(x,z)\} \subseteq \mathcal{A}$ and 2. $\mathsf{func}(R) \in KB$ then set $\mathcal{A}' \leftarrow \mathcal{A}[y/z]$
Rule $\rightarrow_{\mathsf{func}(f)}$	if 1. $\{f : [l,u](x), f : [v,w](x)\} \subseteq \mathcal{A}$ and 2. $\mathsf{func}(f) \in KB$ then set $\mathcal{A}' \leftarrow \mathcal{A} \cup \{f : [\max(l,v), \min(u,w)]\}$
Rule $\rightarrow_{=}$	if 1. $(R_1 \circ \cdots \circ R_n = S_1 \circ \cdots \circ S_m)(x) \in \mathcal{A}$, and 2. there does not exist $y_1, \ldots, y_{n-1}, z_1, \ldots, z_{m-1}, \bar{y}$ s.t. $\{R_1(x, y_1), S_1(x, z_1)\} \subseteq \mathcal{A}$, $R_i(y_{i-1}, y_i) \in \mathcal{A}$, for $i = 1, \ldots, n-1$, $S_j(z_{j-1}, z_j) \in \mathcal{A}$, for $j = 1, \ldots, m-1$, $\{R_n(y_{n-1}, \bar{y}), S_m(z_{m-1}, \bar{y}\} \subseteq \mathcal{A}$ then create new variables $y_1^{new}, \ldots, y_{n-1}^{new}, z_1^{new}, \ldots, z_{m-1}^{new}, \bar{y}^{new}$ and set $\mathcal{A}' \leftarrow \mathcal{A} \cup \{R_1(x, y_1^{new}), \ldots, R_n(y_{n-1}^{new}, \bar{y}^{new})\}$ $\cup \{S_1(x, z_1^{new}), \ldots, S_m(z_{m-1}^{new}, \bar{y}^{new})\}$
Rule $\rightarrow_{\mathsf{disj}}$	if 1. $\{A(x), B(x)\} \subseteq \mathcal{A}$ and 2. $\mathsf{disj}(A, B) \in KB$ then set $\mathcal{A}' \leftarrow \mathcal{A} \cup \{\bot(x)\}$
Rule $\rightarrow_{f_\bot}$	if 1. $f : [l,u](x) \in \mathcal{A}$ and 2. $u < l$ then set $\mathcal{A}' \leftarrow \mathcal{A} \cup \{\bot(x)\}$

Fig. 3. $\mathcal{PL}^+$ tableaux rules

$C \sqsubseteq D$ iff $KB \models C_i \sqsubseteq D$, for all $i = 1, \ldots, n$[6].

After the preprocessing phase, the subsumption problem is reduced to checking whether each simple concept C_i is subsumed by D. For a simple concept C_i, during the second phase, an Abox $\mathcal{A}_0$ is first initialised with a $C_i(x_0)$ and then expanded by exhaustively applying the tableaux rules in Figure 3. The subsumption vacuously holds if the resulting Abox $\mathcal{A}$ contains assertions of the form $\bot(x)$ for some x. In contrary case, the third phase checks whether $x_0 \in \langle\langle D \rangle\rangle_{\mathcal{A}}$, where $\langle\langle D \rangle\rangle_{\mathcal{A}}$ is the set of variable names contained in $\mathsf{VR}_{\mathcal{A}}$ inductively defined as follows:

[6] Note that, differently from the subsumption algorithm presented in [5], D is not normalized. This may significantly impact on performance since the normalization process might exponentially inflate D.

$$
\begin{aligned}
\langle\!\langle A \rangle\!\rangle_{\mathcal{A}} &= \{x \in \mathsf{VR}_{\mathcal{A}} \mid A(x) \in \mathcal{A}\} \\
\langle\!\langle D_1 \sqcap D_2 \rangle\!\rangle_{\mathcal{A}} &= \{x \in \mathsf{VR}_{\mathcal{A}} \mid x \in \langle\!\langle D_1 \rangle\!\rangle_{\mathcal{A}} \cap \langle\!\langle D_2 \rangle\!\rangle_{\mathcal{A}}\} \\
\langle\!\langle D_1 \sqcup D_2 \rangle\!\rangle_{\mathcal{A}} &= \{x \in \mathsf{VR}_{\mathcal{A}} \mid x \in \langle\!\langle D_1 \rangle\!\rangle_{\mathcal{A}} \cup \langle\!\langle D_2 \rangle\!\rangle_{\mathcal{A}}\} \\
\langle\!\langle \exists R.D \rangle\!\rangle_{\mathcal{A}} &= \{x \in \mathsf{VR}_{\mathcal{A}} \mid \exists y \in \mathsf{VR}_{\mathcal{A}} \text{ s.t. } R(x,y) \in \mathcal{A} \\
&\qquad \text{and } y \in \langle\!\langle D \rangle\!\rangle_{\mathcal{A}}\} \\
\langle\!\langle R_1 \circ \cdots \circ R_n = S_1 \circ \cdots \circ S_m \rangle\!\rangle_{\mathcal{A}} &= \{x \in \mathsf{VR}_{\mathcal{A}} \mid \exists (y,z) \in \mathsf{VR}^2_{\mathcal{A}} \text{ s.t. } (y,z) \in \langle\!\langle R_n, S_m \rangle\!\rangle_{\mathcal{A}} \\
&\qquad \text{and } x \in \langle\!\langle R_1 \circ \cdots \circ R_{n-1}, y \rangle\!\rangle_{\mathcal{A}} \\
&\qquad \text{and } x \in \langle\!\langle S_1 \circ \cdots \circ S_{m-1}, z \rangle\!\rangle_{\mathcal{A}}\} \\
\langle\!\langle R, S \rangle\!\rangle_{\mathcal{A}} &= \{(y,z) \in \mathsf{VR}^2_{\mathcal{A}} \mid \exists w \in \mathsf{VR}_{\mathcal{A}} \text{ s.t. } R(y,w) \in \mathcal{A} \\
&\qquad \text{and } S(z,w) \in \mathcal{A}\} \\
\langle\!\langle \cdot, y \rangle\!\rangle_{\mathcal{A}} &= \{y\} \\
\langle\!\langle \hat{R}_1 \circ \cdots \circ \hat{R}_k, y \rangle\!\rangle_{\mathcal{A}} &= \{x \in \mathsf{VR}_{\mathcal{A}} \mid \exists w \in \mathsf{VR}_{\mathcal{A}} \text{ s.t. } \hat{R}_1(x,w) \in \mathcal{A} \\
&\qquad \text{and } w \in \langle\!\langle \hat{R}_2 \circ \cdots \circ \hat{R}_k, y \rangle\!\rangle_{\mathcal{A}}\} \\
\langle\!\langle f : [a,b] \rangle\!\rangle_{\mathcal{A}} &= \{x \in \mathsf{VR}_{\mathcal{A}} \mid \exists a', b' \in \mathbb{N} \text{ s.t. } f : [a',b'](x) \in \mathcal{A} \\
&\qquad \text{and } a \leq a' \leq b' \leq b\}
\end{aligned}
$$

Note that $\langle\!\langle R, S \rangle\!\rangle_{\mathcal{A}}$, $\langle\!\langle \hat{R}_1 \circ \cdots \circ \hat{R}_k, y \rangle\!\rangle_{\mathcal{A}}$, and $\langle\!\langle \cdot, y \rangle\!\rangle_{\mathcal{A}}$ are only used to support the computation of the set $\langle\!\langle R_1 \circ \cdots \circ R_n = S_1 \circ \cdots \circ S_m \rangle\!\rangle_{\mathcal{A}}$ of variable names in $\mathsf{VR}_{\mathcal{A}}$ that satisfies a role chain agreement. Intuitively, $\langle\!\langle R_n, S_m \rangle\!\rangle_{\mathcal{A}}$ computes the set of pairs (y,z) that are connected to a single variable through the role names R_n and S_m. Then, $\langle\!\langle R_1 \circ \cdots \circ R_{n-1}, y \rangle\!\rangle_{\mathcal{A}}$ (resp. $\langle\!\langle S_1 \circ \cdots \circ S_{m-1}, z \rangle\!\rangle_{\mathcal{A}}$) recursively computes the set of variables connected to y (resp. to z) through a role chain $R_1 \circ \cdots \circ R_{n-1}$ (resp. $S_1 \circ \cdots \circ S_{m-1}$).

Thereafter, $\mathcal{A} \to \mathcal{A}'$ means that $\mathcal{A}'$ results from the application of a rule in Figure 3 to $\mathcal{A}$. As usual $\stackrel{*}{\to}$ denotes the reflexive and transitive closure of the $\to$ relation. $\mathcal{A}$ is complete (w.r.t. KB) if no rule is applicable and is clash-free if it does not contain assertions of the form $\bot(x)$. Then, let C be a simple concept and KB a $\mathcal{PL}^+$ knowledge base, we say that $\mathcal{A}$ is the tableaux of C w.r.t. KB if $\mathcal{A}$ is complete and $\{C(x_0)\} \stackrel{*}{\to} \mathcal{A}$.

Lemma 1. *Let $\mathcal{A}$ and $\mathcal{A}'$ be two Aboxes over VR s.t. $\mathcal{A} \to \mathcal{A}'$. For all models $\mathcal{I}$ of KB and $d \in \Delta^{\mathcal{I}}$, we have that $\mathcal{I}, d \models \mathcal{A}$ iff $\mathcal{I}, d \models \mathcal{A}'$.*

Proof. First, consider $\mathcal{A} \to_{\mathsf{func}(R)} \mathcal{A}'$ and assume that $\mathcal{I}, d \models \mathcal{A}$. This means that for some assignment π, with $\pi(x_0) = d$, $\mathcal{I}, \pi \models \mathcal{A}$. Since the rule $\to_{\mathsf{func}(R)}$ is applicable to $\mathcal{A}$, it must be the case that $\mathcal{I}, \pi \models R(x, y_1)$ and $\mathcal{I}, \pi \models R(x, y_2)$, for some $x, y_1, y_2 \in \mathsf{VR}_{\mathcal{A}}$ and a role R s.t. $\mathsf{func}(R) \in K$. Futhermore, $\mathcal{I} \models KB$ by hypothesis, so $\pi(y_1) = \pi(y_2)$ and hence $\mathcal{I}, \pi \models \mathcal{A}'$. Vice versa, assume that $\mathcal{I}, d \models \mathcal{A}'$ and let π be the corresponding assignment. Since y_2 does not occur in $\mathcal{A}'$, π is free to map y_2 arbitrarily. Then, in order to satisfy $\mathcal{A}$, it is sufficient to assign $\pi(y_1)$ to y_2. Since in both directions the assignment of x_0 has not changed, we have that $\mathcal{I}, d \models \mathcal{A}$ iff $\mathcal{I}, d \models \mathcal{A}'$.

Regarding the other rules, note that $\mathcal{A} \subseteq \mathcal{A}'$ and hence $\mathcal{I}, d \models \mathcal{A}'$ implies $\mathcal{I}, d \models \mathcal{A}$ by monotonicity. For the opposite direction, assume that $\mathcal{I}, \pi \models \mathcal{A}$, where $\pi(x_0) = d$.

Case $\mathcal{A} \to_{\sqcap} \mathcal{A}'$. Since $\to_{\sqcap}$ is applicable to $\mathcal{A}$, $(C_1 \sqcap C_2)(x)$ must occur in $\mathcal{A}$, for some simple concepts C_1, C_2 and a variable $x \in \mathsf{VR}_{\mathcal{A}}$. $\mathcal{I}, \pi \models C_1(x)$ and $\mathcal{I}, \pi \models C_2(x)$

directly follow from the assumption $\mathcal{I}, \pi \models \mathcal{A}$. Consequently, $\mathcal{I}, \pi \models \mathcal{A}'$.

Case $\mathcal{A} \rightarrow_{\sqsubseteq} \mathcal{A}'$. Since $\rightarrow_{\sqsubseteq}$ is applicable to $\mathcal{A}$, for some atomic concepts A, B and a variable $x \in \mathsf{VR}_{\mathcal{A}}$, $A(x) \in \mathcal{A}$ and $A \sqsubseteq B \in KB$. Then, from $\mathcal{I}, \pi \models \mathcal{A}$ and $\mathcal{I} \models KB$, it follows that $\mathcal{I}, \pi \models B(x)$ and hence $\mathcal{I}, \pi \models \mathcal{A}'$.

Case $\mathcal{A} \rightarrow_{\exists} \mathcal{A}'$. From the precondition of the rule $\rightarrow_{\exists}$, we have that $\exists R.C(x)$ occurs in $\mathcal{A}$. Moreover, from the assumption $\mathcal{I}, d \models \mathcal{A}$, it holds $d_x \in (\exists R.C)^{\mathcal{I}}$, with $\pi(x) = d_x$, which means that there exists an individual d' s.t. $d' \in C^{\mathcal{I}}$ and $(d_x, d') \in R^{\mathcal{I}}$. Since y^{new} does not occur in $\mathcal{A}$, we can assign $\pi(y^{new}) = d'$ and obtain $\mathcal{I}, \pi \models \mathcal{A}'$.

Case $\mathcal{A} \rightarrow_{\mathsf{dom}} \mathcal{A}'$. From the precondition of $\rightarrow_{\mathsf{dom}}$, $R(x, y) \in \mathcal{A}$ and $\mathsf{dom}(R, A) \in KB$. So, any $\mathcal{I}, \pi$ satisfying $\mathcal{A}$ and KB must also satisfy $A(x)$ and hence $\mathcal{A}'$.

Case $\mathcal{A} \rightarrow_{\mathsf{ran}} \mathcal{A}'$. Similar to the previous case.

Case $\mathcal{A} \rightarrow_{\mathsf{func}(f)} \mathcal{A}'$. From the precondition of the $\rightarrow_{\mathsf{func}(f)}$ rule, we have that $\mathcal{I}, \pi$ satisfy $f : [l, u](x)$ and $f : [v, w](x)$ where, according to KB, f is a function from $\Delta^{\mathcal{I}}$ to $\mathbb{N}$. Then, given $d_x = \pi(x)$, both $l \leq f^{\mathcal{I}}(d_x) \leq u$ and $v \leq f^{\mathcal{I}}(d_x) \leq w$ hold which clearly means that $\max(l, v) \leq f^{\mathcal{I}}(d_x) \leq \min(u, w)$. Consequently, $\mathcal{I}, \pi \models \mathcal{A}'$.

Case $\mathcal{A} \rightarrow_{=} \mathcal{A}'$. By assumption $\mathcal{I}, \pi \models \mathcal{A}$ and $(R_1 \circ \cdots \circ R_n = S_1 \circ \cdots \circ S_m)(x) \in \mathcal{A}$. Let d_x be the individual associated to the variable x by π. There exist two paths $d_x, d'_1, \ldots, d'_{n-1}, \bar{d}$ and $d_x, d''_1, \ldots, d''_{m-1}, \bar{d}$ s.t. (i) $(d_x, d'_1) \in R_1^{\mathcal{I}}$ and $(d_x, d''_1) \in S_1^{\mathcal{I}}$; (ii) $(d'_{i-1}, d'_i) \in R_i^{\mathcal{I}}$, with $i = 2, \ldots, n-1$; (iii) $(d''_{j-1}, d''_j) \in S_j^{\mathcal{I}}$, with $j = 2, \ldots, m-1$; (iv) $(d'_{n-1}, \bar{d}) \in R_n^{\mathcal{I}}$ and $(d''_{m-1}, \bar{d}) \in S_m^{\mathcal{I}}$. Since the variables $y_1^{new}, \ldots, y_{n-1}^{new}, z_1^{new}, \ldots, z_{m-1}^{new}, \bar{y}^{new}$ are fresh, we can extend π by mapping them to $d'_1, \ldots, d'_{n-1}, d''_1, \ldots, d''_{m-1}, \bar{d}$, respectively. $\mathcal{I}, \pi \models \mathcal{A}'$ follows by construction.

Case $\mathcal{A} \rightarrow_{\mathsf{disj}} \mathcal{A}'$. According to the precondition of $\rightarrow_{\mathsf{disj}}$, $\{A(x), B(x)\} \subseteq \mathcal{A}$ and $\mathsf{disj}(A, B) \in KB$. Then, this case is vacuously satisfied since there does not exist a model $\mathcal{I}$ of KB and an assignment π that satisfy $\mathcal{A}$.

Case $\mathcal{A} \rightarrow_{f_\perp} \mathcal{A}'$.Vacuously satisfied, otherwise we have $l \leq f^{\mathcal{I}}(d) \leq u$ and $u < l$.

Finally, note that in all of the previous cases π has been extended by mapping only fresh variables. The assignment of x_0 therefore never changes, hence $\mathcal{I}, d \models \mathcal{A}'$. □

Let $\mathcal{A}$ be a complete and clash-free Abox, a canonical model of $\mathcal{A}$ is an interpretation $\mathcal{J}_{\mathcal{A}} = (\Delta^{\mathcal{J}_{\mathcal{A}}}, \cdot^{\mathcal{J}_{\mathcal{A}}})$ satisfying the following conditions:

1. $\Delta^{\mathcal{J}_{\mathcal{A}}} = \mathsf{VR}_{\mathcal{A}}$;
2. $A^{\mathcal{J}_{\mathcal{A}}} = \{x \in \mathsf{VR}_{\mathcal{A}} \mid A(x) \in \mathcal{A}\}$;
3. $R^{\mathcal{J}_{\mathcal{A}}} = \{(x, y) \in \mathsf{VR}_{\mathcal{A}} \times \mathsf{VR}_{\mathcal{A}} \mid R(x, y) \in \mathcal{A}\}$;
4. if f is non-functional (i.e. $\mathsf{func}(f) \notin KB$), then
   (a) $f^{\mathcal{J}_{\mathcal{A}}} \subseteq \{(x, h) \in \mathsf{VR}_{\mathcal{A}} \times \mathbb{N} \mid \exists l, u \in \mathbb{N}$ s.t. $f : [l, u](x) \in \mathcal{A}$ and $l \leq h \leq u\}$;
   (b) for each $f : [l, u](x) \in \mathcal{A}$, there exists at least one $l \leq h \leq u$ s.t. $(x, h) \in f^{\mathcal{J}_{\mathcal{A}}}$;
5. if f is functional, then $(x, h) \in f^{\mathcal{J}_{\mathcal{A}}}$ iff
   (a) $h = \max\{l \mid f : [l, u](x) \in \mathcal{A}\}$ or $h = \min\{u \mid f : [l, u](x) \in \mathcal{A}\}$;
   (b) for each $h' \neq h$, $(x, h') \notin f^{\mathcal{J}_{\mathcal{A}}}$.

Note that all canonical models share the same domain as well as the same interpretation of atomic concepts and roles. Concrete features involve multiple canonical models, in particular condition 4 requires a canonical model to satisfy all assertions $f : [l, u](x)$ for a not functional f occurring in $\mathcal{A}$. Condition 5 associates to a variable x the maximum lower (resp. minimum upper) bound of the intervals relative to x and a functional f.

Lemmas 2 and 3 show that a canonical model $\mathcal{J}_\mathcal{A}$ of $\mathcal{A}$ satisfies both $\mathcal{A}$ and KB.

Lemma 2. *Let $\mathcal{A}$ be a complete and clash-free Abox, $\mathcal{J}_\mathcal{A}$ a canonical model of $\mathcal{A}$ and π the identity assignment s.t. $\pi(x) = x$, for all x occurring in $\mathcal{A}$. Then, $\mathcal{J}_\mathcal{A}, \pi \models \mathcal{A}$.*

Proof. Clearly, $\mathcal{J}_\mathcal{A}$ and π satisfy all assertions $A(x)$ and $R(x, y)$ occurring in $\mathcal{A}$ by construction. Moreover, given $f : [l, u](x) \in \mathcal{A}$ corresponding to non-functional feature, by construction there exists a value $l \leq h \leq u$ s.t. $(x, h) \in f^\mathcal{J}_\mathcal{A}$ (see condition 4). As a consequence, also $f : [l, u](x) \in \mathcal{A}$ is satisfied.

Let f be a functional concrete feature and assume by contradiction that $\mathcal{J}_\mathcal{A}, \pi \not\models f : [l, u](x)$, with $f : [l, u](x) \in \mathcal{A}$. By construction, $f^\mathcal{J}_\mathcal{A}$ associates to x a single value h which is either $\max\{l \mid f : [l, u](x) \in \mathcal{A}\}$ or $\min\{u \mid f : [l, u](x) \in \mathcal{A}\}$. Consider the first case and let $f : [\hat{l}, \hat{u}](x)$ be an assertion in $\mathcal{A}$ s.t. $\hat{l} = \max\{l \mid f : [l, u](x) \in \mathcal{A}\}$. From $\mathcal{J}_\mathcal{A} \not\models f : [l, u](x)$ and the fact that $l \leq \hat{l}$ it follows that $\hat{l} > u$. Moreover, since $\mathcal{A}$ is complete, by applying $\rightarrow_{\mathsf{func}(f)}$ to $f : [l, u](x)$ and $f : [\hat{l}, \hat{u}](x)$ we obtain that $f : [\max(l, \hat{l}), \min(u, \hat{u})](x)$ is also in $\mathcal{A}$. Then, we have that $\max(l, \hat{l}) = \hat{l}$ and $\min(u, \hat{u}) \leq u < \hat{l}$. Consequently, by application of rule $\rightarrow_{f_\bot}$ we would have that $\bot(x)$ occurs in $\mathcal{A}$ against the hypothesis that $\mathcal{A}$ is clash-free. Symmetrically, the assumption that $f^\mathcal{J}_\mathcal{A}$ associates to x the minimum upper bound once again results in $\bot(x) \in \mathcal{A}$. □

Lemma 3. *Let $\mathcal{A}$ be clash-free and complete w.r.t. KB, then a canonical model $\mathcal{J}_\mathcal{A}$ of $\mathcal{A}$ is also a model of KB.*

Proof. The proof is by reduction ad absurdum. Assume $\mathcal{J}_\mathcal{A}$ does not satisfy $A \sqsubseteq B \in KB$. Then by construction, for some variable $x \in \mathsf{VR}_\mathcal{A}$, $A(x) \in \mathcal{A}$ and $B(x) \notin \mathcal{A}$. However, in this case $\rightarrow_\sqsubseteq$ would be applicable to $\mathcal{A}$. Analogously, the assumptions that $\mathcal{J}_\mathcal{A}$ does not satisfy $\mathsf{dom}(R) \sqsubseteq A \in KB$, $\mathsf{ran}(R) \sqsubseteq A \in KB$ and $\mathsf{func}(R) \in KB$ all lead to a contradiction of the hypothesis that $\mathcal{A}$ is complete. Finally, for each f s.t. $\mathsf{func}(f) \in KB$, $f^\mathcal{J}_\mathcal{A}$ is functional by construction. □

The following lemmas prove that, under the interval safety assumption, the semantic consequences of a knowledge base and a complete Abox can be computed by $\langle\!\langle \cdot \rangle\!\rangle_\mathcal{A}$.

Lemma 4. *Let $\mathcal{A}$ be clash-free and complete w.r.t. KB and D a concept s.t. D and $\mathcal{A}$ are interval safe. Then, $KB, \mathcal{A} \models D(x)$ iff $\mathcal{J}_\mathcal{A}, x \models D$, for all canonical models $\mathcal{J}_\mathcal{A}$.*

Proof. For the the left-to-right direction, we already know that each canonical model $\mathcal{J}_\mathcal{A}$ satisfies KB and $\mathcal{J}_\mathcal{A}, \pi \models \mathcal{A}$, where π is the identity assignment. Then, $KB, \mathcal{A} \models D(x)$ in particular implies $\mathcal{J}_\mathcal{A}, \pi \models D(x)$ and, since $\pi(x) = x$, we have that $\mathcal{J}_\mathcal{A}, x \models D$. The right-to-left direction can be proved by induction on the structure of D.

Case $D = A$. By definition, $\mathcal{J}_\mathcal{A}, x \models A$ iff $x \in A^{\mathcal{J}_\mathcal{A}}$ iff $A(x) \in \mathcal{A}$. Moreover, given a model $\mathcal{I}$ of KB and an assignment π, the facts that $\mathcal{I}, \pi \models \mathcal{A}$ and $A(x) \in \mathcal{A}$ imply by definition that $\mathcal{I}, \pi \models A(x)$. Hence, $KB, \mathcal{A} \models A(x)$.

Case $D = f : [l, u]$, where f is not functional. We show that if $\mathcal{J}_\mathcal{A}, x \models f : [l, u]$, for all canonical models $\mathcal{J}_\mathcal{A}$, then $\mathcal{A}$ contains an assertion $f : [l', u'](x)$ with $l' \geq l$ and $u' \leq u$. This clearly ensures that $KB, \mathcal{A} \models f : [l, u](x)$. Let $f : [l_1, u_1](x), \ldots, f : [l_n, u_n](x)$ be the assertions in $\mathcal{A}$ relative to f and x, and assume by contradiction that

for all $i = 1, \ldots, n$ $[l_i, u_i] \not\subseteq [l, u]$. Since D and $\mathcal{A}$ are interval safe, we have that $[l_i, u_i] \cap [l, u] = \emptyset$. Then, by condition 4a, it follows that $\mathcal{J}_\mathcal{A}$ that does not satisfies $f : [l, u]$, a contradiction.

Case $D = f : [l, u]$, where f is functional. By construction, that all canonical models of $\mathcal{A}$ satisfy $f : [l, u]$ means that both $\hat{l} = \max_{f:[l,u](x)\in\mathcal{A}} l$ and $\hat{u} = \min_{f:[l,u](x)\in\mathcal{A}} u$ lie in the interval $[l, u]$. Then, on the one hand, being $\mathcal{A}$ clash-free ensures that $\hat{l} \leq \hat{u}$ and hence $[\hat{l}, \hat{u}]$ is an interval contained in $[l, u]$. On the other hand, since $\mathcal{A}$ is complete, $f : [\hat{l}, \hat{u}](x)$ occurs in $\mathcal{A}$ and hence $KB, \mathcal{A} \models f : [l, u](x)$.

Cases $D = D_1 \sqcap D_2$, $D = D_1 \sqcup D_2$, $D = \exists R.D'$ and $D = (R_1 \circ \cdots \circ R_n = S_1 \circ \cdots \circ S_m)$ follow directly from the definitions and the induction hypothesis. □

Lemma 5. *Let $\mathcal{A}$ be clash-free and complete w.r.t. KB and D a concept s.t. D and $\mathcal{A}$ are interval safe. Then, $x \in \langle\!\langle D \rangle\!\rangle_\mathcal{A}$ iff $\mathcal{J}_\mathcal{A}, x \models D$, for all canonical models $\mathcal{J}_\mathcal{A}$.*

Proof. The proof is by induction on the structure of D.

Case $D = A$. In this case $\langle\!\langle D \rangle\!\rangle_\mathcal{A} = A^\mathcal{J}_\mathcal{A}$, for all canonical models $\mathcal{J}_\mathcal{A}$.

Case $D = D_1 \sqcap D_2$. We have that $\mathcal{J}_\mathcal{A}, x \models D_1 \sqcap D_2$, for all $\mathcal{J}_\mathcal{A}$ iff $\mathcal{J}_\mathcal{A}, x \models D_1$, for all $\mathcal{J}_\mathcal{A}$, and $\mathcal{J}_\mathcal{A}, x \models D_2$, for all $\mathcal{J}_\mathcal{A}$. By induction, this holds iff $x \in \langle\!\langle D_1 \rangle\!\rangle_\mathcal{A} \cap \langle\!\langle D_2 \rangle\!\rangle_\mathcal{A}$ and hence $x \in \langle\!\langle D_1 \sqcap D_2 \rangle\!\rangle_\mathcal{A}$.

Analogously, the cases when $D = D_1 \sqcup D_2$, $D = \exists R.D'$ and $D = (R_1 \circ \cdots \circ R_n = S_1 \circ \cdots \circ S_m)$ follow directly from the definitions applying the induction hypothesis.

Case $D = f : [l, u]$. For the left-to-right direction, $x \in \langle\!\langle f : [l, u] \rangle\!\rangle_\mathcal{A}$ means that there exists some l', u' s.t. $l \leq l' \leq u' \leq u$ and $f : [l', u'](x) \in \mathcal{A}$. However, this ensures that $KB, \mathcal{A} \models f : [l, u](x)$ and by Lemma 4 it follows that $\mathcal{J}_\mathcal{A}, x \models f : [l, u]$, for all canonical models of $\mathcal{A}$. For the opposite direction, assume that all canonical models satisfy $f : [l, u]$ in x. As we know from Lemma 4, whether f is functional or not, this ensures that there exists an assertion $f : [l', u'](x)$ in $\mathcal{A}$ with $[l', u'] \subseteq [l, u]$. By construction follows that $x \in \langle\!\langle f : [l, u] \rangle\!\rangle_\mathcal{A}$. □

The following theorem shows that the combined subsumption checking algorithm is correct and complete.

Theorem 1. *Let KB be a $\mathcal{PL}^+$ knowledge base, C a simple concept and D a concept s.t. C and D are interval safe. Then, $KB \models C \sqsubseteq D$ iff the tableaux $\mathcal{A}$ of C contains a clash or $x_0 \in \langle\!\langle D \rangle\!\rangle_\mathcal{A}$.*

Proof. For the left-to-right direction, assume by contraposition that $\mathcal{A}$ is clash-free and $x_0 \notin \langle\!\langle D \rangle\!\rangle_\mathcal{A}$. By Lemma 5, we have that $\mathcal{J}_\mathcal{A}, x_0 \not\models D$. Then, by Lemmas 3 and 1, $\mathcal{J}_\mathcal{A}$ is a model of KB and $\mathcal{J}_\mathcal{A}, x_0 \models C$. It follows that $KB \not\models C \sqsubseteq D$.

For the opposite direction, if $\mathcal{A}$ contains a clash, by Lemma 1 no model of KB satisfies C. Hence, $KB \models C \sqsubseteq D$ vacuously holds. Finally, assume that $\mathcal{A}$ is clash-free and $x_0 \in \langle\!\langle D \rangle\!\rangle_\mathcal{A}$. By Lemmas 5 and 3 we have $KB, \mathcal{A} \models D$. Let $\mathcal{I}$ be a model of KB s.t. $\mathcal{I}, d \models C$. Clearly, by assigning x_0 to d, $\mathcal{I}, d \models C(x_0)$ holds and hence $\mathcal{I}, d \models \mathcal{A}$ (Lemma 1). Then, $\mathcal{I}$ is a model of KB and, since $\mathcal{I}, d \models \mathcal{A}$, there exists an assignment π s.t. $\mathcal{I}, \pi \models \mathcal{A}$. Since $KB, \mathcal{A} \models D$ it follows that $\mathcal{I}, d \models D$ □

Finally, we show that tableaux-based expansion in the second phase together with the structural subsumption in the third phase can be performed in polynomial time.

Theorem 2. *Let KB be a $\mathcal{PL}^+$ knowledge base, C a simple concept and D a concept s.t. C and D are interval safe. Subsumption checking $KB \models C \sqsubseteq D$ is PTIME.*

Proof. With respect to the tableaux construction, we show that the resulting Abox $\mathcal{A}$ is polynomial in the size of KB and C. First of all, the number of variable names occurring in $\mathcal{A}$ depends on the application of $\rightarrow_\exists$ and $\rightarrow_=$ rules. Since rule $\rightarrow_\exists$ generates a single variable name, rule $\rightarrow_=$ generates $n + m - 1$ variables at a time and the total number of their applications is bounded by $|C|$, it follows that $\mathsf{VR}_\mathcal{A}$ is proportional to $|C|$.

Being $\mathsf{VR}_\mathcal{A} \propto |C|$, it is straightforward to see that rule $\rightarrow_\sqcap$ can be applied at most $|C|$ times whereas the number of possible applications of rules $\rightarrow_\sqsubseteq$, $\rightarrow_{\mathsf{dom}}$, $\rightarrow_{\mathsf{ran}}$, $\rightarrow_{\mathsf{func}(R)}$, and $\rightarrow_{\mathsf{disj}}$ is bounded by $|C| \cdot |KB|$. Finally, the applications of $\rightarrow_{\mathsf{func}(f)}$ and $\rightarrow_{f_\perp}$ are at most $|C|^3 \cdot |KB|$. Then, the size of the resulting tableaux is polynomial in the size of KB and C and hence the second phase of the subsumption algorithm is performed in polynomial time.

Then, checking $x_0 \in \langle\langle D \rangle\rangle$ incrementally updates the extension of the subconcepts of D over $\mathcal{A}$. Consequently, the third phase can be performed in $O(|\mathcal{A}| \cdot |D|)$. □

As said above, in case C is not simple, the preprocessing reduces it in a normal form $C_1 \sqcup \cdots \sqcup C_n$ that can be exponentially longer than C. This in particular may occur if C and D are not interval safe, due to introduction of disjunctions in the interval normalization (1), thus making the subsumption checking (see also Theorem 7 in [5]).

However, the subsumption algorithm can be optimized in several ways in order to improve performance. A first optimization concerns the interval normalization. Clearly, the number of resulting interval splits depend on the set of distinct end-points occurring in D. We can reduce this set by using the following syntactic replacements in D:

$$f[l_1, u_1] \sqcup f[l_2, u_2] \rightsquigarrow f[\min\{l_1, l_2\}, \max\{u_1, u_2\}] \quad \text{if } [l_1, u_1] \cup [l_2, u_2] \text{ is an interval} \tag{2}$$

$$f[l_1, u_1] \sqcap f[l_2, u_2] \rightsquigarrow f[\bar{l}, \bar{u}] \quad \text{if } \bar{l} \leq \bar{u} \text{ and } f \text{ is functional}. \tag{3}$$

where $\bar{l} = \max\{l_1, l_2\}$ and $\bar{u} = \min\{u_1, u_2\}$.

Standard optimizations can also be used to speed up the tablaux expansion. For instance, rules $\rightarrow_\exists$ and $\rightarrow_{\mathsf{func}(R)}$ can be merged as follows: given $\exists R.C(x)$, if R is not functional or x has no R-successor, then $\rightarrow_\exists$ is applied as usual. Otherwise, C is added to the (single) R-successor of x. The rule $\rightarrow_{\mathsf{func}(f)}$ can as well be optimized. Let $f : [l_1, u_1], \ldots, f : [l_n, u_n]$ be the interval constrains occuring in C that are assigned to a variable x, $\bar{l}$ the maximum lower bound of $\{l_1, \ldots, l_n\}$ and $\bar{u}$ the minimum upper bound of $\{u_1, \ldots, u_n\}$. It is easy to see that either $\bar{u} < \bar{l}$, that is C is inconsistent w.r.t. KB, or $[\bar{l}, \bar{u}]$ is an interval that by construction is contained in all $[l_i, u_i]$. Therefore, we can replace $f : [l_1, u_1](x), \ldots, f : [l_n, u_n](x)$ in $\mathcal{A}$ with a single assertion $f : [\bar{l}, \bar{u}](x)$.

4 Discussion

We have introduced the description logic $\mathcal{PL}^+$ which allows to formalize the data usage policies adopted by controllers as well as the consent to data processing granted by data subjects. Checking whether the controllers' policies comply with the available consent boils down to subsumption checking between $\mathcal{PL}^+$ concepts.

The frequency of compliance checks can be high, so $\mathcal{PL}^+$ has been designed to address scalability requirements by making the language as simple as possible. As a consequence, the ontology and the query languages are notably different from each other. In particular, the ontology language largely intersects some fragments of the EL and DL-lite families of DLs [2, 4, 1]. Conversely, the query language supports disjunctions, role chain agreements, and non-convex interval constraints in order to model limitations on data storage duration. These features place $\mathcal{PL}^+$ outside the space of Horn DLs [10, 11] – and in particular the tractable profiles of OWL2.

We also presented a subsumption algorithm which, in broad terms, reminds the structural subsumption algorithm for the description logic underlying CLASSIC [9]. However, CLASSIC supports neither concept unions ($\sqcup$) nor qualified existential restrictions ($\exists R.C$). On the other hand, CLASSIC additionally supports number restrictions and qualified universal restrictions (that strictly generalize the range restrictions of $\mathcal{PL}^+$), which make it incomparable to $\mathcal{PL}^+$.

To the best of our knowledge all previous encodings of policies in KR languages [14, 12, 6], focus on trust management and access control, rather than data usage control. As a consequence, those languages lack the terms for expressing privacy-related and usage-related concepts. A crucial drawback is that the main reasoning task in those frameworks is permission checking; policy comparison (which is central to our work) is not considered. Both Rei and Protune [12, 6] support logic program rules. However, if rules are recursive the policy comparison results undecidable, otherwise, it is NP-hard. Similar considerations apply to KAoS [14], a framework based on a DL that, in general, is not tractable, and supports role-value maps that in general make reasoning undecidable (see [3], Ch. 5). The authors do not discuss how to avoid this issue. In comparison, checking compliance between $\mathcal{PL}^+$ policies is tractable if there is a constant bound on the number of disjuncts occurring in a normalized concept.

As future work, we are planning to implement the presented algorithm and to assess its scalability experimentally against real data usage policies provided by SPECIAL's industrial partners. Moreover, we are planning to study three further extensions of $\mathcal{PL}$ motivated by SPECIAL's use cases. Two of them are needed by a novel, dynamic method for collecting consent. This method is based on fine-grained authorizations, where the data categories involved consist of single data points (e.g. a specific location). This can be achieved by introducing nominals in $\mathcal{PL}$. Moreover, the data subject should be able to deny consent for some of such specific authorizations, which amounts to introducing some form of negation in the policy language. The third extension is needed to support so-called *sticky policies* [13], that constitute a sort of a license that applies to the data released to third parties. The recipient is allowed to use the data according to the sticky policy. In order to apply the same sticky policy transitively to arbitrary sequences of data transfers, a construct similar to fixpoints is needed (see [8, 7] for an extension of description logics with fixpoints).

Acknowledgments

This research is funded by the European Unions Horizon 2020 research and innovation programme under grant agreement N. 731601.

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. F. Baader, C. Lutz, and S. Brandt. Pushing the EL envelope further. In K. Clark and P. F. Patel-Schneider, editors, *Proceedings of the Fourth OWLED Workshop on OWL: Experiences and Directions, Washington, DC, USA, 1-2 April 2008.*, volume 496 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.
5. 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.
6. P. A. Bonatti, J. L. D. Coi, D. Olmedilla, and L. Sauro. A rule-based trust negotiation system. *IEEE Trans. Knowl. Data Eng.*, 22(11):1507–1520, 2010.
7. P. A. Bonatti, C. Lutz, A. Murano, and M. Y. Vardi. The complexity of enriched mu-calculi. *Logical Methods in Computer Science*, 4(3), 2008.
8. P. A. Bonatti and A. Peron. On the undecidability of logics with converse, nominals, recursion and counting. *Artif. Intell.*, 158(1):75–96, 2004.
9. A. Borgida and P. F. Patel-Schneider. A semantics and complete algorithm for subsumption in the CLASSIC description logic. *J. Artif. Intell. Res.*, 1:277–308, 1994.
10. D. Carral, C. Feier, B. C. Grau, P. Hitzler, and I. Horrocks. *EL*-ifying ontologies. In S. Demri, D. Kapur, and C. Weidenbach, editors, *Automated Reasoning - 7th International Joint Conference, IJCAR 2014, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 19-22, 2014. Proceedings*, volume 8562 of *Lecture Notes in Computer Science*, pages 464–479. Springer, 2014.
11. D. Carral, C. Feier, B. C. Grau, P. Hitzler, and I. Horrocks. Pushing the boundaries of tractable ontology reasoning. In P. Mika, T. Tudorache, A. Bernstein, C. Welty, C. A. Knoblock, D. Vrandecic, P. T. Groth, N. F. Noy, K. Janowicz, and C. A. Goble, editors, *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part II*, volume 8797 of *Lecture Notes in Computer Science*, pages 148–163. Springer, 2014.
12. L. Kagal, T. W. Finin, and A. Joshi. A policy language for a pervasive computing environment. In *4th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY)*, pages 63–, Lake Como, Italy, June 2003. IEEE Computer Society.
13. S. Pearson and M. C. Mont. Sticky policies: An approach for managing privacy across multiple parties. *IEEE Computer*, 44(9):60–68, 2011.
14. A. Uszok, J. M. Bradshaw, R. Jeffers, N. Suri, P. J. Hayes, M. R. Breedy, L. Bunch, M. Johnson, S. Kulkarni, and J. Lott. KAoS policy and domain services: Towards a description-logic approach to policy representation, deconfliction, and enforcement. In *4th IEEE International Workshop on Policies for Distributed Systems and Networks (POLICY)*, pages 93–96, Lake Como, Italy, June 2003. IEEE Computer Society.