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, a knowledge base in OWL 2 RL may be unsatisfiable. The reason is that, when translated into Datalog, the result may consist of a Datalog program and a set of constraints in the form of negative clauses. In this paper we first identify a maximal fragment of OWL 2 RL called OWL 2 RL$^+$ with the property that every knowledge base expressed in this fragment 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 from researchers in the last decade. 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 the W3C consortium in 2009, is based on the description logic $\mathcal{SROIQ}$ [14]. This logic is highly expressive but has intractable combined complexity (N2ExpTime-complete) and data complexity (NP-hard) for basic reasoning problems. Thus, W3C recommended also 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 $\mathcal{EL}$ [1, 2], DL-Lite [4] and DLP (Description Logic Programs) [12], respectively.

OWL 2 RL [23] restricts the full language OWL 2 to allow a translation into Datalog and hence to obtain PTime data complexity and efficient computational methods. However, a knowledge base in OWL 2 RL may be unsatisfiable. Namely, translating such a knowledge base into Datalog may result in a Datalog program plus a set of constraints expressed as negative clauses. In database applications, constraints are important for keeping the database consistent w.r.t. its specification. They are used to block disallowed data modifications. In Semantic Web applications, the situation is a bit different. An ontology may consist of many component ontologies which were developed independently by different people. It may also include component ontologies which will be updated independently by different people. Thus, constraints are less important for ontologies, and when possible, should be replaced by syntactic restrictions. This is the case of OWL 2 RL.

In this paper we first identify a maximal fragment of OWL 2 RL called OWL 2 RL$^+$ with the property that every knowledge base expressed in this fragment can be translated to a Datalog program and hence is satisfiable. We then propose an extension OWL 2 eRL of OWL 2 RL with PTime data complexity, and an extension OWL 2 eRL$^+$ of OWL 2 RL$^+$ that can be translated into eDatalog (an extended version of Datalog). Next, we extend both OWL 2 eRL and OWL 2 eRL$^+$ with eDatalog. Combining OWL 2 eRL or

OWL 2 eRL$^+$ with eDatalog gives us the freedom to use the syntax of both the languages and allows us to represent knowledge about the domain not only in terms of concepts and roles but also by using predicates with higher arities.

1.1 Related Work

This work is a revised and extended version of the conference paper [5].

As stated in [23], OWL 2 RL has been inspired by Description Logic Programs (DLP) [12] and pD^* [32]. The logical base of DLP is the description Horn logic DHL [12]. Some extensions of DHL were studied by Nguyen in [28].

A number of Horn fragments of DLs with PTime data complexity have been studied in [12, 16, 4, 18, 20, 30, 29]. The combined complexities of Horn fragments of DLs were studied, amongst others, in [19]. Some Horn fragments of DLs without ABoxes that have PTime complexity have also been studied in [3, 1]. See [29, Section 4] for an overview of these works.

Combinations of rule languages with description logics have been studied in a considerable number of works: [7] (on $\mathcal{AL}$*-log*), [21] (on *CARIN*), [25] (on *DL-safe rules*), [31] (on $\mathcal{DL}$*+log*), [24, 17] (on *hybrid MKNF*), [8] (on *hybrid programs*), [11] (on *dl-programs*), [6] (on WORL). Only [6] directly involves with OWL 2 RL. In that work we studied a combination of a variant of OWL 2 RL with eDatalog$^\neg$.

Some other related works are [15] (on SWRL), [13] (on description logic programs with negation), [9] (on layered rule-based architecture) and [26, 27, 10] (on Horn fragments of modal logics).

1.2 The Structure of This Paper

The rest of this paper is structured as follows. In Section 2 we specify OWL 2 RL [23] as a logical formalism. Section 3 is devoted to OWL 2 RL$^+$. Section 4 presents extensions of OWL 2 RL and OWL 2 RL$^+$. Section 5 concludes this work. Proofs of the results of this paper are presented in the appendix.

2 The Logical Formalism of OWL 2 RL

In this section we specify OWL 2 RL as a logical formalism, using the syntax of description logics. We strictly follow the specification of OWL 2 RL given in [23].

We denote the set of *concept names* by CNames, and the set of *role names* by RNames. We use the truth symbol $\top$ to denote *owl:Thing* [23], and use:

- a and b to denote *individuals* (i.e. *objects*)
- d to denote a *literal* [23] (i.e. a data constant)
- A and B to denote concept names (i.e. *Class* elements [23])
- C and D to denote *concepts* (i.e. *ClassExpression* elements [23])
- lC to denote a concept standing for a *subClassExpression* of [23]
- rC to denote a concept standing for a *superClassExpression* of [23]
- eC to denote a concept standing for an *equivClassExpression* of [23]
- DT to denote a *data type* (i.e. a *Datatype* of [23])
- DR to denote a *data range* (i.e. a *DataRange* of [23])
- r and s to denote *object role names* (i.e. *ObjectProperty* elements [23])
- R and S to denote *object roles* (i.e. *ObjectPropertyExpression* elements [23])
- σ and ϱ to denote *data role names* (i.e. *DataProperty* elements [23]).

The families of R, DR, lC, rC, eC are defined by the following BNF grammar:

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

The class constructor *ObjectOneOf* [23] can be written as $\{a_1, \ldots, a_k\}$ and expressed as $\{a_1\} \sqcup \ldots \sqcup \{a_k\}$.

We will use the following abbreviations: Disj (Disjoint), Func (Functional), InvFunc (InverseFunctional), Refl (Reflexive), Irref (Irreflexive), Sym (Symmetric), Asym (Asymmetric), Trans (Transitive), Key (HasKey).

A *TBox axiom*, which stands for a *ClassAxiom* or a *DatatypeDefinition* or a *HasKey* axiom [23], is an expression of one of the following forms:

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

An *RBox axiom*, which stands for an *ObjectPropertyAxiom* or a *DataPropertyAxiom* [23], is an expression of one of the following forms:

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

Note that axioms of the form $R = S$, $R = S^-$, $\mathsf{Sym}(R)$ or $\mathsf{Trans}(R)$ are expressible by axioms of the form $R_1 \circ \ldots \circ R_k \sqsubseteq S$, and hence can be deleted from the above list. An RBox axiom of the form $\exists R.\top \sqsubseteq rC$ (resp. $\top \sqsubseteq \forall R.rC$, $\exists \sigma \sqsubseteq rC$, $\top \sqsubseteq \forall \sigma.DR$) stands for an *ObjectPropertyDomain* (resp. *ObjectPropertyRange*, *DataPropertyDomain*, *DataPropertyRange*) axiom [23]. One can classify these latter axioms as TBox axioms instead of RBox axioms. Similarly, $\mathsf{Key}(\ldots)$ axioms can be classified as RBox axioms instead.

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

$$a = b,\ \ a \neq b,\ \ rC(a),\ \ DT(d),\ \ r(a,b),\ \ \neg r(a,b),\ \ \sigma(a,d),\ \ \neg\sigma(a,d).$$

In OWL 2 RL [23], assertions of the form $DT(d)$ are implicitly provided by declarations of DT and d. The other ABox assertions stand for *Assertion* elements of [23]. We also call an ABox assertion as an *ABox axiom*.

In OWL 2 RL [23], there are also axioms standing for declarations and annotation axioms used for keeping meta information about the ontology. These kinds of axioms are inessential from the logical point of view and are omitted here.

An *RBox* (resp. *TBox*, *ABox*) is a finite set of RBox (resp. TBox, ABox) axioms. An ABox $\mathcal{A}$ is said to be *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 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 $\mathcal{A}$. We may present a knowledge base as a set of axioms.

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 with $\Delta_o^{\mathcal{I}}$ called the *data domain* of $\mathcal{I}$, and a function $\cdot^{\mathcal{I}}$ called the *interpretation function* of $\mathcal{I}$, which maps

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

Fig. 1. Interpretation of data ranges, inverse object roles, and complex concepts. In this figure, φ is of the form DR or $\{d\}$, and $\#\Gamma$ denotes the cardinality of the set Γ.

- 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 $\sigma^{\mathcal{I}} \subseteq \Delta_o^{\mathcal{I}} \times \Delta_d^{\mathcal{I}}$.

As OWL 2 RL has no explicit features for declaring whether two literals are equal or not, we adopt the unique name assumption for literals, which means that if $d_1 \neq d_2$ then $d_1^{\mathcal{I}} \neq d_2^{\mathcal{I}}$. The interpretation function is extended to interpret data ranges, inverse object roles and complex concepts as shown in Figure 1.

From now on, if not stated otherwise, by an *axiom* we mean an RBox axiom, a TBox axiom or an ABox axiom. 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 $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$
- $\mathcal{I} \models \sigma(a,d)$ iff $(a^{\mathcal{I}}, d^{\mathcal{I}}) \in \sigma^{\mathcal{I}}$
- $\mathcal{I} \models \varphi = \psi$ iff $\varphi^{\mathcal{I}} = \psi^{\mathcal{I}}$,
  where φ and ψ may be of the form C, R, R^{-}, DT, DR or a
- $\mathcal{I} \models a \neq b$ 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. $\forall x,y,z(R^{\mathcal{I}}(x,y) \wedge R^{\mathcal{I}}(x,z) \rightarrow y = z)$)
- $\mathcal{I} \models \mathsf{InvFunc}(R)$ iff $R^{\mathcal{I}}$ is inverse-functional
  (i.e. $\forall x,y,z(R^{\mathcal{I}}(x,z) \wedge R^{\mathcal{I}}(y,z) \rightarrow x = y)$)
- $\mathcal{I} \models \mathsf{Irref}(R)$ 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 $x, y \in C^{\mathcal{I}}$, $z_1, \ldots, z_k \in \Delta_o^{\mathcal{I}}$ and $d_1, \ldots, d_h \in \Delta_d^{\mathcal{I}}$, if $\{(x, z_i), (y, z_i)\} \subseteq R_i^{\mathcal{I}}$ and $\{(x, d_j), (y, d_j)\} \subseteq \sigma_i^{\mathcal{I}}$ for all $1 \leq i \leq k$ and $1 \leq j \leq h$, then $x = y$.

The semantics of $\mathsf{Key}(C, R_1, \ldots, R_k, \sigma_1, \ldots, \sigma_h)$ is defined according to the semantics of the *HasKey* constructor of [23], but note that it has a clear meaning only when all the roles $R_1, \ldots, R_k$, $\sigma_1, \ldots, \sigma_h$ are assumed to be functional.

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 $\mathcal{A}$.

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

$$a = b, \;\; a \neq b, \;\; A(a), \;\; \neg A(a), \;\; r(a, b), \;\; \neg r(a, b), \;\; \sigma(a, d), \;\; \neg \sigma(a, d).$$

An interpretation $\mathcal{I}$ satisfies a query $\varphi = \varphi_1 \wedge \ldots \wedge \varphi_k$, 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 (with the "range-restrictedness condition"). 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.

The *data complexity* of OWL 2 RL (for the conjunctive query answering problem) is the complexity of checking where 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 $\mathcal{A}$ when 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 unsatisfiable knowledge bases in OWL 2 RL. We then present the restricted version OWL 2 RL$^+$ of OWL 2 RL that overcomes this problem and give some important properties of OWL 2 RL$^+$.

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

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

$$KB_{13} = \{a \neq b,\ a = b\}$$
$$KB_{14} = \{a \neq b,\ A \sqsubseteq \leq 1\, r.B,\ A(c),\ r(c,a),\ B(a),\ r(c,b),\ B(b)\}$$
$$KB_{15} = \{\neg r(a,b),\ r(a,b)\}$$
$$KB_{16} = \{\neg r(a,b),\ s \sqsubseteq r,\ s(a,b)\}$$
$$KB_{17} = \{\neg \sigma(a,d),\ \sigma(a,d)\}$$

Assuming that assertions of the forms $A(a)$, $r(a,b)$, $\sigma(a,d)$, $DT(d)$, $a = b$ are basic and should always be allowed, and that atomic concepts should be allowed in the left hand side of $\sqsubseteq$ in TBox axioms, then it is clear that the above knowledge bases are unsatisfiable due to the occurrences of concepts of the forms $\neg B$, $\leq 0\, r.B$, $\leq 0\, r.\top$, $\leq n\, \sigma.DT$ in the right hand side of $\sqsubseteq$ in TBox axioms and due to axioms of the forms $\mathsf{Disj}(\ldots)$, $\mathsf{Irref}(r)$, $\mathsf{Asym}(r)$, $a \neq b$, $\neg r(a,b)$, and $\neg \sigma(a,d)$. ◁

We define OWL 2 RL$^+$ to be the restriction of OWL 2 RL such that:

- the constructors $\neg lC$, $\leq 0\, R.lC$, $\leq 0\, R.\top$ and $\leq n\, \sigma.DR$ (where n is 0 or 1) are disallowed in the BNF grammar rule defining the rC family
- axioms of the forms $\mathsf{Disj}(\ldots)$, $\mathsf{Irref}(R)$, $\mathsf{Asym}(R)$, $a \neq b$, $\neg r(a,b)$, $\neg \sigma(a,d)$ are disallowed.

These restrictions correspond to the following ones for the OWL 2 RL of [23]:

- the constructors *superComplementOf*, *superObjectMaxCardinality* with limit 0, and *superDataMaxCardinality* are disallowed in the definition of *superClassProperty*
- axioms of the following kinds are disallowed
  - *DisjointClasses*, *DisjointObjectProperties*, *DisjointDataProperties*,
  - *IrreflexiveObjectProperty*, *AsymmetricObjectProperty*,
  - *DifferentIndividuals*,
  - *NegativeObjectPropertyAssertion*, *NegativeDataPropertyAssertion*.

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 $\varphi_1 \wedge \ldots \wedge \varphi_k$, where each φ_i is of one of the forms $a = b$, $A(a)$, $r(a,b)$, $\sigma(a,d)$.

We now recall definitions of Datalog:

- A *term* is either a *constant* or a *variable*.
- If p is an n-array 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 $\varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow \psi$, where $n \geq 0$ and $\varphi_1, \ldots, \varphi_n, \psi$ are atoms. The conjunction $\varphi_1 \wedge \ldots \wedge \varphi_n$ is called the *body* and ψ is called the *head* of the clause. The program clause is required to satisfy the *range-restrictedness* condition, which states that every variable occurring in the clause's head must occur also in the clause's body.
- A *Datalog program* is a finite set of Datalog program clauses.

Theorem 3.2.

1. *OWL 2 RL$^+$ is a maximal fragment (w.r.t. allowed features) of OWL 2 RL 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 $\mathcal{P} \models \varphi$.* ◁

Let $\bot$ stand for the falsity symbol, with the semantics that $\bot^{\mathcal{I}} = \emptyset \subset \Delta_o^{\mathcal{I}}$.

Let KB be a knowledge base in OWL 2 RL. 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 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 is 0 or 1) and deleting all axioms of the forms $A \sqcap lC \sqsubseteq \bot$, $\mathsf{Disj}(\ldots)$, $\mathsf{Irref}(R)$, $\mathsf{Asym}(R)$, $a \neq b$, $\neg R(a, b)$, $\neg\sigma(a, d)$.

Theorem 3.3. *Let KB be a knowledge base in OWL 2 RL, 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 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 can be replaced by OWL 2 RL$^+$ without any further loss of generality.

4 Extensions of OWL 2 RL 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 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* [23] (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 $P(IType \times IType)$, and a data role name is a binary predicate of type $P(IType \times LType)$. Extending OWL 2 RL with Datalog, apart from concept names and role names we will use also a set OPreds of *ordinary predicates* (including data types) and a set ECPreds of *external checkable predicates*. We assume that the sets CNames, RNames, OPreds and ECPreds are finite and pairwise disjoint. Let DPreds = CNames $\cup$ RNames $\cup$ OPreds, which stands for the set of *defined predicates*. A k-ary predicate from OPreds has type $P(T_1 \times \ldots \times T_k)$, where each T_i is either *IType* or *LType*. A k-ary 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-ary 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. 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 $=$ and $\doteq$ (both belonging to OPreds), where $=$ has the type $P(IType \times IType)$ and $\doteq$ has the type $P(LType \times LType)$. These equality predicates have the usual semantics, with the unique name assumption for literals (i.e. data constants).

Extending Datalog to eDatalog, we want to drop the range-restrictedness condition. However, to allow external checkable predicates we cannot do so 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:

- 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 $\varphi_1 \wedge \ldots \wedge \varphi_n \rightarrow \psi$, 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.

Related 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 name assumption for literals (i.e. data constants).

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. The reason is probably that translating $\mathsf{Refl}(R)$ into Datalog we get a program clause $\forall x\, R(x, x)$ which violates the range-restrictedness condition. Similarly, $\top$ is disallowed as lC in OWL 2 RL. However, these restrictions are unnecessary. The Horn fragment without function symbols of predicate logic also has PTime data complexity. Furthermore, as shown in [22], evaluation methods of Datalog can be extended to Horn knowledge bases without function symbols in predicate logic. Thus, our first proposal is to extend OWL 2 RL with the feature of *ReflexiveObjectProperty* axioms and allowing $\top$ as lC.

Our second proposal for extending OWL 2 RL is to allow unary predicates from ECPreds to appear in the places of *DataRange* elements. For example, 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 the other comparison operators, the *between* operator, the operator used for checking pattern of a string. RDF or XML syntax for these operators and for constructing complex checkable predicates should be standardized.

By *OWL 2 eRL* we denote the extension of OWL 2 RL 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 or OWL 2 eRL$^+$ is defined as usual.

Theorem 4.1.

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 here we assume that all data role names belong to RRPreds (i.e. are range-restricted). A *knowledge base* in the combined language OWL 2 eRL-eDatalog (resp. 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 (resp. OWL 2 eRL$^+$), $\mathcal{T}$ is a TBox of OWL 2 eRL (resp. OWL 2 eRL$^+$), $\mathcal{P}$ is an eDatalog program, and $\mathcal{A}$ is a set consisting of ABox assertions of OWL 2 eRL (resp. 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*.

A (*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 $\mathrm{DPreds} \setminus \{\doteq\}$ or a formula of one of the forms $a \neq b$, $\neg A(a)$, $\neg r(a, b)$, $\neg \sigma(a, d)$.

A (*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 $\mathrm{DPreds} \setminus \{\doteq\}$.

Other related notions are defined in the usual way.

Theorem 4.2.

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 involves car insurance discounts. It comes from [28].

Example 4.3. Consider the knowledge base in OWL 2 eRL$^+$-eDatalog with:[4]

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

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

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

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

The query $\mathit{discount}(x, y)$ to this knowledge base has answers $(\mathit{Jane}, 15)$ and $(\mathit{Mike}, 10)$. (The discounts are in percentage.) ◁

[4] OWL 2 eRL$^+$-eDatalog is a more general language than EDHL-Datalog [28].

5 Conclusions

We have identified the maximal fragment OWL 2 RL$^+$ of OWL 2 RL with the property that every knowledge base expressed in this fragment is satisfiable. We have also proposed extensions of OWL 2 RL and OWL 2 RL$^+$ with PTime data complexity by allowing *ReflexiveObjectProperty* axioms, external checkable predicates, eDatalog program clauses, and allowing $\top$ as lC. These extensions are novel and very natural. They allow efficient computational methods (based on the ones of Datalog) and are very useful for practical applications of Semantic Web.

Acknowledgements. This work was supported by the Polish National Science Centre (NCN) under Grant No. 2011/01/B/ST6/02769 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. 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.
2. 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.
3. 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.
4. 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.
5. 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.
6. 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.
7. 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.
8. 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.
9. 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.
10. 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.
11. 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.
12. 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.
13. 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.
14. 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.
15. 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.
16. 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.
17. 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.
18. 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.

19. M. Krötzsch, S. Rudolph, and P. Hitzler. Complexity boundaries for Horn description logics. In *Proceedings of AAAI'2007*, pages 452–457. AAAI Press, 2007.
20. 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.
21. A.Y. Levy and M.-Ch. Rousset. Combining Horn rules and description logics in CARIN. *Artificial Intelligence*, 104(1-2):165–209, 1998.
22. E. Madalińska-Bugaj and L.A. Nguyen. A generalized QSQR evaluation method for Horn knowledge bases. To appear in ACM Transactions on Computational Logic (available at `http://tocl.acm.org/accepted/455nguyen.pdf`), 2012.
23. B. Motik, B.C. Grau, I. Horrocks, Z. Wu, A. Fokoue, C. Lutz, D. Calvanese, J. Carroll, G. De Giacomo, J. Hendler, I. Herman, B. Parsia, P.F. Patel-Schneider, A. Ruttenberg, U. Sattler, and M. Schneider. OWL 2 RL. `http://www.w3.org/TR/owl2-profiles/#OWL_2_RL`, 2009.
24. B. Motik and R. Rosati. Reconciling description logics and rules. *J. ACM*, 57(5), 2010.
25. B. Motik, U. Sattler, and R. Studer. Query answering for OWL-DL with rules. *J. Web Sem.*, 3(1):41–60, 2005.
26. 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.
27. 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.
28. 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.
29. L.A. Nguyen. Horn knowledge bases in regular description logics with PTime data complexity. *Fundamenta Informaticae*, 104(4):349–384, 2010.
30. R. Rosati. On conjunctive query answering in $\mathcal{EL}$. In *Proceedings of DL'2007*, pages 451–458.
31. R. Rosati. DL+log: Tight integration of description logics and disjunctive Datalog. In *Proceedings of KR'2006*, pages 68–78. AAAI Press, 2006.
32. 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 into Datalog and give some lemmas.

Let π be the translation specified in Figure 2 (on page 12), which translates each axiom of OWL 2 RL to a set of formulas. It uses an auxiliary translation $\pi_{(x)}$ that translates each concept or data range to a formula, where x denotes a variable. For $\pi_{(x)}(\varphi)$ in the cases when φ is $\exists R.C$, $\exists R.\top$ or $\exists \sigma.DR$ note that φ occurs in the left hand side of $\rightarrow$ and the introduced variables are existentially quantified. Those quantifiers change to universal when taken out of the scope of $\rightarrow$.

We define

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

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

$$
\begin{array}{ll}
\pi_{(x)}(DT) & = DT(x) \\
\pi_{(x)}(DT \sqcap DR) & = DT(x) \wedge \pi_{(x)}(DR) \\
\pi_{(x)}(A) & = A(x) \\
\pi_{(x)}(\{a\}) & = (x = a) \\
\pi_{(x)}(\neg C) & = \pi_{(x)}(C) \rightarrow \bot \\
\pi_{(x)}(C \sqcap D) & = \pi_{(x)}(C) \wedge \pi_{(x)}(D) \\
\pi_{(x)}(C \sqcup D) & = \pi_{(x)}(C) \vee \pi_{(x)}(D) \\
\pi_{(x)}(\forall R.C) & = R(x,y) \rightarrow \pi_{(y)}(C) \\
\pi_{(x)}(\exists R.C) & = R(x,y) \wedge \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) \rightarrow \pi_{(y)}(DR) \\
\pi_{(x)}(\exists \sigma.DR) & = \sigma(x,y) \wedge \pi_{(y)}(DR) \\
\pi_{(x)}(\exists \sigma.\{d\}) & = \sigma(x,d) \\
\pi_{(x)}(\leq 1\, R.C) & = R(x,y) \wedge R(x,z) \wedge \pi_{(y)}(C) \wedge \pi_{(z)}(C) \rightarrow y = z \\
\pi_{(x)}(\leq 0\, R.C) & = R(x,y) \wedge \pi_{(y)}(C) \rightarrow \bot \\
\pi_{(x)}(\leq 1\, R.\top) & = R(x,y) \wedge R(x,z) \rightarrow y = z \\
\pi_{(x)}(\leq 0\, R.\top) & = R(x,y) \rightarrow \bot \\
\pi_{(x)}(\leq 1\, \sigma.DR) & = \sigma(x,y) \wedge \sigma(x,z) \wedge \pi_{(y)}(DR) \wedge \pi_{(z)}(DR) \rightarrow y \doteq z \\
\pi_{(x)}(\leq 0\, \sigma.DR) & = \sigma(x,y) \wedge \pi_{(y)}(DR) \rightarrow \bot \\
\pi(\top \sqsubseteq C) & = \{\pi_{(x)}(C)\} \\
\pi(\exists \sigma \sqsubseteq C) & = \{\sigma(x,y) \rightarrow \pi_{(x)}(C)\} \\
\pi(C \sqsubseteq D) & = \{\pi_{(x)}(C) \rightarrow \pi_{(x)}(D)\} \\
\pi(C = D) & = \{\pi_{(x)}(C) \rightarrow \pi_{(x)}(D),\ \pi_{(x)}(D) \rightarrow \pi_{(x)}(C)\} \\
\pi(DT = DR) & = \{\pi_{(x)}(DT) \rightarrow \pi_{(x)}(DR),\ \pi_{(x)}(DR) \rightarrow \pi_{(x)}(DT)\} \\
\pi(R = S) & = \{R(x,y) \rightarrow S(x,y),\ S(x,y) \rightarrow R(x,y)\} \\
\pi(R = S^{-}) & = \{R(x,y) \rightarrow S(y,x),\ S(y,x) \rightarrow 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) \rightarrow S(x_0,x_k)\} \\
\pi(\sigma \sqsubseteq \varrho) & = \{\sigma(x,y) \rightarrow \varrho(x,y)\} \\
\pi(\sigma = \varrho) & = \{\sigma(x,y) \rightarrow \varrho(x,y),\ \varrho(x,y) \rightarrow \sigma(x,y)\} \\
\pi(\mathsf{Disj}(C_1,\ldots,C_k)) & = \{\pi_{(x)}(C_i) \wedge \pi_{(x)}(C_j) \rightarrow \bot \mid 1 \leq i < j \leq k\} \\
\pi(\mathsf{Disj}(R_1,\ldots,R_k)) & = \{R_i(x,y) \wedge R_j(x,y) \rightarrow \bot \mid 1 \leq i < j \leq k\} \\
\pi(\mathsf{Disj}(\sigma_1,\ldots,\sigma_k)) & = \{\sigma_i(x,y) \wedge \sigma_j(x,y) \rightarrow \bot \mid 1 \leq i < j \leq k\} \\
\pi(\mathsf{Func}(R)) & = \{R(x,y) \wedge R(x,z) \rightarrow y = z\} \\
\pi(\mathsf{Func}(\sigma)) & = \{\sigma(x,y) \wedge \sigma(x,z) \rightarrow y \doteq z\} \\
\pi(\mathsf{InvFunc}(R)) & = \{R(y,x) \wedge R(z,x) \rightarrow y = z\} \\
\pi(\mathsf{Refl}(R)) & = \{R(x,x)\} \\
\pi(\mathsf{Irref}(R)) & = \{R(x,x) \rightarrow \bot\} \\
\pi(\mathsf{Sym}(R)) & = \{R(x,y) \rightarrow R(y,x)\} \\
\pi(\mathsf{Asym}(R)) & = \{R(x,y) \wedge R(y,x) \rightarrow \bot\} \\
\pi(\mathsf{Trans}(R)) & = \{R(x,y) \wedge R(y,z) \rightarrow R(x,z)\} \\
\pi(A(a)) & = \{A(a)\} \\
\pi(C(a)) & = \{A(a)\} \cup \pi(A \sqsubseteq C) \text{ for the case when } C \text{ is a complex concept,} \\
 & \qquad\qquad \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{array}
$$

$\pi(\mathsf{Key}(C, R_1, \ldots, R_k, \sigma_1, \ldots, \sigma_h)) = [\pi_{(x)}(C) \wedge \pi_{(y)}(C) \wedge R_1(x,u_1) \wedge R_1(y,u_1) \wedge \ldots \wedge R_k(x,u_k) \wedge R_k(y,u_k) \wedge \sigma_1(x,v_1) \wedge \sigma_1(y,v_1) \wedge \ldots \wedge \sigma_h(x,v_h) \wedge \sigma_h(y,v_h) \rightarrow x = y]$

Fig. 2. A translation π for axioms of OWL 2 RL. All variables for $\pi(.)$ like x, y, z, u, v are fresh (new) variables. Variables y and z used for $\pi_{(x)}(.)$ are also fresh variables.

Given an axiom φ of OWL 2 RL, define:

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

Given a knowledge base KB in OWL 2 RL, 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.

Define a *negative clause* to be a formula of the form $a \neq b$, $\neg r(a,b)$, $\neg\sigma(a,d)$ or $\varphi_1 \wedge \ldots \wedge \varphi_k \rightarrow \bot$, where $\varphi_1, \ldots, \varphi_k$ are atomic formulas.

Lemma A.1. *Let KB be a knowledge base in OWL 2 RL. 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:

$$l\varphi := \alpha \mid r^-(t,t') \mid l\varphi \wedge l\varphi \mid l\varphi \vee l\varphi$$
$$r\varphi := \alpha \mid r^-(t,t') \mid r\varphi \wedge r\varphi \mid l\varphi \rightarrow r\varphi \mid l\varphi \rightarrow \bot$$

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 translating φ to the conjunctive normal form by using the distributive laws of $\wedge$ and $\vee$ results in $\varphi_1 \vee \ldots \vee \varphi_k$ (where each φ_i does not contains $\vee$) 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) in the left hand side of some $\rightarrow$ 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 straightforward 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, if ψ is an ABox assertion and $\mathcal{I} \models \pi_3(\psi)$ then $\mathcal{I} \models \psi$. Therefore, every model of $\pi_3(KB)$ is also a model of KB.

Consider the third assertion of the lemma.

Let φ be a query in the language of KB.

Suppose 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 suppose 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 $\pi(C(a))$, $A^{\mathcal{I}'} = C^{\mathcal{I}}$. 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-ary predicate of DPreds different from $=$ and $\doteq$, and i is any natural number between 1 and k such that the i-th argument of p is of type *IType*:

$$x = x$$
$$x = y \rightarrow y = x$$
$$x = y \wedge y = z \rightarrow x = z$$
$$x_i = x'_i \wedge p(x_1, \ldots, x_k) \rightarrow p(x_1, \ldots, x_{i-1}, x'_i, x_{i+1}, \ldots, x_k).$$

Since unique name assumption is adopted for literals (i.e. data constants), to deal with the equality predicate $\doteq$ between literals we use a simpler approach: having a ground atom $d_1 \doteq d_2$, 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 $=$ but without $\doteq$. 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 $=$ as a normal predicate. Let $\mathcal{I}$ be the interpretation specified as follows:

- $\Delta^{\mathcal{I}}_o$ is the set of all individuals occurring in $\mathcal{H}$
- $\Delta^{\mathcal{I}}_d$ is the set of all data constants occurring in $\mathcal{H}$
- for every k-ary predicate $p \in$ DPreds,

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

Observe that $=^{\mathcal{I}}$ is a congruence of $\mathcal{I}$. Clearly, the quotient $\mathcal{I}/_{=}$ of $\mathcal{I}$ by the congruence $=^{\mathcal{I}}$ 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 recall each of the theorems before presenting its proof.

Theorem 3.2.

1. *OWL 2 RL$^+$ is a maximal fragment (w.r.t. allowed features) of OWL 2 RL 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 $\doteq$.[5] By Lemma A.1(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 lemma follows from this fact and Example 3.1. The second assertion of the lemma follows from Lemma A.1(3). ◁

Theorem 3.3. *Let KB be a knowledge base in OWL 2 RL, 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.

Suppose 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 suppose 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 suppose 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 of only axioms which are translated to negative clauses or clauses of the form $(\psi \to y \doteq z)$ (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 4.1 and 4.2 we need the following definitions and lemma.

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 16.

[5] One can apply also the translation specified in [23] 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  I := {φ ∈ KB | φ is of the form ⊥ or ψ or ¬ψ where ψ is a ground atom};
2  foreach formula (i.e. program clause or constraint clause or assertion) ρ of KB ∪ EqAxioms do
3      reorder the body of ρ so that ρ = (φ_1 ∧ ... ∧ φ_k ∧ ψ_1 ∧ ... ∧ ψ_h → ξ), where k ≥ 0, h ≥ 0, and
       φ_1, ..., φ_k are atoms of predicates from DPreds, and ψ_1, ..., ψ_k are atoms of predicates from
       ECPreds;
4      foreach instance ρ' = (φ'_1 ∧ ... ∧ φ'_k ∧ ψ'_1 ∧ ... ∧ ψ'_h → ξ') of ρ such that, for every 1 ≤ i ≤ k,
       φ'_i ∈ I or φ'_i is of the form d ≐ d do
5          if ψ'_1, ..., ψ'_h are all true (note that these atoms are ground) then
6              if ξ' is ⊥ or a ground atom then add ξ' to 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 I
10 if I changed during the last execution of Step 2 then goto Step 2;
11 return I
```

Lemma A.2. *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 $d_1 \doteq d_2$, 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 $d \doteq d$.*
3. *The set $\mathcal{I}$ can be computed in polynomial time in the size of the ABox of KB.* ◁

The proof of this lemma is straightforward.

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 4.1.

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.1, it can be seen that, for every query φ in the language of KB, $KB \models \varphi$ iff $KB' \models \varphi$. By Lemma A.2, it follows that both OWL 2 eRL and OWL 2 eRL$^+$ have PTIME data complexity. ◁

Theorem 4.2 can be proved analogously.