On the Web Ontology Rule Language OWL 2 RL

Son Thanh Cao[1], Linh Anh Nguyen[2], and Andrzej Szałas[2,3]

[1] Faculty of Information Technology, Vinh University
182 Le Duan street, Vinh, Nghe An, Vietnam
sonct@vinhuni.edu.vn
[2] Institute of Informatics, University of Warsaw
Banacha 2, 02-097 Warsaw, Poland
{nguyen,andsz}@mimuw.edu.pl
[3] Dept. of Computer and Information Science, Linköping University
SE-581 83 Linköping, Sweden

Abstract. It is known that the OWL 2 RL Web Ontology Language Profile has PTime data complexity and can be translated into Datalog. However, the result of translation may consist of a Datalog program and a set of constraints in the form of negative clauses. Therefore, a knowledge base in OWL 2 RL may be unsatisfiable. In the current paper we first identify a maximal fragment of OWL 2 RL, called OWL 2 RL$^+$, with the property that every knowledge base expressed in OWL 2 RL$^+$ can be translated to a Datalog program and hence is satisfiable. We then propose some extensions of OWL 2 RL and OWL 2 RL$^+$ that still have PTime data complexity.

1 Introduction

Semantic Web is a rapidly growing research area that has received lots of attention in the last decade. As Semantic Web deals with ontologies and intelligent software agents distributed over the Internet, it overlaps with the research area of computational collective intelligence. One of the layers of Semantic Web is OWL (Web Ontology Language), which is used to specify knowledge of the domain in terms of concepts, roles and individuals. The second version OWL 2 of OWL, recommended by W3C in 2009, is based on the description logic $\mathcal{SROIQ}$ [17]. This logic is highly expressive but has intractable combined complexity (N2ExpTime-complete) and data complexity (NP-hard) for basic reasoning problems. Thus, W3C also recommended profiles OWL 2 EL, OWL 2 QL and OWL 2 RL, which are restricted sublanguages of OWL 2 Full with PTime data complexity. These profiles are based on the families of description logics $\mathcal{EL}$ [2, 3], DL-Lite [5] and DLP (Description Logic Programs) [15], respectively.

In the current paper we concentrate on OWL 2 RL. To achieve PTime data complexity of computing queries, OWL 2 RL restricts the full language OWL 2. The accepted restrictions ensure a translation into Datalog, where purely negative clauses are allowed. It is well-known that the data complexity of Datalog is PTime [1], so the data complexity of OWL 2 RL is also guaranteed to be PTime. Moreover, efficient computational methods designed for Datalog can immediately be applied.

1.1 Motivation and Contributions

Knowledge bases in OWL 2 RL may be unsatisfiable (that is, inconsistent), since their translations into Datalog may also need negative clauses as constraints. Moreover, OWL 2 RL can be extended in various directions without losing its PTime data complexity. That is, on the one hand, OWL 2 RL is too expressive as it may lead to unsatisfiable knowledge bases. On the other hand, it can be made more expressive. Therefore in the current paper we consider the following issues:

1. how to restrict OWL 2 RL so that knowledge bases are always satisfiable;
2. how to extend such restricted OWL 2 RL so that both satisfiability of knowledge bases and tractability of computing queries are preserved.

Unsatisfiability of knowledge bases is a serious issue. OWL 2 RL reasoners provide a functionality to check satisfiability of knowledge bases and even find the sources of inconsistency. However, it is still desirable to identify in OWL 2 RL features used for constructing positive (definite) rules as well as features used for constructing negative clauses as constraints. There are two reasons:

1. when a given knowledge base is consistent, negative clauses do not participate in drawing "positive conclusions", so the ontology engineer may want to use syntactic restrictions to guarantee consistency;
2. the departure point in Datalog-like languages are programs consisting of non-negative clauses only; based on such programs one can introduce negation in bodies of rules, like in stratified Datalog$^\neg$ as well as Datalog$^\neg$ with well-founded semantics [1]; similarly, one can develop variants of OWL 2 RL with nonmonotonic semantics and PTime data complexity starting from the fragment of OWL 2 RL without constraints.

For simplicity, when specifying OWL 2 RL we ignore the predefined data types and call the resulting logical formalism OWL 2 RL$_0$. In this paper, we achieve the following goals:

- we identify a maximal fragment of OWL 2 RL$_0$, called OWL 2 RL$^+$, with the property that every knowledge base expressed in OWL 2 RL$^+$ can be translated to a Datalog program without negative clauses and hence is satisfiable;
- we prove that whenever a knowledge base *KB* in OWL 2 RL$_0$ is satisfiable then its corresponding version in OWL 2 RL$^+$ is equivalent to *KB* w.r.t. positive queries;[4]
- we propose some natural extensions of OWL 2 RL$_0$ and OWL 2 RL$^+$ (respectively denoted by OWL 2 eRL and OWL 2 eRL$^+$); the ideas behind these extensions are natural and ideas around them may have been known earlier, but here we formalize them and prove that both OWL 2 eRL and OWL 2 eRL$^+$ have PTime data complexity, and that every knowledge base in OWL 2 eRL$^+$ can be translated to a knowledge base without negative clauses in eDatalog, an extension of Datalog;
- we extend both OWL 2 eRL and OWL 2 eRL$^+$ with eDatalog itself; combining OWL 2 eRL or OWL 2 eRL$^+$ with eDatalog gives one the freedom to use the syntax of both languages and allows one to represent knowledge not only in terms of concepts and roles but also by predicates of higher arities.

1.2 Related Work

This work is a revised and extended version of our conference paper [6]. Comparing to [6], we extend discussions and additionally provide full proofs of the results.

OWL 2 RL has been inspired by Description Logic Programs (DLP) [15] and pD^* [37] (see [34]). The logical base of DLP is the description Horn logic DHL [15].

[4] That is, ignoring constraints and considering only positive queries, OWL 2 RL$_0$ can be replaced by OWL 2 RL$^+$.

Some extensions of DHL were considered in [30]. The pD^* semantics [37] is a precursor for OWL 2 RL and for work on supporting OWL through Horn fragments.

A number of Horn fragments of DLs with PTime data complexity have also been investigated in [5, 15, 19, 21, 23, 31, 32, 35]. The combined complexities of Horn fragments of DLs were considered, amongst others, in [22]. Some tractable Horn fragments of DLs without ABoxes have also been isolated in [2, 4]. The work [32] studies Horn fragments of the DLs $\mathcal{SHOIQ}$ and $\mathcal{SROIQ}$. This Horn-$\mathcal{SROIQ}$ fragment is expressive, but does not extend OWL 2 RL as it does not allow for data roles and restricts role inclusion axioms by regularity conditions. For an overview of most of these works see [31, Section 4].

Various combinations of rule languages with description logics have been studied in a considerable number of works, including [8] (on *$\mathcal{AL}$-log*), [24] (on *CARIN*), [27] (on *DL-safe rules*), [36] (on *$\mathcal{DL}$+log*), [20, 26] (on *hybrid MKNF*), [9] (on *hybrid programs*), [12] (on *dl-programs*), [7] (on WORL). Among these works, only [7] directly deals with OWL 2 RL. In that work we have considered a combination of a variant of OWL 2 RL with eDatalog$^\neg$.

Some other related results are [18] (on SWRL), [16] (on description logic programs with negation), [10] (on layered rule-based architecture) and [11, 28, 29] (on Horn fragments of modal logics).

1.3 The Structure of This Paper

The rest of this paper is structured as follows. In Section 2 we specify the logical formalism OWL 2 RL_0. Section 3 is devoted to OWL 2 RL^+. Section 4 presents extensions of OWL 2 RL_0 and OWL 2 RL^+. Section 5 concludes this work. Proofs of the results of this paper are presented in the appendix.

2 A Logical Formalism of OWL 2 RL

In this section we specify OWL 2 RL as a description logic-based formalism. We focus on logical aspects of this language while ignoring the concrete data types predefined for OWL 2 RL [34]. In particular, we assume that considered knowledge bases are type-correct. We call the resulting formalism OWL 2 RL_0. The semantics of OWL 2 RL_0 follows the "direct semantics" of OWL 2 [14].

In addition to notation listed in Table 1 (page 4), we shall use the following notational convention:

- CNames stands for the set of *concept names*;
- RNames stands for the set of *role names*;
- INames stands for the set of *individual names*.

The syntax of families R, DR, lC, rC, eC is defined in Figure 1.[5]
We also use abbreviations: Disj (Disjoint), Func (Functional), $\mathsf{InvFunc}$ (InverseFunctional), Refl (Reflexive), Irref (Irreflexive), Sym (Symmetric), Asym (Asymmetric), Trans (Transitive), Key (HasKey).

Definition 2.1.

- *A* TBox axiom*, standing for a ClassAxiom or a DatatypeDefinition or a HasKey axiom [34], is an expression of one of the following forms:*

$$lC \sqsubseteq rC,\ eC \equiv eC',\ \mathsf{Disj}(lC_1, \ldots, lC_k),\ DT \equiv DR,$$
$$\mathsf{Key}(lC, R_1, \ldots, R_k, \sigma_1, \ldots, \sigma_h).$$

[5] In comparison to [6], the definitions of lC, rC and eC are extended with $\bot$.

Table 1. Correspondences between logical notation and the notation used in [34].

Logical notation	Notation of [34]
$\top$ (truth)	*owl:Thing*
$\bot$ (falsity)	*owl:Nothing*
a, b	*individuals* (i.e. *objects*)
d	a *literal* (i.e. a data constant)
A, B	concept names (i.e., *Class* elements)
C, D	*concepts* (i.e., *ClassExpression* elements)
lC	a concept standing for a *subClassExpression*
rC	a concept standing for a *superClassExpression*
eC	a concept standing for an *equivClassExpression*
DT	a *data type* (i.e., a *Datatype*)
DR	a *data range* (i.e., a *DataRange*)
r, s	*object role names* (i.e., *ObjectProperty* elements)
R, S	*object roles* (i.e., *ObjectPropertyExpression* elements)
σ, ϱ	*data role names* (i.e., *DataProperty* elements)
$\{a_1\} \sqcup \ldots \sqcup \{a_k\}$	the class constructor *ObjectOneOf*

$$\begin{aligned}
R &:= r \mid r^- \\
DR &:= DT \mid DT \sqcap DR \\
lC &:= \bot \mid A \mid \{a\} \mid lC \sqcap lC \mid lC \sqcup lC \mid \exists R.lC \mid \exists R.\top \mid \exists \sigma.DR \mid \exists \sigma.\{d\} \\
rC &:= \bot \mid A \mid rC \sqcap rC \mid \neg lC \mid \forall R.rC \mid \exists R.\{a\} \mid \forall \sigma.DR \mid \exists \sigma.\{d\} \mid \\
&\qquad \leq 1\, R.lC \mid \leq 0\, R.lC \mid \leq 1\, R.\top \mid \leq 0\, R.\top \mid \leq 1\, \sigma.DR \mid \leq 0\, \sigma.DR \\
eC &:= \bot \mid A \mid eC \sqcap eC \mid \exists R.\{a\} \mid \exists \sigma.\{d\}
\end{aligned}$$

Fig. 1. The BNF grammar for families R, DR, lC, rC and eC.

– *An* RBox axiom*, standing for an ObjectPropertyAxiom or a DataPropertyAxiom [34], is an expression of one of the following forms:*[6]

$$\begin{gathered}
R_1 \circ \ldots \circ R_k \sqsubseteq S,\ R \equiv S,\ R \equiv S^-, \\
\mathsf{Disj}(R_1, \ldots, R_k),\ \exists R.\top \sqsubseteq rC,\ \top \sqsubseteq \forall R.rC, \\
\mathsf{Func}(R),\ \mathsf{InvFunc}(R),\ \mathsf{Irref}(R),\ \mathsf{Sym}(R),\ \mathsf{Asym}(R),\ \mathsf{Trans}(R), \\
\sigma \sqsubseteq \varrho,\ \sigma \equiv \varrho,\ \mathsf{Disj}(\sigma_1, \ldots, \sigma_k),\ \exists \sigma \sqsubseteq rC,\ \top \sqsubseteq \forall \sigma.DR,\ \mathsf{Func}(\sigma).
\end{gathered}$$ □

Table 2 lists some correspondences between RBoxes axioms expressed in logical notation and the notation of [34]. One can classify these axioms as TBox axioms instead of RBox axioms. Similarly, $\mathsf{Key}(\ldots)$ axioms can be classified as RBox axioms instead.

Definition 2.2. *An* ABox assertion *is a formula of one of the following forms:*

$$a \approx b,\ \ a \not\approx b,\ \ rC(a),\ \ DT(d),\ \ r(a,b),\ \ \neg r(a,b),\ \ \sigma(a,d),\ \ \neg\sigma(a,d).$$

We call an ABox assertion also as an ABox axiom. □

[6] Axioms of the form $R \equiv S$, $R \equiv S^-$, $\mathsf{Sym}(R)$ or $\mathsf{Trans}(R)$ are expressible by axioms of the form $R_1 \circ \ldots \circ R_k \sqsubseteq S$, so can be deleted from this list.

Table 2. Correspondences between axioms expressed in logical notation and the notation used in [34].

Logical notation	Notation of [34]
$\exists R.\top \sqsubseteq rC$	*ObjectPropertyDomain*
$\top \sqsubseteq \forall R.rC$	*ObjectPropertyRange*
$\exists \sigma \sqsubseteq rC$	*DataPropertyDomain*
$\top \sqsubseteq \forall \sigma.DR$	*DataPropertyRange*

Note that:

- assertions of the form $DT(d)$ are implicitly provided in OWL 2 RL [34] by declarations of DT and d;
- the other ABox assertions listed in Definition 2.2 stand for *Assertion* elements of [34];
- in OWL 2 RL [34] there are also declaration and annotation axioms used for expressing meta information about ontologies; these kinds of axioms are inessential from the logical point of view and are omitted here.

Definition 2.3. *An* RBox *(respectively,* TBox*,* ABox*) is a finite set of RBox (respectively, TBox, ABox) axioms. An ABox is* extensionally reduced *if it does not contain axioms of the form $C(a)$ with C being a complex concept (i.e., not a concept name).*

A knowledge base *(i.e., an* ontology*) in OWL 2 RL$_0$ is defined to be a tuple $\langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$ consisting of an RBox $\mathcal{R}$, a TBox $\mathcal{T}$, and an ABox $\mathcal{A}$.*[7] *We may present a knowledge base as a set of axioms.* □

Let us now define interpretations.

Definition 2.4. *An* interpretation $\mathcal{I} = \langle \Delta_o^\mathcal{I}, \Delta_d^\mathcal{I}, \cdot^\mathcal{I} \rangle$ *consists of a non-empty set* $\Delta_o^\mathcal{I}$ *called the* object domain *of* $\mathcal{I}$*, a non-empty set* $\Delta_d^\mathcal{I}$ *disjoint from* $\Delta_o^\mathcal{I}$*, called the* data domain *of* $\mathcal{I}$*, and a function* $\cdot^\mathcal{I}$ *called the* interpretation function *of* $\mathcal{I}$*, which maps:*

- *every individual a to an element $a^\mathcal{I} \in \Delta_o^\mathcal{I}$;*
- *every literal d to an element $d^\mathcal{I} \in \Delta_d^\mathcal{I}$;*
- *every concept name A to a subset $A^\mathcal{I}$ of $\Delta_o^\mathcal{I}$;*
- *every data type DT to a subset $DT^\mathcal{I}$ of $\Delta_d^\mathcal{I}$;*
- *every object role name r to a binary relation $r^\mathcal{I} \subseteq \Delta_o^\mathcal{I} \times \Delta_o^\mathcal{I}$;*
- *every data role name σ to a binary relation $\sigma^\mathcal{I} \subseteq \Delta_o^\mathcal{I} \times \Delta_d^\mathcal{I}$.* □

It is expected that, when an ontology is loaded, appropriate preprocessing is done to standardize literals. For example, literals 1 (of type "integer"), 1.0, 1.00 in expressions of type "decimal" should be represented by the same value. Assuming that such a standardization has been done for the considered knowledge base in OWL 2 RL$_0$, we adopt the Unique Names Assumption for literals, i.e.,

if $d_1 \neq d_2$ then we assume that $d_1^\mathcal{I} \neq d_2^\mathcal{I}$, too.

This assumption is suitable for OWL 2 RL$_0$, as OWL 2 RL$_0$ does not deal with predefined data types.

The interpretation function is extended to interpret data ranges, inverse object roles and complex concepts as shown in Figure 2.

---

[7] One can convert a knowledge base to the form with an extensionally reduced ABox by replacing every ABox assertion $rC(a)$ by an ABox assertion $A(a)$ and a TBox axiom $A \sqsubseteq rC$, where A is a new concept name.

$$\{d\}^{\mathcal{I}} = \{d^{\mathcal{I}}\},\ \ (DT \sqcap DR)^{\mathcal{I}} = DT^{\mathcal{I}} \cap DR^{\mathcal{I}}$$
$$(R^-)^{\mathcal{I}} = (R^{\mathcal{I}})^{-1} = \{(y,x) \mid (x,y) \in R^{\mathcal{I}}\}$$
$$\top^{\mathcal{I}} = \Delta_o^{\mathcal{I}},\ \ \bot^{\mathcal{I}} = \emptyset,\ \ \{a\}^{\mathcal{I}} = \{a^{\mathcal{I}}\},\ \ (\neg C)^{\mathcal{I}} = \Delta_o^{\mathcal{I}} \setminus C^{\mathcal{I}}$$
$$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}},\ \ (C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$$
$$(\forall R.C)^{\mathcal{I}} = \{x \in \Delta_o^{\mathcal{I}} \mid \forall y[(x,y) \in R^{\mathcal{I}} \text{ implies } y \in C^{\mathcal{I}}]\}$$
$$(\exists R.C)^{\mathcal{I}} = \{x \in \Delta_o^{\mathcal{I}} \mid \exists y[(x,y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}]\}$$
$$(\forall \sigma.DR)^{\mathcal{I}} = \{x \in \Delta_o^{\mathcal{I}} \mid \forall y[(x,y) \in \sigma^{\mathcal{I}} \text{ implies } y \in DR^{\mathcal{I}}]\}$$
$$(\exists \sigma.\varphi)^{\mathcal{I}} = \{x \in \Delta_o^{\mathcal{I}} \mid \exists y[(x,y) \in \sigma^{\mathcal{I}} \text{ and } y \in \varphi^{\mathcal{I}}]\}$$
$$(\exists \sigma)^{\mathcal{I}} = \{x \in \Delta_o^{\mathcal{I}} \mid \exists y\,(x,y) \in \sigma^{\mathcal{I}}\}$$
$$(\leq n\, R.C)^{\mathcal{I}} = \{x \in \Delta_o^{\mathcal{I}} \mid \#\{y \in \Delta_o^{\mathcal{I}} \mid (x,y) \in R^{\mathcal{I}} \text{ and } y \in C^{\mathcal{I}}\} \leq n\}$$
$$(\leq n\, \sigma.DR)^{\mathcal{I}} = \{x \in \Delta_o^{\mathcal{I}} \mid \#\{y \in \Delta_d^{\mathcal{I}} \mid (x,y) \in \sigma^{\mathcal{I}} \text{ and } y \in DR^{\mathcal{I}}\} \leq n\}$$

Fig. 2. Interpretation of data ranges, inverse object roles, and complex concepts. We assume here that φ is of the form DR or $\{d\}$ and that $\#\Gamma$ denotes the cardinality of the set Γ.

From now on, if not stated otherwise, by an *axiom* we mean an RBox axiom, a TBox axiom or an ABox axiom.

Definition 2.5. *The satisfaction relation $\mathcal{I} \models \varphi$ between an interpretation $\mathcal{I}$ and an axiom φ is defined below and stands for "$\mathcal{I}$ validates φ":*

- $\mathcal{I} \models R_1 \circ \ldots \circ R_k \sqsubseteq S$ *iff* $R_1^{\mathcal{I}} \circ \ldots \circ R_k^{\mathcal{I}} \sqsubseteq S^{\mathcal{I}}$,
- $\mathcal{I} \models C \sqsubseteq D$ *iff* $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$,
- $\mathcal{I} \models C(a)$ *iff* $a^{\mathcal{I}} \in C^{\mathcal{I}}$,
- $\mathcal{I} \models r(a,b)$ *iff* $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in r^{\mathcal{I}}$,
- $\mathcal{I} \models \neg r(a,b)$ *iff* $(a^{\mathcal{I}}, b^{\mathcal{I}}) \notin r^{\mathcal{I}}$,
- $\mathcal{I} \models \sigma(a,d)$ *iff* $(a^{\mathcal{I}}, d^{\mathcal{I}}) \in \sigma^{\mathcal{I}}$,
- $\mathcal{I} \models \neg\sigma(a,d)$ *iff* $(a^{\mathcal{I}}, d^{\mathcal{I}}) \notin \sigma^{\mathcal{I}}$,
- $\mathcal{I} \models (\varphi \equiv \psi)$ *iff* $\varphi^{\mathcal{I}} = \psi^{\mathcal{I}}$,
  where φ and ψ may be of the form C, R, R^-, DT or DR,
- $\mathcal{I} \models a \approx b$ *iff* $a^{\mathcal{I}} = b^{\mathcal{I}}$,
- $\mathcal{I} \models a \not\approx b$ *iff* $a^{\mathcal{I}} \neq b^{\mathcal{I}}$,
- $\mathcal{I} \models \mathsf{Disj}(\varphi_1, \ldots, \varphi_k)$ *iff* $\varphi_i^{\mathcal{I}} \cap \varphi_j^{\mathcal{I}} = \emptyset$ *for all* $1 \leq i < j \leq k$,
  where $\varphi_1, \ldots \varphi_k$ are of the form C, R or σ,
- $\mathcal{I} \models \mathsf{Func}(R)$ *iff* $R^{\mathcal{I}}$ *is functional*
  (i.e. $\forall x,y,z(R^{\mathcal{I}}(x,y) \land R^{\mathcal{I}}(x,z) \to y = z)$),
- $\mathcal{I} \models \mathsf{InvFunc}(R)$ *iff* $R^{\mathcal{I}}$ *is inverse-functional*
  (i.e. $\forall x,y,z(R^{\mathcal{I}}(x,z) \land R^{\mathcal{I}}(y,z) \to x = y)$),
- $\mathcal{I} \models \mathsf{Irref}(R)$ *iff* $R^{\mathcal{I}}$ *is irreflexive,*
- $\mathcal{I} \models \mathsf{Sym}(R)$ *iff* $R^{\mathcal{I}}$ *is symmetric,*
- $\mathcal{I} \models \mathsf{Asym}(R)$ *iff* $R^{\mathcal{I}}$ *is asymmetric,*
- $\mathcal{I} \models \mathsf{Trans}(R)$ *iff* $R^{\mathcal{I}}$ *is transitive,*
- $\mathcal{I} \models \mathsf{Func}(\sigma)$ *iff* $\sigma^{\mathcal{I}}$ *is functional,*
- $\mathcal{I} \models \mathsf{Key}(C, R_1, \ldots, R_k, \sigma_1, \ldots, \sigma_h)$ *iff,*
  for every $a, b \in$ INames, $z_1, \ldots, z_k \in \Delta_o^{\mathcal{I}}$ and $d_1, \ldots, d_h \in \Delta_d^{\mathcal{I}}$,
  if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, $b^{\mathcal{I}} \in C^{\mathcal{I}}$, and
  $\{(a^{\mathcal{I}}, z_i), (b^{\mathcal{I}}, z_i)\} \subseteq R_i^{\mathcal{I}}$ for all $1 \leq i \leq k$, and
  $\{(a^{\mathcal{I}}, d_j), (b^{\mathcal{I}}, d_j)\} \subseteq \sigma_j^{\mathcal{I}}$ for all $1 \leq j \leq h$
  then $a^{\mathcal{I}} = b^{\mathcal{I}}$.

When φ is an ABox axiom, we also say $\mathcal{I}$ satisfies *φ to mean $\mathcal{I}$ validates φ.*

Let Γ be an RBox, a TBox or an ABox. An interpretation $\mathcal{I}$ is called a model *of Γ, denoted by $\mathcal{I} \models \Gamma$, if it validates all axioms of Γ. $\mathcal{I}$ is called a model of a knowledge base $\langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$, denoted by $\mathcal{I} \models \langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$, if it is a model of all $\mathcal{R}$, $\mathcal{T}$ and $\mathcal{A}$.* □

Definition 2.6. *A* (ground conjunctive) query *is a formula of the form $\varphi_1 \wedge \ldots \wedge \varphi_k$, where each φ_i is of one of the following forms:*

$$a \approx b, \;\; a \not\approx b, \;\; A(a), \;\; \neg A(a), \;\; r(a,b), \;\; \neg r(a,b), \;\; \sigma(a,d), \;\; \neg\sigma(a,d).$$

An interpretation $\mathcal{I}$ satisfies the query $\varphi = \varphi_1 \wedge \ldots \wedge \varphi_k$, which is denoted by $\mathcal{I} \models \varphi$, if $\mathcal{I} \models \varphi_i$ for all $1 \leq i \leq k$. We say that a query φ is a logical consequence *of a knowledge base $\langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$, denoted by $\langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle \models \varphi$, if every model of $\langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$ satisfies φ.* □

Note that queries are defined to be ground. In a more general context, one can allow queries to contain variables for individuals or literals, accepting the *range-restrictedness condition* stating that every variable occurring under negation occurs also in an atomic formula not under negation. However, one of the approaches to deal with such queries is to instantiate variables by individuals or literals occurring in the knowledge base or the query.

Definition 2.7. *The* data complexity *of OWL 2 RL$_0$ (for the ground conjunctive query answering problem) is the complexity of checking whether a query φ is a logical consequence of a knowledge base $\langle \mathcal{R}, \mathcal{T}, \mathcal{A} \rangle$, measured w.r.t. the size of the ABox $\mathcal{A}$, assuming that $\mathcal{A}$ is extensionally reduced and $\mathcal{R}$, $\mathcal{T}$ and φ are fixed.* □

3 The Fragment OWL 2 RL$^+$

In this section we first give some examples of unsatisfiable knowledge bases in OWL 2 RL$_0$. Next, we present the restricted version OWL 2 RL$^+$ of OWL 2 RL$_0$ ensuring that all knowledge bases are satisfiable. We also provide some important properties of OWL 2 RL$^+$.

Example 3.1. All the following knowledge bases in OWL 2 RL$_0$ are unsatisfiable:

$$\begin{aligned}
KB_1 &= \{A \equiv \bot,\ A(a)\},\\
KB_2 &= \{A \sqsubseteq \bot,\ A(a)\},\\
KB_3 &= \{A \sqsubseteq \neg B,\ A(a),\ B(a)\},\\
KB_4 &= \{A \sqsubseteq\ \leq 0\, r.B,\ A(a),\ r(a,b),\ B(b)\},\\
KB_5 &= \{A \sqsubseteq\ \leq 0\, r.\top,\ A(a),\ r(a,b)\},\\
KB_6 &= \{A \sqsubseteq\ \leq 0\, \sigma.DT,\ A(a),\ \sigma(a,d),\ DT(d)\},\\
KB_7 &= \{A \sqsubseteq\ \leq 1\, \sigma.DT,\ A(a),\ \sigma(a,d_1),\ DT(d_1),\ \sigma(a,d_2),\ DT(d_2)\},\\
&\quad\ \text{where } d_1 \neq d_2,\\
KB_8 &= \{\mathsf{Disj}(A,B),\ A(a),\ B(a)\},\\
KB_9 &= \{\mathsf{Disj}(r,s),\ r(a,b),\ s(a,b)\},\\
KB_{10} &= \{\mathsf{Disj}(\sigma,\sigma'),\ \sigma(a,d),\ \sigma'(a,d)\},\\
KB_{11} &= \{\mathsf{Irref}(r),\ r(a,a)\},\\
KB_{12} &= \{\mathsf{Irref}(r),\ s \sqsubseteq r,\ r \circ r \sqsubseteq r,\ s(a,b),\ r(b,a)\},
\end{aligned}$$

$KB_{13} = \{\mathsf{Asym}(r),\ r(a,b),\ r(b,a)\}$,
$KB_{14} = \{\mathsf{Asym}(r),\ s \sqsubseteq r,\ s(a,b),\ r(b,a)\}$,
$KB_{15} = \{a \not\approx b,\ a \approx b\}$,
$KB_{16} = \{a \not\approx b,\ A \sqsubseteq\ \leq 1\, r.B,\ A(c),\ r(c,a),\ B(a),\ r(c,b),\ B(b)\}$,
$KB_{17} = \{\neg r(a,b),\ r(a,b)\}$,
$KB_{18} = \{\neg r(a,b),\ s \sqsubseteq r,\ s(a,b)\}$,
$KB_{19} = \{\neg\sigma(a,d),\ \sigma(a,d)\}$.

Assuming that assertions of the forms $A(a)$, $r(a,b)$, $\sigma(a,d)$, $DT(d)$, $a \approx b$ are basic and should always be allowed, and that atomic concepts should be allowed at the left hand side of $\sqsubseteq$ in TBox axioms, then it is clear that the above knowledge bases are unsatisfiable. □

Definition 3.2. *We define OWL 2 RL$^+$ to be the restriction of OWL 2 RL$_0$ such that:*

- *the concept $\bot$ is disallowed;*[8]
- *the constructors $\neg lC$, $\leq 0\, R.lC$, $\leq 0\, R.\top$ and $\leq n\, \sigma.DR$ (where $n \in \{0,1\}$) are disallowed in the BNF grammar rule defining the rC family;*
- *axioms of the forms* Disj(...), Irref(R), Asym(R), *$a \not\approx b$, $\neg r(a,b)$, $\neg\sigma(a,d)$ are disallowed.* □

Restrictions listed in Definition 3.2 correspond to the following ones for OWL 2 RL [34]:

- the class *owl:Nothing* is disallowed;
- the grammar elements *superComplementOf*, *superObjectMaxCardinality* with limit 0, and *superDataMaxCardinality* are disallowed in the definition of *superClassProperty*;
- axioms of the following forms are disallowed:
  - *DisjointClasses*, *DisjointObjectProperties*, *DisjointDataProperties*,
  - *IrreflexiveObjectProperty*, *AsymmetricObjectProperty*,
  - *DifferentIndividuals*,
  - *NegativeObjectPropertyAssertion*, *NegativeDataPropertyAssertion*.

Definition 3.3. *A query is said to be* in the language of *KB if it does not use predicates not occurring in KB.*

A positive query *is a formula $\varphi_1 \land \ldots \land \varphi_k$, where each φ_i is of one of the forms $a \approx b$, $A(a)$, $r(a,b)$, $\sigma(a,d)$.* □

Let us now recall the definition of Datalog.

Definition 3.4.

- *A* term *is either a* constant *or a* variable*.*
- *If p is an n-argument predicate and $t_1, \ldots, t_n$ are terms then $p(t_1, \ldots, t_n)$ is an* atomic formula*, which is also called an* atom*.*
- *A* Datalog program clause *is a formula of the form $\varphi_1 \land \ldots \land \varphi_n \to \psi$, where $n \geq 0$ and $\varphi_1, \ldots, \varphi_n, \psi$ are atoms. The conjunction $\varphi_1 \land \ldots \land \varphi_n$ is called the* body *and ψ is called the* head *of the clause. The program clause is required to satisfy the* range-restrictedness *condition stating that every variable occurring in the clause's head must occur also in the clause's body.*

[8] In comparison to [6], we must add this restriction for OWL 2 RL$^+$ as we do now allow $\bot$ for OWL 2 RL$_0$.

– *A* Datalog program *is a finite set of Datalog program clauses.* □

Theorem 3.5.

1. *OWL 2 RL*$^+$ *is a maximal fragment (w.r.t. allowed features) of OWL 2 RL*$_0$ *such that every knowledge base expressed in the fragment is satisfiable.*
2. *Every knowledge base KB in OWL 2 RL*$^+$ *can be translated to a Datalog program* $\mathcal{P}$ *which is equivalent to KB in the sense that, for every query* φ *in the language of KB,* $KB \models \varphi$ *iff* $\mathcal{P} \models \varphi$. □

Definition 3.6. *Let KB be a knowledge base in OWL 2 RL*$_0$*. The* normal form *of KB is the knowledge base obtained from KB as follows: if* $\neg lC$ *occurs as an* rC *in the knowledge base then replace it by a fresh (new) concept name A and add to the knowledge base the TBox axiom* $A \sqcap lC \sqsubseteq \bot$.

The corresponding version *of KB* in OWL 2 RL$^+$ *is the knowledge base obtained from the normal form of KB by deleting all axioms containing* $\leq 0\, R.lC$, $\leq 0\, R.\top$ *or* $\leq n\, \sigma.DR$ *(where* $n \in \{0, 1\}$*) and deleting all axioms of the forms* $A \sqcap lC \sqsubseteq \bot$, $\mathsf{Disj}(\ldots)$, $\mathsf{Irref}(R)$, $\mathsf{Asym}(R)$, $a \not\approx b$, $\neg r(a, b)$, $\neg\sigma(a, d)$. □

Theorem 3.7. *Let KB be a knowledge base in OWL 2 RL*$_0$, KB' *be the normal form of KB, and* KB'' *be the corresponding version of KB in OWL 2 RL*$^+$*. Then:*

1. KB' *is equivalent to KB in the sense that, for every query* φ *in the language of KB,* $KB \models \varphi$ *iff* $KB' \models \varphi$;
2. *if KB is satisfiable and* φ *is a positive query in the language of KB then* $KB \models \varphi$ *iff* $KB'' \models \varphi$. □

The second assertion of Theorem 3.7 states that if *KB* is satisfiable then the corresponding version of *KB* in OWL 2 RL$^+$ is equivalent to *KB* w.r.t. positive queries. This means that, ignoring constraints and considering only positive queries, OWL 2 RL$_0$ can be replaced by OWL 2 RL$^+$ without any further loss of expressiveness.

4 Extensions of OWL 2 RL$_0$ with PTime Data Complexity

In this section we first define an extension of Datalog called eDatalog. We then propose an extension OWL 2 eRL of OWL 2 RL$_0$ with PTime data complexity, and an extension OWL 2 eRL$^+$ of OWL 2 RL$^+$ that can be translated into eDatalog. Next, we extend both OWL 2 eRL and OWL 2 eRL$^+$ with eDatalog.

4.1 eDatalog

From the point of view of OWL, there are two basic types: *individual* (i.e., *object*) and *literal* [34] (i.e., *data constant*). We denote the *individual* type by *IType*, and the *literal* type by *LType*. Thus,

– a concept name is a unary predicate of type $P(IType)$;
– a data type is a unary predicate of type $P(LType)$;
– an object role name is a binary predicate of type $P(IType \times IType)$;
– a data role name is a binary predicate of type $P(IType \times LType)$.

Extending OWL 2 RL$_0$ with Datalog, in addition to concept names and role names, we will also use:

– a set OPreds of *ordinary predicates* (including data types);

– a set ECPreds of *external checkable predicates*.

We assume that the sets CNames, RNames, OPreds and ECPreds are finite and pairwise disjoint. Let *DPreds* stand for the set of *defined predicates*,

$$\text{DPreds} = \text{CNames} \cup \text{RNames} \cup \text{OPreds}.$$

A k-argument predicate from OPreds has type $P(T_1 \times \ldots \times T_k)$, where each T_i is either *IType* or *LType*. A k-argument predicate from ECPreds has type $P(LType^k)$.

We assume that each predicate from ECPreds has a fixed meaning which is checkable in the sense that, if p is a k-argument predicate from ECPreds and $d_1, \ldots, d_k$ are constant elements of *LType*, then the truth value of $p(d_1, \ldots, d_k)$ is *fixed* and *computable in constant time*. For example, one may want to use the binary predicates $>$, $\geq$, $<$, $\leq$ on real numbers with the usual semantics.

We assume there are two different equality predicates $\approx$ and $\asymp$ (both belonging to OPreds), where $\approx$ has the type $P(IType \times IType)$ and $\asymp$ has the type $P(LType \times LType)$. These equality predicates have the standard semantics, with the Unique Names Assumption for literals (i.e., data constants).

While extending Datalog to eDatalog, we want to drop the range-restrictedness condition. However, to allow external checkable predicates we cannot do that totally. For this reason, we distinguish a subset RRPreds $\subseteq$ DPreds as the set of *range-restricted predicates*, which is required to contain both the equality predicates.

We define eDatalog as follows.

Definition 4.1.

– *A* term *is either an individual (of type IType) or a literal (of type LType) or a* variable *(of type IType or LType).*
– *If p is a predicate of type $P(T_1 \times \ldots \times T_k)$ and for $1 \leq i \leq k$, t_i is a term of type T_i, then $p(t_1, \ldots, t_k)$ is an* atomic formula *(also called an* atom*). An atom is* ground *if it contains no variables.*
– *An* eDatalog program clause *is a formula of the form $\varphi_1 \land \ldots \land \varphi_n \to \psi$, where $n \geq 0$ and $\varphi_1, \ldots, \varphi_n, \psi$ are atomic formulas such that:*
  • *ψ is an atom of a predicate from* DPreds*;*
  • *if the predicate of ψ belongs to* RRPreds *then every variable occurring in ψ occurs also in some φ_i whose predicate also belongs to* RRPreds*;*
  • *every variable occurring in some φ_i whose predicate belongs to* ECPreds *occurs also in some atom φ_j whose predicate belongs to* RRPreds.
– *An* eDatalog program *is a finite set of eDatalog program clauses.*
– *A* knowledge base in eDatalog *is a pair $\langle \mathcal{P}, \mathcal{A} \rangle$, where $\mathcal{P}$ is an eDatalog program and $\mathcal{A}$ is an* ABox *consisting of ground atoms of predicates from* DPreds. □

The notions for eDatalog like interpretation, model and data complexity are defined in the usual way, assuming the usual semantics for the equality predicates and the Unique Names Assumption for literals.

4.2 OWL 2 eRL and OWL 2 eRL$^+$

Axioms of the form $\mathsf{Refl}(R)$ (i.e., reflexive object property axioms) are disallowed for OWL 2 RL. Translating $\mathsf{Refl}(R)$ into Datalog we get a program clause $\forall x\, R(x, x)$ that violates the range-restrictedness condition, which seems to be the reason of this restriction. Similarly, $\top$ is disallowed as lC in OWL 2 RL. However, these restrictions are unnecessary. The Horn fragment of predicate logic without function symbols also has PTime data complexity. Furthermore, as shown in [25], evaluation methods of Datalog can be extended to Horn knowledge bases in predicate logic without function symbols. Therefore, we propose the following extensions of OWL 2 RL$_0$:

1. we allow *ReflexiveObjectProperty* axioms and $\top$ as lC;
2. we allow unary predicates from ECPreds to appear in the places of *DataRange* elements.

To motivate the second proposal let us indicate that it is desirable to express concepts like the class of all laptops with price not greater than 1000 USD. Using the syntax of description logic, the concept can be written as:

$$laptop \sqcap \exists price.(\leq 1000).$$

Here, "≤ 1000" is a unary predicate. Other useful predicates are, e.g., other comparison operators, string pattern matching operator and many other operators, used in programming languages or SQL-based query languages.

The use of built-in predicates in rules has been suggested earlier for SWRL [18]. Some combined OWL 2 RL/SWRL tools with this capability have been implemented [13]. DataTypeRestrictions using XML Schema facets [33] are a kind of unary external checkable predicates.

Let us emphasize that in our second proposal all unary external checkable predicates can be used and we still have Theorem 4.2 given below, where:

- by *OWL 2 eRL* we denote the extension of OWL 2 RL_0 according to the two above mentioned proposals;
- by *OWL 2 eRL*$^+$ we denote the extension of OWL 2 RL^+ by allowing axioms of the form $\mathsf{Refl}(R)$ (i.e. *ReflexiveObjectProperty* axioms), allowing $\top$ as lC, and allowing unary predicates from ECPreds to appear in the places of DR in the BNF grammar rule defining lC.

Clearly, OWL 2 eRL$^+$ is a sublanguage of OWL 2 eRL. The data complexity of OWL 2 eRL and OWL 2 eRL$^+$ is defined as usual.

Theorem 4.2.

1. *The languages OWL 2 eRL and OWL 2 eRL*$^+$ *have* PTime *data complexity.*
2. *Every knowledge base KB in OWL 2 eRL*$^+$ *can be translated to a knowledge base* KB' *in eDatalog which is equivalent to KB in the sense that, for every query* φ *in the language of KB,* $KB \models \varphi$ *iff* $KB' \models \varphi$. □

4.3 Combining OWL 2 eRL and OWL 2 eRL$^+$ with eDatalog

For the combined languages *OWL 2 eRL-eDatalog* and *OWL 2 eRL*$^+$*-eDatalog* studied in the current section we assume that all data role names belong to RRPreds (i.e., are range-restricted).

Definition 4.3. *A* knowledge base *in the combined language OWL 2 eRL-eDatalog (respectively, OWL 2 eRL*$^+$*-eDatalog) is a tuple* $\langle \mathcal{R}, \mathcal{T}, \mathcal{P}, \mathcal{A} \rangle$*, where:*

- $\mathcal{R}$ *is an RBox of OWL 2 eRL (respectively, OWL 2 eRL*$^+$*);*
- $\mathcal{T}$ *is a TBox of OWL 2 eRL (respectively, OWL 2 eRL*$^+$*);*
- $\mathcal{P}$ *is an eDatalog program;*
- $\mathcal{A}$ *is a set consisting of ABox assertions of OWL 2 eRL (respectively, OWL 2 eRL*$^+$*) and ground atoms of ordinary predicates (from* OPreds*).*

The set $\mathcal{A}$ *is called an* ABox *and its elements are called* ABox assertions*.* □

Definition 4.4. *A (*ground conjunctive*) query to a knowledge base of* OWL 2 eRL-eDatalog *is a formula of the form $\varphi_1 \wedge \ldots \wedge \varphi_k$, where each φ_i is either a ground atom of a predicate from* DPreds $\setminus \{\asymp\}$ *or a formula of one of the forms $a \not\approx b$, $\neg A(a)$, $\neg r(a,b)$, $\neg\sigma(a,d)$.*

*A (*ground conjunctive*) query to a knowledge base of* OWL 2 eRL$^+$-eDatalog *is a formula of the form $\varphi_1 \wedge \ldots \wedge \varphi_k$, where each φ_i is a ground atom of a predicate from* DPreds $\setminus \{\asymp\}$. □

Other related notions are defined in the usual way.

We now have the following theorem.

Theorem 4.5.

1. *The combined languages OWL 2 eRL-eDatalog and OWL 2 eRL$^+$-eDatalog have* PTime *data complexity.*
2. *Every knowledge base KB in OWL 2 eRL$^+$-eDatalog can be translated to a knowledge base KB′ in eDatalog which is equivalent to KB in the sense that, for every query φ in the language of KB, KB $\models \varphi$ iff KB′ $\models \varphi$.* □

The following example, considered in [30], involves car insurance discounts.

Example 4.6. Consider the knowledge base in OWL 2 eRL$^+$-eDatalog with:[9]

$$\mathcal{R} = \emptyset$$

$$\begin{aligned}\mathcal{T} = \{&\exists has\_child.\top \sqsubseteq parent,\\ &parent \sqcap male \sqsubseteq father,\\ &parent \sqcap female \sqsubseteq mother\}\end{aligned}$$

$$\begin{aligned}\mathcal{P} = \{&father(x) \wedge has\_child(x,y) \wedge age(y,k) \wedge k \leq 3 \rightarrow discount(x,10),\\ &mother(x) \wedge has\_child(x,y) \wedge age(y,k) \wedge k \leq 3 \rightarrow discount(x,15)\}\end{aligned}$$

$$\begin{aligned}\mathcal{A} = \{&female(Jane), male(Mike), male(Peter),\\ &has\_child(Jane, Peter), has\_child(Mike, Peter), age(Peter, 2)\}.\end{aligned}$$

The query $discount(x,y)$ to this knowledge base has answers $(Jane, 15)$ and $(Mike, 10)$. □

5 Conclusions

In this paper we have identified the maximal fragment OWL 2 RL$^+$ of OWL 2 RL$_0$ with the property that every knowledge base expressed in this fragment is satisfiable. Identifying OWL 2 RL$^+$ is a relatively simple step. More important are our results about OWL 2 RL$^+$ like the one stating that whenever a knowledge base *KB* in OWL 2 RL$_0$ is satisfiable then its corresponding version in OWL 2 RL$^+$ is equivalent to *KB* w.r.t. positive queries. Furthermore, OWL 2 RL$^+$ itself constitutes a base for the development of WORL [7], which combines Datalog$^\neg$ with a variant of OWL 2 RL, using nonmonotonic semantics.

We have also proposed extensions of OWL 2 RL$_0$ and OWL 2 RL$^+$ by allowing *ReflexiveObjectProperty* axioms, external checkable predicates, eDatalog program clauses, and allowing $\top$ as lC. These extensions are very natural and some of the ideas may be known already. Here, we have proved that our extensions OWL 2 eRL and OWL 2 eRL$^+$ have PTime data complexity. They allow efficient computational methods based on the ones of Datalog and are useful for Semantic Web applications.

[9] OWL 2 eRL$^+$-eDatalog is a more general language than EDHL-Datalog [30].

Acknowledgements

This work was supported by the Polish National Science Centre (NCN) under Grants No. 2011/01/B/ST6/02769 and 2011/01/B/ST6/02759 as well as by the Polish National Centre for Research and Development (NCBiR) under Grant No. SP/I/1/77065/10 by the strategic scientific research and experimental development program: “Interdisciplinary System for Interactive Scientific and Scientific-Technical Information”.

References

1. S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison Wesley, 1995.
2. F. Baader, S. Brandt, and C. Lutz. Pushing the EL envelope. In L.P. Kaelbling and A. Saffiotti, editors, *Proceedings of IJCAI'2005*, pages 364–369. Morgan-Kaufmann Publishers, 2005.
3. F. Baader, S. Brandt, and C. Lutz. Pushing the EL envelope further. In *Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*, 2008.
4. S. Brandt. Polynomial time reasoning in a description logic with existential restrictions, GCI axioms, and - what else? In R.L. de Mántaras and L. Saitta, editors, *Proceedings of ECAI'2004*, pages 298–302. IOS Press, 2004.
5. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-Lite family. *J. Autom. Reasoning*, 39(3):385–429, 2007.
6. S.T. Cao, L.A. Nguyen, and A. Szałas. On the Web ontology rule language OWL 2 RL. In P. Jedrzejowicz, N.-T. Nguyen, and K. Hoang, editors, *Proceedings of ICCCI'2011*, volume 6922 of *LNCS*, pages 254–264. Springer, 2011.
7. S.T. Cao, L.A. Nguyen, and A. Szałas. WORL: A Web ontology rule language. In *Proceedings of KSE'2011*, pages 32–39. IEEE, 2011.
8. F.M. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. AL-log: Integrating Datalog and description logics. *J. Intell. Inf. Syst.*, 10(3):227–252, 1998.
9. W. Drabent and J. Maluszynski. Well-founded semantics for hybrid rules. In *Proceedings of RR'2007*, volume 4524 of *LNCS*, pages 1–15. Springer, 2007.
10. B. Dunin-Kęplicz, L.A. Nguyen, and A. Szałas. A layered rule-based architecture for approximate knowledge fusion. *Computer Science and Information Systems*, 7(3):617–642, 2010.
11. B. Dunin-Kęplicz, L.A. Nguyen, and A. Szałas. Tractable approximate knowledge fusion using the Horn fragment of serial propositional dynamic logic. *Int. J. Approx. Reasoning*, 51(3), 2010.
12. T. Eiter, G. Ianni, T. Lukasiewicz, and R. Schindlauer. Well-founded semantics for description logic programs in the Semantic Web. *ACM Trans. Comput. Log.*, 12(2):11, 2011.
13. D. Elenius. SWRL-IQ: A prolog-based query tool for OWL and SWRL. In *Proceedings of OWL ED Workshop 2012*.
14. B. Motik et al. (editors) and I. Horrocks et al. (contributors). OWL 2 Web Ontology Language Direct Semantics. `http://www.w3.org/TR/2009/REC-owl2-direct-semantics-20091027`, 2009.
15. B.N. Grosof, I. Horrocks, R. Volz, and S. Decker. Description logic programs: combining logic programs with description logic. In *Proceedings of WWW'2003*, pages 48–57, 2003.
16. S. Heymans, T. Eiter, and G. Xiao. Tractable reasoning with DL-programs over Datalog-rewritable description logics. In H. Coelho, R. Studer, and M. Wooldridge, editors, *Proceedings of ECAI'2010*, pages 35–40. IOS Press, 2010.
17. I. Horrocks, O. Kutz, and U. Sattler. The even more irresistible SROIQ. In P. Doherty, J. Mylopoulos, and C.A. Welty, editors, *Proceedings of KR'2006*, pages 57–67. AAAI Press, 2006.

18. I. Horrocks, P.F. Patel-Schneider, S. Bechhofer, and D. Tsarkov. OWL rules: A proposal and prototype implementation. *J. Web Sem.*, 3(1):23–40, 2005.
19. U. Hustadt, B. Motik, and U. Sattler. Reasoning in description logics by a reduction to disjunctive Datalog. *J. Autom. Reasoning*, 39(3):351–384, 2007.
20. M. Knorr, J.J. Alferes, and P. Hitzler. A coherent well-founded model for hybrid MKNF knowledge bases. In *Proceedings of ECAI'2008*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 99–103. IOS Press, 2008.
21. A. Krisnadhi and C. Lutz. Data complexity in the EL family of description logics. In N. Dershowitz and A. Voronkov, editors, *Proceedings of LPAR'2007, LNCS 4790*, pages 333–347. Springer, 2007.
22. M. Krötzsch, S. Rudolph, and P. Hitzler. Complexity boundaries for Horn description logics. In *Proceedings of AAAI'2007*, pages 452–457. AAAI Press, 2007.
23. M. Krötzsch, S. Rudolph, and P. Hitzler. Conjunctive queries for a tractable fragment of OWL 1.1. In *Proceedings of ISWC'2007 + ASWC'2007, LNCS 4825*, pages 310–323. Springer, 2007.
24. A.Y. Levy and M.-Ch. Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104(1-2):165–209, 1998.
25. E. Madalińska-Bugaj and L.A. Nguyen. A generalized QSQR evaluation method for Horn knowledge bases. *ACM Trans. Comput. Log.*, 13(4):32, 2012.
26. B. Motik and R. Rosati. Reconciling description logics and rules. *J. ACM*, 57(5), 2010.
27. B. Motik, U. Sattler, and R. Studer. Query answering for OWL-DL with rules. *J. Web Sem.*, 3(1):41–60, 2005.
28. L.A. Nguyen. On the deterministic Horn fragment of test-free PDL. In I. Hodkinson and Y. Venema, editors, *Advances in Modal Logic - Volume 6*, pages 373–392. King's College Publications, 2006.
29. L.A. Nguyen. Constructing finite least Kripke models for positive logic programs in serial regular grammar logics. *Logic Journal of the IGPL*, 16(2):175–193, 2008.
30. L.A. Nguyen. Extending the description Horn logic DHL. In L. Czaja and M. Szczuka, editors, *Proceedings of CS&P'2009*, pages 419–430, 2009.
31. L.A. Nguyen. Horn knowledge bases in regular description logics with PTime data complexity. *Fundamenta Informaticae*, 104(4):349–384, 2010.
32. M. Ortiz, S. Rudolph, and M. Simkus. Query answering in the Horn fragments of the description logics SHOIQ and SROIQ. In T. Walsh, editor, *Proceedings of IJCAI 2011*, pages 1039–1044, 2011.
33. `http://www.w3.org/TR/owl2-syntax/#Datatype_Restrictions`.
34. `http://www.w3.org/TR/owl2-profiles/#OWL_2_RL`. In B. Motik et al. (editors) and D. Calvanese et al. (contributors), "OWL 2 Web Ontology Language Profiles", `http://www.w3.org/TR/owl2-profiles`, 2009.
35. R. Rosati. On conjunctive query answering in $\mathcal{EL}$. In *Proceedings of DL'2007*, pages 451–458.
36. R. Rosati. DL+log: Tight integration of description logics and disjunctive Datalog. In *Proceedings of KR'2006*, pages 68–78. AAAI Press, 2006.
37. H.J. ter Horst. Completeness, decidability and complexity of entailment for RDF Schema and a semantic extension involving the OWL vocabulary. *J. Web Sem.*, 3(2-3):79–115, 2005.

A Proofs

In this appendix we provide proofs for the theorems given earlier in the paper. We first present a translation of OWL 2 RL_0 into Datalog and give some lemmas.

Definition A.1. *By a* negative clause *we understand a formula of the form* $a \not\approx b$, $\neg r(a,b)$, $\neg\sigma(a,d)$ *or* $\varphi_1 \wedge \ldots \wedge \varphi_k \to \bot$, *where* $\varphi_1, \ldots, \varphi_k$ *are atomic formulas.* □

$$
\begin{aligned}
&\pi(\top \sqsubseteq C) &&= \{\pi_{(x)}(C)\}\\
&\pi(\exists\sigma \sqsubseteq C) &&= \{\sigma(x,y) \to \pi_{(x)}(C)\}\\
&\pi(C \sqsubseteq D) &&= \{\pi_{(x)}(C) \to \pi_{(x)}(D)\}\\
&\pi(C \equiv D) &&= \{\pi_{(x)}(C) \to \pi_{(x)}(D),\ \pi_{(x)}(D) \to \pi_{(x)}(C)\}\\
&\pi(DT \equiv DR) &&= \{\pi_{(x)}(DT) \to \pi_{(x)}(DR),\ \pi_{(x)}(DR) \to \pi_{(x)}(DT)\}\\
&\pi(R \equiv S) &&= \{R(x,y) \to S(x,y),\ S(x,y) \to R(x,y)\}\\
&\pi(R \equiv S^-) &&= \{R(x,y) \to S(y,x),\ S(y,x) \to R(x,y)\}\\
&\pi(R_1 \circ \ldots \circ R_k \sqsubseteq S) &&= \{R_1(x_0,x_1) \wedge \ldots \wedge R_k(x_{k-1},x_k) \to S(x_0,x_k)\}\\
&\pi(\sigma \sqsubseteq \varrho) &&= \{\sigma(x,y) \to \varrho(x,y)\}\\
&\pi(\sigma \equiv \varrho) &&= \{\sigma(x,y) \to \varrho(x,y),\ \varrho(x,y) \to \sigma(x,y)\}\\
&\pi(\mathsf{Disj}(C_1,\ldots,C_k)) &&= \{\pi_{(x)}(C_i) \wedge \pi_{(x)}(C_j) \to \bot \mid 1 \le i < j \le k\}\\
&\pi(\mathsf{Disj}(R_1,\ldots,R_k)) &&= \{R_i(x,y) \wedge R_j(x,y) \to \bot \mid 1 \le i < j \le k\}\\
&\pi(\mathsf{Disj}(\sigma_1,\ldots,\sigma_k)) &&= \{\sigma_i(x,y) \wedge \sigma_j(x,y) \to \bot \mid 1 \le i < j \le k\}\\
&\pi(\mathsf{Func}(R)) &&= \{R(x,y) \wedge R(x,z) \to y \approx z\}\\
&\pi(\mathsf{Func}(\sigma)) &&= \{\sigma(x,y) \wedge \sigma(x,z) \to y \asymp z\}\\
&\pi(\mathsf{InvFunc}(R)) &&= \{R(y,x) \wedge R(z,x) \to y \approx z\}\\
&\pi(\mathsf{Refl}(R)) &&= \{R(x,x)\}\\
&\pi(\mathsf{Irref}(R)) &&= \{R(x,x) \to \bot\}\\
&\pi(\mathsf{Sym}(R)) &&= \{R(x,y) \to R(y,x)\}\\
&\pi(\mathsf{Asym}(R)) &&= \{R(x,y) \wedge R(y,x) \to \bot\}\\
&\pi(\mathsf{Trans}(R)) &&= \{R(x,y) \wedge R(y,z) \to R(x,z)\}\\
&\pi(A(a)) &&= \{A(a)\}\\
&\pi(C(a)) &&= \{A(a)\} \cup \pi(A \sqsubseteq C) \text{ when } C \text{ is a complex concept,}\\
& && \quad \text{where } A \text{ is a fresh concept name}\\
&\pi(\varphi) &&= \{\varphi\} \text{ if } \varphi \text{ is an ABox assertion not of the form } C(a)
\end{aligned}
$$

$$
\begin{aligned}
&\pi(\mathsf{Key}(C, R_1, \ldots, R_k, \sigma_1, \ldots, \sigma_h)) =\\
&\quad \{\pi_{(x)}(C) \wedge \pi_{(y)}(C) \wedge\\
&\qquad R_1(x,u_1) \wedge R_1(y,u_1) \wedge \ldots \wedge R_k(x,u_k) \wedge R_k(y,u_k) \wedge\\
&\qquad \sigma_1(x,v_1) \wedge \sigma_1(y,v_1) \wedge \ldots \wedge \sigma_h(x,v_h) \wedge \sigma_h(y,v_h) \to x \approx y\}
\end{aligned}
$$

Fig. 3. A translation π for axioms of OWL 2 RL_0. All variables like x, y, z, u, v are fresh (new) variables. The auxiliary translation $\pi_{(x)}$ is defined in Figure 4. For $\pi(\mathsf{Key}(\ldots))$, note that OWL 2 RL_0 does not "create" new objects and x, y will only be instantiated by named individuals.

Let π be the translation specified in Figure 3. It translates each axiom of OWL 2 RL_0 to a set of formulas, using an auxiliary translation $\pi_{(x)}$, where x denotes a variable. The auxiliary translation is specified in Figure 4. It translates each concept or data range to a formula.

Note that for $\pi_{(x)}(\varphi)$, in the cases when φ is $\exists R.C$, $\exists R.\top$ or $\exists\sigma.DR$:

– φ occurs at the left hand side of $\to$;

$$
\begin{array}{ll}
\pi_{(x)}(DT) & = DT(x) \\
\pi_{(x)}(DT \sqcap DR) & = DT(x) \land \pi_{(x)}(DR) \\
\pi_{(x)}(A) & = A(x) \\
\pi_{(x)}(\{a\}) & = (x \approx a) \\
\pi_{(x)}(\neg C) & = \pi_{(x)}(C) \to \bot \\
\pi_{(x)}(C \sqcap D) & = \pi_{(x)}(C) \land \pi_{(x)}(D) \\
\pi_{(x)}(C \sqcup D) & = \pi_{(x)}(C) \lor \pi_{(x)}(D) \\
\pi_{(x)}(\forall R.C) & = R(x,y) \to \pi_{(y)}(C) \\
\pi_{(x)}(\exists R.C) & = R(x,y) \land \pi_{(y)}(C) \\
\pi_{(x)}(\exists R.\{a\}) & = R(x,a) \\
\pi_{(x)}(\exists R.\top) & = R(x,y) \\
\pi_{(x)}(\top) & = \top \\
\pi_{(x)}(\forall \sigma.DR) & = \sigma(x,y) \to \pi_{(y)}(DR) \\
\pi_{(x)}(\exists \sigma.DR) & = \sigma(x,y) \land \pi_{(y)}(DR) \\
\pi_{(x)}(\exists \sigma.\{d\}) & = \sigma(x,d) \\
\pi_{(x)}(\leq 1\, R.C) & = R(x,y) \land R(x,z) \land \pi_{(y)}(C) \land \pi_{(z)}(C) \to y \approx z \\
\pi_{(x)}(\leq 0\, R.C) & = R(x,y) \land \pi_{(y)}(C) \to \bot \\
\pi_{(x)}(\leq 1\, R.\top) & = R(x,y) \land R(x,z) \to y \approx z \\
\pi_{(x)}(\leq 0\, R.\top) & = R(x,y) \to \bot \\
\pi_{(x)}(\leq 1\, \sigma.DR) & = \sigma(x,y) \land \sigma(x,z) \land \pi_{(y)}(DR) \land \pi_{(z)}(DR) \to y \asymp z \\
\pi_{(x)}(\leq 0\, \sigma.DR) & = \sigma(x,y) \land \pi_{(y)}(DR) \to \bot
\end{array}
$$

Fig. 4. An auxiliary translation $\pi_{(x)}$ used for the translation π defined in Figure 3. All variables y and z are fresh (new) variables.

– the introduced variables are existentially quantified, so these quantifiers change to universal ones when taken out of the scope of $\to$.

Example A.2. For $\varphi = (\exists r.(A_1 \sqcup A_2) \sqsubseteq \forall r.B)$, we have

$$\pi(\varphi) = \{r(x,y) \land (A_1(y) \lor A_2(y)) \to (r(x,z) \to B(z))\}.$$

As for free variables, x, y and z are universally quantified. The only formula ψ of $\pi(\varphi)$ is not a Datalog program clause. The intended translation of φ to a set of Datalog program clauses is

$$\begin{aligned}\pi_3(\varphi) = \pi_2(\psi) = \{&r(x,y) \land A_1(y) \land r(x,z) \to B(z),\\ &r(x,y) \land A_2(y) \land r(x,z) \to B(z)\}.\end{aligned}$$

To specify the translation π_3, we use auxiliary translations $\pi_{2,l}$ and π_2 such that:

– when $\pi_{2,l}$ is applicable to a formula ψ of predicate logic, $\pi_{2,l}(\psi)$ is a set of conjunctions of atomic formulas, and for any interpretation $\mathcal{I}$, $\mathcal{I} \models \bigvee \pi_{2,l}(\psi)$ iff $\mathcal{I} \models \psi$,
– when π_2 is applicable to a formula ψ of predicate logic, $\pi_2(\psi)$ is a set of Datalog program clauses and/or negative clauses, and for any interpretation $\mathcal{I}$, $\mathcal{I} \models \bigwedge \pi_2(\psi)$ iff $\mathcal{I} \models \psi$.

For example, $\pi_{2,l}(r(x,y) \land (A_1(y) \lor A_2(y))) = \{r(x,y) \land A_1(y),\ r(x,y) \land A_2(y)\}$. $\square$

We define:

$$\begin{array}{ll}
\pi_{2,l}(\xi) & = \{\xi\} \text{ if } \xi \text{ is not of any of the forms } \varphi \wedge \psi,\ \varphi \vee \psi,\ r^-(t,t') \\
\pi_{2,l}(r^-(t,t')) & = \{r(t',t)\} \\
\pi_{2,l}(\varphi \vee \psi) & = \begin{cases} \{\top\} \text{ if } \pi_{2,l}(\varphi) = \{\top\} \text{ or } \pi_{2,l}(\psi) = \{\top\} \\ \pi_{2,l}(\varphi) \cup \pi_{2,l}(\psi) \text{ otherwise} \end{cases} \\
\pi_{2,l}(\varphi \wedge \psi) & = \begin{cases} \pi_{2,l}(\psi) \text{ if } \pi_{2,l}(\varphi) = \{\top\} \\ \pi_{2,l}(\varphi) \text{ if } \pi_{2,l}(\psi) = \{\top\} \\ \{\varphi' \wedge \psi' \mid \varphi' \in \pi_{2,l}(\varphi) \text{ and } \psi' \in \pi_{2,l}(\psi)\} \text{ otherwise} \end{cases}
\end{array}$$

$$\begin{array}{ll}
\pi_2(\xi) & = \{\xi\} \text{ if } \xi \text{ is not of any of the forms } \varphi \wedge \psi,\ \varphi \to \psi,\ r^-(t,t') \\
\pi_2(r^-(t,t')) & = \{r(t',t)\} \\
\pi_2(\varphi \to \psi) & = \begin{cases} \pi_2(\psi) \text{ if } \pi_{2,l}(\varphi) = \{\top\} \\ \{\varphi' \wedge \xi' \to \zeta' \mid \varphi' \in \pi_{2,l}(\varphi) \text{ and } (\xi' \to \zeta') \in \pi_2(\psi)\} \cup \\ \quad \{\varphi' \to \psi' \mid \varphi' \in \pi_{2,l}(\varphi) \text{ and } \psi' \in \pi_2(\psi) \text{ and} \\ \qquad \psi' \text{ is not of the form } \xi' \to \zeta'\} \text{ otherwise} \end{cases} \\
\pi_2(\varphi \wedge \psi) & = \pi_2(\varphi) \cup \pi_2(\psi).
\end{array}$$

Given an axiom φ of OWL 2 RL_0, define:

$$\pi_3(\varphi) = \bigcup_{\psi \in \pi(\varphi)} \pi_2(\psi).$$

Given a knowledge base KB in OWL 2 RL_0, define:

$$\pi_3(KB) = \bigcup_{\varphi \in KB} \pi_3(\varphi).$$

Note that, when the ABox of KB is not extensionally reduced, $\pi_3(KB)$ may contain new concept names (not occurring in KB). Recall that queries in the language of KB do not use predicates not occurring in KB.

Lemma A.3. *Let KB be a knowledge base in OWL 2 RL_0. Then:*

1. *$\pi_3(KB)$ contains only Datalog program clauses and negative clauses.*
2. *Every model of $\pi_3(KB)$ is also a model of KB.*
3. *For every query φ in the language of KB, $KB \models \varphi$ iff $\pi_3(KB) \models \varphi$.*

Proof. In the following, let α denote an atomic formula. We define the families of $l\varphi$ and $r\varphi$ by the following BNF grammar:

$$\begin{aligned} l\varphi &:= \alpha \mid r^-(t,t') \mid l\varphi \wedge l\varphi \mid l\varphi \vee l\varphi \\ r\varphi &:= \alpha \mid r^-(t,t') \mid r\varphi \wedge r\varphi \mid l\varphi \to r\varphi \mid l\varphi \to \bot \end{aligned}$$

First, it is straightforward to prove by induction on the structure of C that:

- if C is a concept of the lC family then $\pi_{(x)}(C)$ is a formula φ of the $l\varphi$ family such that applying distribution laws for $\wedge$ and $\vee$ to φ results in $\varphi_1 \vee \ldots \vee \varphi_k$ (where each φ_i does not contain $\vee$) such that the variable x occurs in each φ_i,
- if C is a concept of the rC family then $\pi_{(x)}(C)$ is a formula of the $r\varphi$ family such that if a variable y different from x occurs in the formula then it occurs (among others) at the left hand side of some $\to$ in the formula.

Next, it can be proved by induction on the structure of φ that:

- if φ is a formula of the $l\varphi$ family then $\pi_{2,l}(\varphi)$ is a set of formulas of the $l\varphi$ family without the connective $\vee$ and atoms of the form $r^-(t,t')$;
- if φ is a TBox axiom or an RBox axiom and $\psi \in \pi(\varphi)$ then $\pi_2(\psi)$ contains only formulas of the $r\varphi$ family that are Datalog program clauses or negative clauses.

Therefore, $\pi_3(KB)$ contains only Datalog program clauses and negative clauses.

Let $\mathcal{I}$ be an arbitrary interpretation. It is easy to prove by induction on the structure of ψ that:

- if ψ is a TBox axiom or an RBox axiom then $\mathcal{I} \models \psi$ iff $\mathcal{I} \models \pi(\psi)$,
- if ψ is a formula of predicate logic then:
  - $\mathcal{I} \models \bigvee \pi_{2,l}(\psi)$ iff $\mathcal{I} \models \psi$,
  - $\mathcal{I} \models \bigwedge \pi_2(\psi)$ iff $\mathcal{I} \models \psi$.

Consequently, if ψ is a TBox axiom or an RBox axiom then:

$$\mathcal{I} \models \pi_3(\psi) \;\text{ iff }\; \mathcal{I} \models \pi(\psi), \text{ and iff } \; \mathcal{I} \models \psi. \tag{1}$$

Also observe that:

$$\text{if } \psi \text{ is an ABox assertion and } \mathcal{I} \models \pi_3(\psi) \text{ then } \mathcal{I} \models \psi.$$

Therefore, every model of $\pi_3(KB)$ is also a model of KB.

To consider the third assertion of the lemma assume that φ be a query in the language of KB.

Assume that $KB \models \varphi$ and $\mathcal{I} \models \pi_3(KB)$. We need to show that $\mathcal{I} \models \varphi$. Since $\mathcal{I} \models \pi_3(KB)$, by the second assertion of the lemma, $\mathcal{I} \models KB$, and hence $\mathcal{I} \models \varphi$.

Now assume that $\pi_3(KB) \models \varphi$ and $\mathcal{I} \models KB$. We need to show that $\mathcal{I} \models \varphi$. Let $\mathcal{I}'$ be the interpretation that differs from $\mathcal{I}$ only in that: for every concept name A occurring in $\pi_3(KB)$ but not in KB, which is used to represent a complex concept C as in the translation of $\pi(C(a))$, we have that $A^{\mathcal{I}'} = C^{\mathcal{I}}$. Thus, if $\mathcal{I} \models C(a)$ then $\mathcal{I}' \models A(a)$ and $\mathcal{I}' \models A \sqsubseteq C$. Since $\mathcal{I} \models KB$, by (1), we can derive that $\mathcal{I}' \models \pi_3(KB)$. Since $\pi_3(KB) \models \varphi$, it follows that $\mathcal{I}' \models \varphi$, and hence $\mathcal{I} \models \varphi$. □

Let *EqAxioms* be the set of the following axioms, where p is any k-argument predicate of DPreds different from $\approx$ and $\asymp$, and i is any natural number between 1 and k such that the i-th argument of p is of type *IType*:

$$\begin{gathered} x \approx x \\ x \approx y \to y \approx x \\ x \approx y \land y \approx z \to x \approx z \\ x_i \approx x_i' \land p(x_1, \ldots, x_k) \to p(x_1, \ldots, x_{i-1}, x_i', x_{i+1}, \ldots, x_k). \end{gathered}$$

Since the Unique Names Assumption is adopted for literals (i.e. data constants), to deal with the equality predicate $\asymp$ between literals we use a simpler approach: having a ground atom $d_1 \asymp d_2$, we replace it by $\top$ if d_1 and d_2 are the same literals, and by $\bot$ otherwise, and then simplify the context in which that atom occurs.

Let $\mathcal{P}$ be a Datalog program in the language with $\approx$ but without $\asymp$. Then $\mathcal{P} \cup$ *EqAxioms* is a Datalog program. Let $\mathcal{H}$ be the least Herbrand model of $\mathcal{P} \cup$ *EqAxioms*, computed in the usual way, treating $\approx$ as a normal predicate. Let $\mathcal{I}$ be the interpretation specified as follows:

- $\Delta_o^{\mathcal{I}}$ is the set of all individuals occurring in $\mathcal{H}$,
- $\Delta_d^{\mathcal{I}}$ is the set of all data constants occurring in $\mathcal{H}$,

– for every k-argument predicate $p \in \text{DPreds}$,

$$p^{\mathcal{I}} = \{\langle t_1, \ldots, t_k \rangle \mid p(t_1, \ldots, t_k) \in \mathcal{H}\}.$$

Observe that $\approx^{\mathcal{I}}$ is a congruence of $\mathcal{I}$. Clearly, the quotient $\mathcal{I}/_{\approx}$ of $\mathcal{I}$ by the congruence $\approx^{\mathcal{I}}$ is a model of $\mathcal{P}$. We call it the *standard model* of $\mathcal{P}$.

We now prove the theorems given earlier in the paper. To increase readability we remind each of the theorems before presenting its proof.

Theorem 3.5.

1. *OWL 2 RL$^+$ is a maximal fragment (w.r.t. allowed features) of OWL 2 RL$_0$ such that every knowledge base expressed in the fragment is satisfiable.*
2. *Every knowledge base KB in OWL 2 RL$^+$ can be translated to a Datalog program $\mathcal{P}$ which is equivalent to KB in the sense that, for every query φ in the language of KB, $KB \models \varphi$ iff $\mathcal{P} \models \varphi$.*

Proof. Let KB be a knowledge base in OWL 2 RL$^+$. Observe that $\mathcal{P} = \pi_3(KB)$ is a Datalog program without $\asymp$.[10] By Lemma A.3(2), the standard model of the Datalog program $\pi_3(KB)$ is also a model of KB. Hence KB is satisfiable. The first assertion of the theorem follows from this fact and Example 3.1. The second assertion of the theorem follows from Lemma A.3(3). □

Theorem 3.7. *Let KB be a knowledge base in OWL 2 RL$_0$, KB' be the normal form of KB, and KB'' be the corresponding version of KB in OWL 2 RL$^+$. Then:*

1. *KB' is equivalent to KB in the sense that, for every query φ in the language of KB, $KB \models \varphi$ iff $KB' \models \varphi$;*
2. *if KB is satisfiable and φ is a positive query in the language of KB then $KB \models \varphi$ iff $KB'' \models \varphi$.*

Proof. Consider the first assertion. Let φ be a query in the language of KB.

Assume that $KB \models \varphi$ and let $\mathcal{I}$ be a model of KB'. We show that $\mathcal{I} \models \varphi$. Recall that a replacement of $\neg lC$ by A for KB' occurs only in positions for rC (i.e., in the right hand side of $\sqsubseteq$ and not in the scope of $\neg$). If $A \sqcap lC \sqsubseteq \bot$ is an axiom of KB' then $\mathcal{I}$ validates also the axiom $A \sqsubseteq \neg lC$. Since $\mathcal{I}$ is a model of KB', it follows that $\mathcal{I}$ is also a model of KB, and hence $\mathcal{I} \models \varphi$.

Now assume that $KB' \models \varphi$ and let $\mathcal{I}$ be a model of KB. We show that $\mathcal{I} \models \varphi$. Let $\mathcal{I}'$ be the interpretation that extends $\mathcal{I}$ by interpreting each concept name A occurring in an axiom $A \sqcap lC \sqsubseteq \bot$ of KB' by $A^{\mathcal{I}'} = (\neg lC)^{\mathcal{I}}$. (Note that, for each concept name A occurring in KB' but not occurring in KB, KB' contains exactly one axiom of the form $A \sqcap lC \sqsubseteq \bot$.) Clearly, $\mathcal{I}'$ is a model of KB', and hence $\mathcal{I}' \models \varphi$. It follows that $\mathcal{I} \models \varphi$.

Consider the second assertion and assume that KB is satisfiable and φ is a positive query in the language of KB. It suffices to show that $KB' \models \varphi$ iff $KB'' \models \varphi$. Clearly, $KB'' \models \varphi$ implies $KB' \models \varphi$. Since φ is a positive query and $KB' \setminus KB''$ consists only of axioms which are translated to negative clauses or clauses of the form $(\psi \to y \asymp z)$ (whose ground instances are either trivially true or equivalent to negative clauses), we can also conclude that $KB' \models \varphi$ implies $KB'' \models \varphi$, which completes the proof. □

To prove Theorems 4.2 and 4.5 we need the following definition and lemma.

[10] One can apply also the translation specified in [34] to KB to get a Datalog program, which uses RDF triples as atoms and uses also constants like rdf:type, rdfs:subClassOf, owl:hasValue. The program obtained in this way is "equivalent" to KB in a certain sense.

Function ground-atomic-consequences(KB)

Input: a knowledge base with constraints KB in eDatalog.
Output: the set of ground atomic consequences of KB.

1 $\mathcal{I} := \{\varphi \in KB \mid \varphi$ is of the form $\bot$ or ψ or $\neg\psi$ where ψ is a ground atom$\}$;
2 **foreach** *formula (i.e., program clause or constraint clause or assertion)* ρ *of* $KB \cup EqAxioms$ **do**
3   reorder the body of ρ so that $\rho = (\varphi_1 \wedge \ldots \wedge \varphi_k \wedge \psi_1 \wedge \ldots \wedge \psi_h \to \xi)$, where:
    – $k \geq 0$, $h \geq 0$,
    – $\varphi_1, \ldots, \varphi_k$ are atoms of predicates from DPreds,
    – $\psi_1, \ldots, \psi_k$ are atoms of predicates from ECPreds;
4   **foreach** *instance* $\rho' = (\varphi'_1 \wedge \ldots \wedge \varphi'_k \wedge \psi'_1 \wedge \ldots \wedge \psi'_h \to \xi')$ *of* ρ *such that for every* $1 \leq i \leq k$, $\varphi'_i \in \mathcal{I}$ *or* φ'_i *is of the form* $d \asymp d$ **do**
5   **if** $\psi'_1, \ldots, \psi'_h$ *are all true (note that these atoms are ground)* **then**
6   **if** ξ' *is* $\bot$ *or a ground atom* **then** add ξ' to $\mathcal{I}$
7   **else** `// the predicate of` ξ' `belongs to` DPreds
8   **foreach** *well-typed ground instance* ξ'' *of* ξ' *that uses individuals and literals (i.e. data constants) only from* KB **do**
9   add ξ'' to $\mathcal{I}$
10 **if** $\mathcal{I}$ *changed during the last execution of Step 2* **then goto** Step 2;
11 **return** $\mathcal{I}$

Definition A.4. *An* eDatalog constraint clause *is a formula of the form*

$$\varphi_1 \wedge \ldots \wedge \varphi_n \to \psi,$$

where:

– $n \geq 0$ *and* $\varphi_1, \ldots, \varphi_n$ *are atomic formulas,*
– ψ *is either* $\bot$ *or an atom of a predicate from* ECPreds*,*
– *every variable occurring in* ψ *occurs also in some* φ_i *whose predicate belongs to* RRPreds*,*
– *every variable occurring in some* φ_i *whose predicate belongs to* ECPreds *occurs also in some atom* φ_j *whose predicate belongs to* RRPreds*.*

A knowledge base with constraints in eDatalog *is a pair* $\langle \mathcal{P}, \mathcal{A} \rangle$*, where* $\mathcal{P}$ *is a finite set consisting of eDatalog program clauses and constraint clauses, and* $\mathcal{A}$*, called the* ABox of the knowledge base*, is a finite set of formulas of the form* φ *or* $\neg\varphi$*, where* φ *is a ground atom of a predicate from* DPreds*. We sometimes treat the knowledge base as the set* $\mathcal{P} \cup \mathcal{A}$*.* □

Given a knowledge base with constraints KB in eDatalog, the *set of ground atomic consequences* of KB is specified by function `ground-atomic-consequences`(KB) given on page 20.

The following lemma can easily be proved.

Lemma A.5. *Let* KB *be a knowledge base with constraints in eDatalog and let* $\mathcal{I} =$ `ground-atomic-consequences`(KB)*. Then:*

1. KB *is unsatisfiable iff* $\mathcal{I}$ *contains* $\bot$ *or a pair* φ *and* $\neg\varphi$ *or an atom* $d_1 \asymp d_2$*, where* d_1 *and* d_2 *are different literals, or a ground atom of a predicate from* ECPreds *whose value is false.*

2. *If KB is satisfiable and* $\varphi = (\varphi_1 \wedge \ldots \wedge \varphi_k)$ *is a query of OWL 2 eRL-eDatalog then* $KB \models \varphi$ *iff, for every* $1 \leq i \leq k$, $\varphi_i \in \mathcal{I}$ *or* φ_i *is of the form* $d \asymp d$.
3. *The set* $\mathcal{I}$ *can be computed in deterministic polynomial time in the size of the ABox of KB.* □

Let p be a unary predicate from ECPreds. Define $\pi_{(x)}(p) = p(x)$. This leads to the corresponding extensions of translations π, π_2 and π_3 for OWL 2 eRL and OWL 2 eRL$^+$.

Theorem 4.2.

1. *The languages OWL 2 eRL and OWL 2 eRL$^+$ have* PTIME *data complexity.*
2. *Every knowledge base KB in OWL 2 eRL$^+$ can be translated to a knowledge base* KB' *in eDatalog which is equivalent to KB in the sense that, for every query* φ *in the language of KB,* $KB \models \varphi$ *iff* $KB' \models \varphi$.

Proof. Let KB be a knowledge base in OWL 2 eRL and let $KB' = \pi_3(KB)$. Observe that KB' is a knowledge base with constraints in eDatalog, and if KB is a knowledge base in OWL 2 eRL$^+$ then KB' is a knowledge base in eDatalog. As for Lemma A.3, it can be seen that, for every query φ in the language of KB, $KB \models \varphi$ iff $KB' \models \varphi$. By Lemma A.5, it follows that both OWL 2 eRL and OWL 2 eRL$^+$ have PTIME data complexity. □

Theorem 4.5 can be proved analogously.