WORL: a Web Ontology Rule Language*

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. We develop a new Web ontology rule language, called WORL, which combines a variant of OWL 2 RL with eDatalog$^\neg$. We allow additional features like negation, the minimal number restriction and unary external checkable predicates to occur at the left hand side of concept inclusion axioms. Some restrictions are adopted to guarantee a translation into eDatalog$^\neg$. We also develop the well-founded semantics for WORL and the standard semantics for stratified WORL (SWORL) via translation into eDatalog$^\neg$. Both WORL and SWORL have PTime data complexity. In contrast to the existing combined formalisms, in WORL and SWORL negation in concept inclusion axioms is interpreted using nonmonotonic semantics.

Keywords: OWL 2 RL, Datalog with negation, Semantic Web, rule languages.

1 Introduction

In recent years, the Semantic Web area has been rapid developed and attracted lots of attention from researchers. A central idea of the Semantic Web is that more accurate web search should take into account ontologies as a proper bridge among users and search engines. Therefore, Web Ontology Language (OWL), built on the top of XML and RDF, serves as an important tool for specifying ontologies and reasoning about them. Together with rule languages it serves as a main knowledge representation formalism for the Semantic Web.

Description logics (DLs) have been accepted as the main logical foundation of OWL. Such logics represent the domain of interest in terms of concepts, individuals, and roles. A concept is interpreted as a set of individuals, while a role is interpreted as a binary relation between individuals. A knowledge base in a DL consists of an RBox of role axioms, a TBox of terminological axioms and an ABox of facts about individuals.

The second version OWL 2 of OWL, recommended by the W3C consortium in 2009, is based on the description logic $\mathcal{SROIQ}$ [12]. 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 $\mathcal{EL}$ [3, 4], DL-Lite [5] and DLP (Description Logic Programs) [11], respectively.

As fragments of DLs with PTime data complexity, apart from the families $\mathcal{EL}$, DL-Lite, and DLP there are also Horn-$\mathcal{SHIQ}$ [14], Horn-$\mathcal{SROIQ}$ [22], and the deterministic Horn fragments of $\mathcal{ALC}$ and regular description logics [20, 21].

Rule languages are a very useful knowledge representation formalism with applications to the Semantic Web. Some fragments of DLs like DLP [11] can be translated into a rule language. But most importantly, rule languages can be combined with DLs to give more expressive knowledge representation formalisms for the Semantic Web.

* This is a revised and extended version of the conference paper [7]. Comparing to [7], in the current paper we additionally provide proofs of the results as well as a direct method for checking stratifiability of TBoxes.

An early attempt to achieve such a combination was to use SWRL [13], a rule language using only concept names, role names and the equality predicate. Without restrictions its combination with OWL DL is an undecidable formalism.

A knowledge base in other combined languages is usually specified as a pair $\langle \mathcal{O}, \mathcal{P} \rangle$, where $\mathcal{O}$ is an ontology in some DL and $\mathcal{P}$ is a set of rules, e.g., specified in Datalog or its suitable extension, which can use concept names and role names. Interaction between $\mathcal{O}$ and $\mathcal{P}$ is either one-way ($\mathcal{O}$ affects $\mathcal{P}$) or two-way (where $\mathcal{P}$ may also affect $\mathcal{O}$). The approach of defining a knowledge base as a pair $\langle \mathcal{O}, \mathcal{P} \rangle$ is adopted in a considerable number of works, including [8] (on $\mathcal{AL}$-*log*), [16] (on *CARIN*), [19] (on *DL-safe rules*), [24] (on $\mathcal{DL}$+*log*), [18, 15] (on *hybrid MKNF*), [9] (on *hybrid programs*), [23] (on *OntoDLV*), [10] (on *dl-programs*). In these works, if negation is allowed in $\mathcal{P}$ then $\mathcal{P}$ and its interaction with $\mathcal{O}$ are interpreted using some nonmonotonic semantics (e.g., the stable model semantics, the MKNF semantics or the well-founded semantics); however, $\mathcal{O}$ is always interpreted using the usual (monotonic) semantics.

In the current paper we treat such a pair $\langle \mathcal{O}, \mathcal{P} \rangle$ just as a layer and study the case when $\mathcal{O}$ can be translated to an eDatalog$^\neg$ program and $\mathcal{P}$ is an eDatalog$^\neg$ program, where eDatalog$^\neg$ extends Datalog$^\neg$ by allowing two basic types (for individuals and data constants), external checkable predicates and the equality predicate (between individuals). Concept names and role names are allowed both in heads and bodies of program clauses. Our approach is novel in the following aspects:

- Negation in $\mathcal{O}$ is interpreted using a nonmonotonic semantics (the well-founded semantics, or the standard semantics for stratified knowledge bases); this differs from all the above mentioned works [8–10, 15, 16, 18, 19, 23, 24].
- We combine $\mathcal{O}$ and $\mathcal{P}$ into one set (called a layer, which is divided into a TBox consisting of concept inclusion axioms / program clauses and an ABox consisting of facts). This allows for a tighter integration between DLs and rules. It may seem similar to the approach of SWRL, but we also allow ordinary predicates, use a nonmonotonic semantics for negation, and design the language appropriately to get decidability and PTime data complexity.
- To reflect modularity of ontologies (e.g., the import feature of ontologies) we define a knowledge base to be a hierarchy of layers (a tree or a rooted directed acyclic graph of layers). Each layer in turn may be stratifiable and divided further into strata. The granulation is not substantial for the well-founded semantics, as the whole knowledge base will be flattened to a set of program clauses and facts. However, when each layer of the considered knowledge base is stratifiable and the standard semantics is used for it, layers not only emphasize modularity but also affect the semantics (flattening the knowledge base may result in an unstratifiable layer).

The Web ontology rule language we define in this paper, WORL, combines a variant of OWL 2 RL with eDatalog$^\neg$. Similarly to our previous work on OWL 2 eRL$^+$ [6], we:

- disallow those features of OWL 2 RL that play the role of constraints (i.e., the ones that are translated to negative clauses of the form $\varphi \rightarrow \bot$);
- allow unary external checkable predicates;
- allow additional features like negation and the constructor $\geq n\, R.C$ to occur at the left hand side of $\sqsubseteq$ in concept inclusion axioms.

Some restrictions are adopted for the additional features to guarantee a translation of WORL programs into eDatalog$^\neg$. We also define the rule language SWORL (stratified WORL) and develop the well-founded semantics for WORL and the standard semantics for SWORL via translation into eDatalog$^\neg$. Both WORL and SWORL have PTime data complexity.

The rest of this paper is structured as follows. In Section 2 we recall eDatalog$^\neg$, stratified eDatalog$^\neg$, and their semantics. In Section 3 we present WORL, a translation of WORL into eDatalog$^\neg$, and its well-founded semantics. In Section 4 we present SWORL and its standard

semantics. Section 5 contains an example illustrating the usefulness of SWORL, and Section 6 concludes this work.

2 Preliminaries

We denote the set of *concept names* by CNames, and denote the set of *role names* by RNames. From the point of view of OWL, there are two basic types: *individual* (i.e. *object*) and *literal* [17] (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 $P(IType \times LType)$. For simplicity, we do not provide specific data types like integer, real or string. 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. By a set of *defined predicates* we mean:

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

With each k-ary predicate from OPreds we associate its 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 the type $P(LType^k)$. We assume that each predicate from ECPreds has a fixed meaning which is checkable in the following sense:

> if p is a k-ary predicate from ECPreds and $d_1, \ldots, d_k$ are constants of *LType*, then the truth value of $p(d_1, \ldots, d_k)$ is fixed and computable in polynomial time (in the number of bits used for $d_1, \ldots, d_k$).

For example, one may want to use the binary predicates $>$, $\geq$, $<$, $\leq$ on real numbers with the usual semantics. We assume there is only one equality predicate '=', which belongs to OPreds and has the type $P(IType \times IType)$. For data constants we assume the Unique Names Assumption instead.

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 *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:

- every individual a to an element $a^{\mathcal{I}} \in \Delta_o^{\mathcal{I}}$,
- every literal d to a unique[4] 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 predicate of type $P(T_1 \times \ldots \times T_k)$ in DPreds different from '=' to a subset of $\Delta_1 \times \ldots \times \Delta_k$, where $\Delta_i = \Delta_o^{\mathcal{I}}$ if $T_i = IType$, and $\Delta_i = \Delta_d^{\mathcal{I}}$ if $T_i = LType$,
- predicate '=' to a congruence of $\mathcal{I}$.[5]

A *Herbrand interpretation* is a set of ground atoms of predicates from DPreds. An *ABox* is a finite Herbrand interpretation.

The *size* of a ground atom is the number of bits used for its representation. The *size* of an ABox is the sum of the sizes of its atoms.

[4] I.e., if $d_1 \neq d_2$ then $d_1^{\mathcal{I}} \neq d_2^{\mathcal{I}}$.

[5] Recall that a congruence is an equivalence relation preserving functions and relations occurring in the language.

By *EqAxioms* we denote the following set of axioms:

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

where p is any k-ary predicate of DPreds different from '=' and i is any natural number between 1 and k such that the i-th argument of p is of type *IType*.

A Herbrand interpretation $\mathcal{H}$ is *closed w.r.t. EqAxioms* if for every ground instance $\varphi_1 \wedge \ldots \wedge \varphi_k \rightarrow \psi$ (with $k \geq 0$) of an axiom in *EqAxioms* using the individuals and data constants occurring in $\mathcal{H}$, if $\{\varphi_1, \ldots, \varphi_k\} \subseteq \mathcal{H}$ then $\psi \in \mathcal{H}$.

Given a Herbrand interpretation $\mathcal{H}$ that is closed w.r.t. *EqAxioms*, 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 $\mathcal{I}$. We call the quotient $\mathcal{I}/_{=}$ of $\mathcal{I}$ by the congruence $=^{\mathcal{I}}$ the *traditional interpretation corresponding to* $\mathcal{H}$.

2.1 The Rule Language eDatalog$^\neg$

In [6] we defined eDatalog as an extension of Datalog with the equality predicate, external checkable predicates, and a relaxed range-restrictedness condition. In this subsection we define the rule language eDatalog$^\neg$ similarly as an extension of Datalog$^\neg$, but using the full range-restrictedness condition.

An *eDatalog$^\neg$ program clause* is a formula of the form

$$(\varphi_1 \wedge \ldots \wedge \varphi_h \wedge \neg\psi_1 \wedge \ldots \wedge \neg\psi_k \wedge \xi_1 \wedge \ldots \wedge \xi_l \wedge \neg\zeta_1 \wedge \ldots \wedge \neg\zeta_m) \rightarrow \alpha \qquad (1)$$

where $h, k, l, m \geq 0$, $\varphi_1, \ldots, \varphi_h, \psi_1, \ldots, \psi_k, \alpha$ are atoms of predicates from DPreds, and $\xi_1, \ldots, \xi_l, \zeta_1, \ldots, \zeta_m$ are atoms of predicates from ECPreds, with the property that every variable occurring in α or some ψ_i, ξ_i or ζ_i occurs also in some atom φ_j (this is the *range-restrictedness condition*).

The atom α in (1) is called the *head* of the program clause. If p is the predicate of α then the clause is called a *program clause defining* p. The whole formula at the left hand side of $\rightarrow$ in (1) is called the *body* of the program clause.

An *eDatalog$^\neg$ program* is a finite set of eDatalog$^\neg$ program clauses. An *eDatalog$^\neg$ knowledge base* is a pair $\langle \mathcal{P}, \mathcal{A}\rangle$ consisting of an eDatalog$^\neg$ program $\mathcal{P}$ and an ABox $\mathcal{A}$. A *query* is defined to be a formula that can be the body of an eDatalog$^\neg$ program clause.

Example 2.1. Let $\mathcal{P}$ be the following eDatalog$^\neg$ program:

$$[\, \mathit{acceptable}(X) \wedge \mathit{hasPrice}(X, Y) \wedge \\ \mathit{acceptable}(X') \wedge \mathit{hasPrice}(X', Y') \wedge Y < Y' \,] \rightarrow \mathit{excluded}(X')$$
$$\mathit{acceptable}(X) \wedge \neg \mathit{excluded}(X) \rightarrow \mathit{preferable}(X)$$

and let $\mathcal{A} = \{\mathit{acceptable}(a), \mathit{acceptable}(b), \mathit{hasPrice}(a, 100), \mathit{hasPrice}(b, 120)\}$. Then $KB = \langle \mathcal{P}, \mathcal{A}\rangle$ is an eDatalog$^\neg$ knowledge base. Here, '$<$' is an external checkable predicate with the usual semantics; X and X' are variables of type *IType*; Y and Y' are variables of type *LType*; a and b are objects (of type *IType*); 100 and 120 are data constants (of type *LType*). □

2.2 The Well-Founded Semantics of eDatalog$^\neg$

Let $\langle \mathcal{P}, \mathcal{A} \rangle$ be an eDatalog$^\neg$ knowledge base. By $\mathcal{P}_{\mathcal{A}}^{gr}$ we denote the set of all ground instances of the program clauses of $\mathcal{P} \cup \mathit{EqAxioms}$ that use only individuals and data constants occurring in $\mathcal{P}$ or $\mathcal{A}$. By $\mathcal{P}_{\mathcal{A}}$ we denote the set of all clauses:

$$(\varphi_1 \wedge \ldots \wedge \varphi_h \wedge \neg\psi_1 \wedge \ldots \wedge \neg\psi_k) \rightarrow \alpha$$

such that $\mathcal{P}_{\mathcal{A}}^{gr}$ contains a program clause

$$(\varphi_1 \wedge \ldots \wedge \varphi_h \wedge \neg\psi_1 \wedge \ldots \wedge \neg\psi_k \wedge \xi_1 \wedge \ldots \wedge \xi_l \wedge \neg\zeta_1 \wedge \ldots \wedge \neg\zeta_m) \rightarrow \alpha \tag{2}$$

where all $\xi_1, \ldots, \xi_l$ are true and all $\zeta_1, \ldots, \zeta_m$ are false (by the fixed meaning of external checkable predicates).

Note that if $\langle \mathcal{P}, \mathcal{A} \rangle$ is an eDatalog$^\neg$ knowledge base then $\mathcal{P}_{\mathcal{A}}$ and $\mathcal{P}_{\mathcal{A}} \cup \mathcal{A}$ are ground Datalog$^\neg$ programs.

Example 2.2. Consider the eDatalog$^\neg$ knowledge base $\langle \mathcal{P}, \mathcal{A} \rangle$ given in Example 2.1. Then $\mathcal{P}_{\mathcal{A}}$ consists of a number of ground instances of clauses of *EqAxioms* and the following clauses:

$$\begin{aligned}
&[\, \mathit{acceptable}(a) \wedge \mathit{hasPrice}(a, 100) \wedge \\
&\quad \mathit{acceptable}(a) \wedge \mathit{hasPrice}(a, 120)\,] \rightarrow \mathit{excluded}(a) \\
&[\, \mathit{acceptable}(a) \wedge \mathit{hasPrice}(a, 100) \wedge \\
&\quad \mathit{acceptable}(b) \wedge \mathit{hasPrice}(b, 120)\,] \rightarrow \mathit{excluded}(b) \\
&[\, \mathit{acceptable}(b) \wedge \mathit{hasPrice}(b, 100) \wedge \\
&\quad \mathit{acceptable}(a) \wedge \mathit{hasPrice}(a, 120)\,] \rightarrow \mathit{excluded}(a) \\
&[\, \mathit{acceptable}(b) \wedge \mathit{hasPrice}(b, 100) \wedge \\
&\quad \mathit{acceptable}(b) \wedge \mathit{hasPrice}(b, 120)\,] \rightarrow \mathit{excluded}(b) \\
&\mathit{acceptable}(a) \wedge \neg \mathit{excluded}(a) \rightarrow \mathit{preferable}(a) \\
&\mathit{acceptable}(b) \wedge \neg \mathit{excluded}(b) \rightarrow \mathit{preferable}(b).
\end{aligned}$$

Note that the predicate '$<$' does not occur any more in $\mathcal{P}_{\mathcal{A}}$. □

Let $\mathsf{WF}_{\langle \mathcal{P}, \mathcal{A} \rangle}$ denote the *well-founded model* of the Datalog$^\neg$ program $\mathcal{P}_{\mathcal{A}} \cup \mathcal{A}$ (as defined, e.g., in [1]). It is a set of formulas of the form α or $\neg\alpha$ where α is a ground atom. We call $\mathsf{WF}_{\langle \mathcal{P}, \mathcal{A} \rangle}$ the *well-founded (Herbrand) model* of $\langle \mathcal{P}, \mathcal{A} \rangle$.

Given a query

$$\varphi = (\varphi_1 \wedge \ldots \wedge \varphi_h \wedge \neg\psi_1 \wedge \ldots \wedge \neg\psi_k \wedge \xi_1 \wedge \ldots \wedge \xi_l \wedge \neg\zeta_1 \wedge \ldots \wedge \neg\zeta_m),$$

a (correct) *answer* to φ w.r.t. $\langle \mathcal{P}, \mathcal{A} \rangle$ and the *well-founded semantics* is a ground substitution θ for all the variables of φ such that $\{\varphi_1\theta, \ldots, \varphi_h\theta, \neg\psi_1\theta, \ldots, \neg\psi_k\theta\} \subseteq \mathsf{WF}_{\langle \mathcal{P}, \mathcal{A} \rangle}$, all $\xi_1\theta$, …, $\xi_l\theta$ are true and all $\zeta_1\theta$, …, $\zeta_m\theta$ are false (by the fixed meaning of external checkable predicates).

The *data complexity* of eDatalog$^\neg$ w.r.t. the well-founded semantics is the complexity of the problem of finding all answers to a query φ w.r.t. an eDatalog$^\neg$ knowledge base $\langle \mathcal{P}, \mathcal{A} \rangle$ and the well-founded semantics, measured w.r.t. the size of $\mathcal{A}$ when assuming that φ, $\mathcal{P}$, DPreds are fixed and checking whether a ground atom of an external checkable predicate is true or false can be done in polynomial time.

Proposition 2.3. *The data complexity of eDatalog$^\neg$ w.r.t. the well-founded semantics is in* PTime.

Proof. Let $\langle \mathcal{P}, \mathcal{A} \rangle$ be an eDatalog$^\neg$ knowledge base. The set $\mathcal{P}^{gr}_{\mathcal{A}}$ can be constructed in polynomial time and has polynomial size in the size of $\mathcal{A}$. As the truth values of the atoms of external checkable predicates that occur in $\mathcal{P}^{gr}_{\mathcal{A}}$ can be computed in polynomial time, $\mathcal{P}_{\mathcal{A}}$ can also be constructed in polynomial time and has polynomial size in the size of $\mathcal{A}$. It is well-known that the well-founded model of the Datalog$^\neg$ program $\mathcal{P}_{\mathcal{A}} \cup \mathcal{A}$ can be constructed in polynomial time and has polynomial size in the size of $\mathcal{P}_{\mathcal{A}} \cup \mathcal{A}$ (see, e.g., [1]). Thus, $\mathsf{WF}_{\langle \mathcal{P}, \mathcal{A} \rangle}$ can be constructed in polynomial time and has polynomial size in the size of $\mathcal{A}$. Consequently, answering queries to $\langle \mathcal{P}, \mathcal{A} \rangle$ w.r.t. the well-founded semantics can be done in polynomial time in the size of $\mathcal{A}$. □

2.3 Stratified eDatalog$^\neg$

A *stratification* of an eDatalog$^\neg$ program $\mathcal{P}$ is a sequence of eDatalog$^\neg$ programs $\mathcal{P}_1, \ldots, \mathcal{P}_n$ such that:

- $\{\mathcal{P}_1, \ldots, \mathcal{P}_n\}$ is a partition of $\mathcal{P} \cup \mathit{EqAxioms}$,
- for some mapping $f : \mathrm{DPreds} \to \{1, \ldots, n\}$, every predicate $p \in \mathrm{DPreds}$ satisfies the following conditions:
  - the program clauses in $\mathcal{P} \cup \mathit{EqAxioms}$ defining p are in $\mathcal{P}_{f(p)}$,
  - if $\mathcal{P} \cup \mathit{EqAxioms}$ contains a program clause defining p in the form

$$(\varphi_1 \wedge \ldots \wedge \varphi_h \wedge \neg\psi_1 \wedge \ldots \wedge \neg\psi_k \wedge \xi_1 \wedge \ldots \wedge \xi_l \wedge \neg\zeta_1 \wedge \ldots \wedge \neg\zeta_m) \to \alpha$$

    then for every $1 \leq i \leq h$ and $1 \leq j \leq k$:
    - if p'_i is the predicate of φ_i then $f(p'_i) \leq f(p)$,
    - if p''_j is the predicate of ψ_j then $f(p''_j) < f(p)$.

Given a stratification $\mathcal{P}_1, \ldots, \mathcal{P}_n$ of $\mathcal{P}$, each $\mathcal{P}_i$ is called a *stratum* of the stratification, and f is called the *stratification mapping*. Let us emphasize that $f(\text{`='}) \leq f(p)$ for all $p \in \mathrm{DPreds}$.

An eDatalog$^\neg$ program $\mathcal{P}$ is called a *stratified eDatalog$^\neg$ program* if it has a stratification. It is called a *semipositive eDatalog$^\neg$ program* if it has a stratification with only one stratum.[6]

Example 2.4. The eDatalog$^\neg$ program $\mathcal{P}$ given in Example 2.1 is a stratified eDatalog$^\neg$ program with two strata. Each program clause of $\mathcal{P}$ forms a stratum. □

A pair $\langle \mathcal{P}, \mathcal{A} \rangle$ is called a *stratified eDatalog$^\neg$ knowledge base* if it is an eDatalog$^\neg$ knowledge base with $\mathcal{P}$ being a stratified eDatalog$^\neg$ program.

2.4 The Standard Semantics of Stratified eDatalog$^\neg$

In this subsection, we extend the standard semantics of stratified Datalog$^\neg$ to stratified eDatalog$^\neg$.

Given an eDatalog$^\neg$ knowledge base $KB = \langle \mathcal{P}, \mathcal{A} \rangle$ with $\mathcal{P}$ being a semipositive eDatalog$^\neg$ program, the *standard Herbrand model* of KB, denoted by $\mathcal{H}_{KB}$, is the smallest Herbrand interpretation $\mathcal{H}$ such that $\mathcal{A} \subseteq \mathcal{H}$ and for every ground instance

$$(\varphi_1 \wedge \ldots \wedge \varphi_h \wedge \neg\psi_1 \wedge \ldots \wedge \neg\psi_k \wedge \xi_1 \wedge \ldots \wedge \xi_l \wedge \neg\zeta_1 \wedge \ldots \wedge \neg\zeta_m) \to \alpha$$

of a program clause in $\mathcal{P} \cup \mathit{EqAxioms}$,

if $\{\varphi_1, \ldots, \varphi_h\} \subseteq \mathcal{H}$, $\{\psi_1, \ldots, \psi_k\} \cap \mathcal{H} = \emptyset$, all $\xi_1, \ldots, \xi_l$ are true and all $\zeta_1, \ldots, \zeta_m$ are false, then $\alpha \in \mathcal{H}$.

Lemma 2.5. *Given an eDatalog$^\neg$ knowledge base $KB = \langle \mathcal{P}, \mathcal{A} \rangle$ with $\mathcal{P}$ being a semipositive eDatalog$^\neg$ program, the standard Herbrand model of KB can be computed in polynomial time and has polynomial size in the size of $\mathcal{A}$.*

[6] Facts supplied to that only stratum are kept separately, e.g., in an ABox.

Proof. Recall that we use $\mathcal{P}^{gr}_{\mathcal{A}}$ to denote the set of all ground instances of the program clauses of $\mathcal{P} \cup \mathit{EqAxioms}$ that use only individuals and data constants occurring in $\mathcal{P}$ or $\mathcal{A}$. Clearly, $\mathcal{P}^{gr}_{\mathcal{A}}$ has polynomial size in the size of $\mathcal{A}$ (when $\mathcal{P}$ is fixed). Let $\mathcal{P}'_{\mathcal{A}}$ be the set of all the program clauses

$$\varphi_1 \wedge \ldots \wedge \varphi_h \rightarrow \alpha$$

such that $\mathcal{P}^{gr}_{\mathcal{A}}$ contains a program clause

$$(\varphi_1 \wedge \ldots \wedge \varphi_h \wedge \neg\psi_1 \wedge \ldots \wedge \neg\psi_k \wedge \xi_1 \wedge \ldots \wedge \xi_l \wedge \neg\zeta_1 \wedge \ldots \wedge \neg\zeta_m) \rightarrow \alpha$$

where $\{\psi_1, \ldots, \psi_k\} \cap \mathcal{A} = \emptyset$, all $\xi_1, \ldots, \xi_l$ are true and all $\zeta_1, \ldots, \zeta_m$ are false (by the fixed meaning of external checkable predicates). The set $\mathcal{P}'_{\mathcal{A}}$ is a Datalog program, which can be computed in polynomial time and has polynomial size in the size of $\mathcal{A}$. The least Herbrand model of $\mathcal{P}'_{\mathcal{A}}$ can be computed in polynomial time and has polynomial size in the size of $\mathcal{P}'_{\mathcal{A}}$ (see, e.g., [1]). Thus, it can be computed in polynomial time and has polynomial size in the size of $\mathcal{A}$. That model is the same as the standard Herbrand model of KB. □

Given an eDatalog$^\neg$ knowledge base $KB = \langle \mathcal{P}, \mathcal{A} \rangle$, where $\mathcal{P}$ is an eDatalog$^\neg$ program with stratification $\mathcal{P}_1, \ldots, \mathcal{P}_n$, the *standard Herbrand model* of KB is defined to be $\mathcal{H}_{KB} = \mathcal{A}_{n+1}$, where $\mathcal{A}_1 = \mathcal{A}$, and $\mathcal{A}_{i+1} = \mathcal{H}_{\langle \mathcal{P}_i, \mathcal{A}_i \rangle}$ for $1 \leq i \leq n$. The *standard model* of such a knowledge base KB is defined to be the traditional interpretation corresponding to $\mathcal{H}_{KB}$ and is denoted by $\mathcal{M}_{KB}$.

Example 2.6. Consider the stratified eDatalog$^\neg$ knowledge base KB given in Example 2.1. We have that $\mathcal{A}_1 = \mathcal{A}$, $\mathcal{A}_2 = \mathcal{A}_1 \cup \{a = a, b = b, \mathit{excluded}(b)\}$ and $\mathcal{H}_{KB} = \mathcal{A}_3 = \mathcal{A}_2 \cup \{\mathit{preferable}(a)\}$. □

Proposition 2.7. *For a stratified eDatalog$^\neg$ knowledge base $KB = \langle \mathcal{P}, \mathcal{A} \rangle$, all possible stratifications of $\mathcal{P}$ give the same standard Herbrand model $\mathcal{H}_{KB}$. That is, the standard Herbrand model $\mathcal{H}_{KB}$ does not depend on how $\mathcal{P}$ is stratified.*

Proof. Every stratification of $\mathcal{P}$ specifies a stratification for $\mathcal{P}^{gr}_{\mathcal{A}}$ and a stratification for $\mathcal{P}_{\mathcal{A}}$. It is well-known that the standard Herbrand model $\mathcal{H}_{\langle \mathcal{P}_{\mathcal{A}}, \mathcal{A} \rangle}$ of the Datalog$^\neg$ knowledge base $\langle \mathcal{P}_{\mathcal{A}}, \mathcal{A} \rangle$ does not depend on stratification of $\mathcal{P}_{\mathcal{A}}$ (see, e.g., [1]). Furthermore, it is the same as the standard Herbrand model of $KB = \langle \mathcal{P}, \mathcal{A} \rangle$. Hence, all possible stratifications of $\mathcal{P}$ give the same standard Herbrand model $\mathcal{H}_{KB}$. □

Given a stratified eDatalog$^\neg$ knowledge base $KB = \langle \mathcal{P}, \mathcal{A} \rangle$ and a query φ, a (correct) *answer* to φ w.r.t. KB and the *standard semantics* is a ground substitution θ for all the variables of φ such that $\mathcal{M}_{KB} \models \varphi\theta$, where $\models$ is the satisfaction relation defined in the usual way.

The *data complexity* of stratified eDatalog$^\neg$ w.r.t. the standard semantics is defined as usual.

Proposition 2.8. *The data complexity of stratified eDatalog$^\neg$ w.r.t. the standard semantics is in* PTime.

Proof. Let $KB = \langle \mathcal{P}, \mathcal{A} \rangle$ be any stratified eDatalog$^\neg$ knowledge base. By the definition of $\mathcal{H}_{KB}$ and by Lemma 2.5, $\mathcal{H}_{KB}$ can be computed in polynomial time and has polynomial size in the size of $\mathcal{A}$. Hence, answering queries to KB w.r.t. the standard semantics can be done in polynomial time in the size of $\mathcal{A}$. □

As in the case of Datalog$^\neg$, the standard semantics of stratified eDatalog$^\neg$ coincides with the well-founded semantics when restricting to queries of the form $(\varphi_1 \wedge \ldots \wedge \varphi_h \wedge \xi_1 \wedge \ldots \wedge \xi_l \wedge \neg\zeta_1 \wedge \ldots \wedge \neg\zeta_m)$, where $\varphi_1, \ldots, \varphi_h$ are atoms of predicates from DPreds and $\xi_1, \ldots, \xi_l, \zeta_1, \ldots, \zeta_m$ are atoms of predicates from ECPreds.

3 The Web Ontology Rule Language WORL

3.1 Syntax and Notation of WORL

We use:

- the truth symbol $\top$ to denote *owl:Thing* [17],
- a and b to denote *individuals* (i.e. *objects*),
- d to denote a *literal* (i.e. a data constant),
- A and B to denote concept names (i.e. *Class* elements [17]),
- C and D to denote *concepts* (i.e. *ClassExpression* elements [17]),
- $lC_{\pm}$ and lC to denote concepts like a *subClassExpression* of [17],
- rC to denote a concept like a *superClassExpression* of [17],
- eC to denote a concept like an *equivClassExpression* of [17],
- DT to denote a *data type* (i.e. a *Datatype* of [17]),
- DR to denote a *data range* (i.e. a *DataRange* of [17]),
- p_{uec} to denote a unary predicate from ECPreds,
- r and s to denote *object role names* (i.e. *ObjectProperty* elements [17]),
- R and S to denote *object roles* (i.e. *ObjectPropertyExpression* elements [17]),
- σ and ϱ to denote *data role names* (i.e. *DataProperty* elements [17]).

The families of R, DR, $lC_{\pm}$, lC, rC, eC are defined by the following BNF grammar, where $n \geq 2$:

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

Notice the occurrences of $lC_{\pm}$ in the definition of lC. They are accompanied by lC or R to guarantee the so called *safeness* (*range-restrictedness*) condition. Comparing with [6], it can be seen that $\neg A$, $\geq n\, R.lC_{\pm}$ and $\exists \sigma.p_{uec}$ for $lC_{\pm}$ are additional features w.r.t. OWL 2 RL.

The class constructor *ObjectOneOf* [17] 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: Func (Functional), $\mathsf{InvFunc}$ (Inverse-Functional), Sym (Symmetric), Trans (Transitive), Key (HasKey).

A *DL TBox axiom*, like a *ClassAxiom* or a *DatatypeDefinition* or a *HasKey* axiom of OWL 2 RL [17], is an expression of one of the following forms, where $h, k \geq 0$ and $h + k \geq 1$:

$$\begin{gathered}
lC \sqsubseteq rC,\ eC = eC', \\
DT = DR,\ \mathsf{Key}(lC_{\pm}, R_1, \ldots, R_h, \sigma_1, \ldots, \sigma_k).
\end{gathered} \tag{3}$$

An *RBox axiom*, like an *ObjectPropertyAxiom* or a *DataPropertyAxiom* of OWL 2 RL [17], is an expression of one of the following forms:

$$\begin{gathered}
R_1 \circ \ldots \circ R_k \sqsubseteq S,\ R = S,\ R = S^{-},\ \exists R.\top \sqsubseteq rC,\ \top \sqsubseteq \forall R.rC, \\
\mathsf{Func}(R),\ \mathsf{InvFunc}(R),\ \mathsf{Sym}(R),\ \mathsf{Trans}(R), \\
\sigma \sqsubseteq \varrho,\ \sigma = \varrho,\ \exists \sigma \sqsubseteq rC,\ \top \sqsubseteq \forall \sigma.DR.
\end{gathered} \tag{4}$$

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 as in [17]. One can classify these latter axioms as DL TBox axioms instead of RBox axioms. Similarly, $\mathsf{Key}(\ldots)$ axioms can be classified as RBox axioms instead.

We accept the following definitions:

- A (WORL) *TBox axiom* is either a DL TBox axiom (as defined by (3)) or an RBox axiom (as defined by (4)) or an eDatalog$^\neg$ program clause.
- A (WORL) *TBox* is a finite set of TBox axioms.
- A *WORL knowledge layer* is a pair $\mathcal{L} = \langle \mathcal{T}, \mathcal{A} \rangle$ consisting of a TBox $\mathcal{T}$ and an ABox $\mathcal{A}$.

Note that we defined an ABox to be a finite set of ground atoms of predicates from DPreds. If one wants to add an assertion of the form $C(a)$ to a WORL knowledge layer $\langle \mathcal{T}, \mathcal{A} \rangle$, where C is a complex concept belonging to the rC family, he or she can add the assertion $A(a)$ to $\mathcal{A}$ and add the axiom $A \sqsubseteq C$ to $\mathcal{T}$, where A is a fresh concept name.

WORL knowledge bases are defined inductively as follows:

- a WORL knowledge layer is a WORL knowledge base,
- if $\mathcal{L}$ is a WORL knowledge layer and $KB_1, \ldots, KB_k$ are WORL knowledge bases then the pair $KB = \langle \mathcal{L}, \{KB_1, \ldots, KB_k\} \rangle$ is a WORL knowledge base.

A WORL knowledge base $\langle \mathcal{L}, \{KB_1, \ldots, KB_k\} \rangle$ can be thought of as an ontology with $\mathcal{L}$ being a set of direct statements, and $KB_1, \ldots, KB_k$ being subontologies.

3.2 Translating WORL into eDatalog$^\neg$

We first define a translation π that translates a TBox axiom to a set of formulas of classical first-order logic. After that we will refine π to get a translation that converts a TBox to an eDatalog$^\neg$ program.

For an eDatalog$^\neg$ program clause φ, let $\pi(\varphi) = \{\varphi\}$. For a DL TBox axiom or an RBox axiom φ, let $\pi(\varphi)$ be defined as in Figure 1, where $\pi_{(x)}$ is an auxiliary translation 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 $\exists R.C$, $\exists R.\top$, $\geq n\, R.C$, $\exists \sigma.DR$ or $\exists \sigma.p_{uec}$, 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 $\rightarrow$.

The translation π is very intuitive and we use it also for specifying the meanings of TBox axioms. The meaning 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 [17].

Given an interpretation $\mathcal{I}$ and a DL TBox axiom or an RBox axiom φ, we define that $\mathcal{I} \models \varphi$ iff $\mathcal{I} \models \pi(\varphi)$, where the latter satisfaction relation $\models$ is defined as usual. We say that $\mathcal{I}$ is a model of a TBox $\mathcal{T}$, denoted by $\mathcal{I} \models \mathcal{T}$, if $\mathcal{I} \models \varphi$ for all $\varphi \in \mathcal{T}$.

We define:

$$
\begin{array}{l}
\pi_{2,l}(\xi) = \{\xi\} \text{ if } \xi \text{ is not of any of the forms } \varphi \wedge \psi,\ \varphi \vee \psi,\ r^-(x,y) \\
\pi_{2,l}(r^-(x,y)) = \{r(y,x)\} \\
\pi_{2,l}(\varphi \vee \psi) = \pi_{2,l}(\varphi) \cup \pi_{2,l}(\psi) \\
\pi_{2,l}(\varphi \wedge \psi) = \{\varphi' \wedge \psi' \mid \varphi' \in \pi_{2,l}(\varphi) \text{ and } \psi' \in \pi_{2,l}(\psi)\}
\end{array}
$$

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

$\pi(\top \sqsubseteq C)$	$= \{\pi_{(x)}(C)\}$
$\pi(\exists\sigma \sqsubseteq C)$	$= \{\sigma(x,y) \rightarrow \pi_{(x)}(C)\}$
$\pi(C \sqsubseteq D)$	$= \{\pi_{(x)}(C) \rightarrow \pi_{(x)}(D)\}$
$\pi(C = D)$	$= \{\pi_{(x)}(C) \rightarrow \pi_{(x)}(D),\ \pi_{(x)}(D) \rightarrow \pi_{(x)}(C)\}$
$\pi(DT = DR)$	$= \{\pi_{(x)}(DT) \rightarrow \pi_{(x)}(DR),\ \pi_{(x)}(DR) \rightarrow \pi_{(x)}(DT)\}$
$\pi(R = S)$	$= \{R(x,y) \rightarrow S(x,y),\ S(x,y) \rightarrow R(x,y)\}$
$\pi(R = S^-)$	$= \{R(x,y) \rightarrow S(y,x),\ S(y,x) \rightarrow R(x,y)\}$
$\pi(R_1 \circ \ldots \circ R_k \sqsubseteq S)$	$= \{R_1(x_0,x_1) \wedge \ldots \wedge R_k(x_{k-1},x_k) \rightarrow S(x_0,x_k)\}$
$\pi(\sigma \sqsubseteq \varrho)$	$= \{\sigma(x,y) \rightarrow \varrho(x,y)\}$
$\pi(\sigma = \varrho)$	$= \{\sigma(x,y) \rightarrow \varrho(x,y),\ \varrho(x,y) \rightarrow \sigma(x,y)\}$
$\pi(\mathsf{Func}(R))$	$= \{R(x,y) \wedge R(x,z) \rightarrow y = z\}$
$\pi(\mathsf{InvFunc}(R))$	$= \{R(y,x) \wedge R(z,x) \rightarrow y = z\}$
$\pi(\mathsf{Sym}(R))$	$= \{R(x,y) \rightarrow R(y,x)\}$
$\pi(\mathsf{Trans}(R))$	$= \{R(x,y) \wedge R(y,z) \rightarrow R(x,z)\}$
$\pi(\mathsf{Key}(C,R_1,\ldots,R_h,\sigma_1,\ldots,\sigma_k))$	$= \{[\pi_{(x)}(C) \wedge \pi_{(y)}(C) \wedge$ $\bigwedge_{1\leq i\leq h}(R_i(x,u_i) \wedge R_i(y,u_i)) \wedge \bigwedge_{1\leq i\leq k}(\sigma_i(x,v_i) \wedge \sigma_i(y,v_i))] \rightarrow x = y\}$
$\pi_{(x)}(DT)$	$= DT(x)$
$\pi_{(x)}(DT \sqcap DR)$	$= DT(x) \wedge \pi_{(x)}(DR)$
$\pi_{(x)}(A)$	$= A(x)$
$\pi_{(x)}(\neg A)$	$= \neg A(x)$
$\pi_{(x)}(\{a\})$	$= (x = a)$
$\pi_{(x)}(C \sqcap D)$	$= \pi_{(x)}(C) \wedge \pi_{(x)}(D)$
$\pi_{(x)}(C \sqcup D)$	$= \pi_{(x)}(C) \vee \pi_{(x)}(D)$
$\pi_{(x)}(\forall R.C)$	$= R(x,y) \rightarrow \pi_{(y)}(C)$
$\pi_{(x)}(\exists R.C)$	$= R(x,y) \wedge \pi_{(y)}(C)$
$\pi_{(x)}(\exists R.\{a\})$	$= R(x,a)$
$\pi_{(x)}(\exists R.\top)$	$= R(x,y)$
$\pi_{(x)}(\geq n\, R.C)$	$= \bigwedge_{1\leq i\leq n}(R(x,y_i) \wedge \pi_{(y_i)}(C)) \wedge \bigwedge_{1\leq i\neq j\leq n} \neg(y_i = y_j)$
$\pi_{(x)}(\forall\sigma.DR)$	$= \sigma(x,y) \rightarrow \pi_{(y)}(DR)$
$\pi_{(x)}(\exists\sigma.DR)$	$= \sigma(x,y) \wedge \pi_{(y)}(DR)$
$\pi_{(x)}(\exists\sigma.p_{uec})$	$= \sigma(x,y) \wedge p_{uec}(y)$
$\pi_{(x)}(\exists\sigma.\{d\})$	$= \sigma(x,d)$
$\pi_{(x)}(\leq 1\, R.C)$	$= R(x,y) \wedge R(x,z) \wedge \pi_{(y)}(C) \wedge \pi_{(z)}(C) \rightarrow y = z$
$\pi_{(x)}(\leq 1\, R.\top)$	$= R(x,y) \wedge R(x,z) \rightarrow y = z$

Fig. 1. The translation π for DL TBox axioms and RBox axioms. 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.

We also need the following definitions of π_3:

- if φ is an eDatalog$^\neg$ program clause then $\pi_3(\varphi) = \{\varphi\}$,
- if φ is a DL TBox axiom or an RBox axiom φ then

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

- if φ is a TBox $\mathcal{T}$ then $\pi_3(\mathcal{T}) = \bigcup_{\varphi \in \mathcal{T}} \pi_3(\varphi)$.

Example 3.1. For $\varphi = (\exists r.(A_1 \sqcup A_2) \sqsubseteq \forall r.B)$, we have:

$$\begin{aligned}\pi(\varphi) &= \{r(x,y) \wedge (A_1(y) \vee A_2(y)) \rightarrow (r(x,z) \rightarrow B(z))\}\\ \pi_3(\varphi) &= \{r(x,y) \wedge A_1(y) \wedge r(x,z) \rightarrow B(z),\\ &\qquad r(x,y) \wedge A_2(y) \wedge r(x,z) \rightarrow B(z)\}.\end{aligned}$$

□

Lemma 3.2. *For any (WORL) TBox $\mathcal{T}$, $\pi_3(\mathcal{T})$ is an eDatalog$^\neg$ program equivalent to $\mathcal{T}$ in the sense that, for any interpretation $\mathcal{I}$, $\mathcal{I} \models \pi_3(\mathcal{T})$ iff $\mathcal{I} \models \mathcal{T}$.*

Proof. Let ψ denote a formula of classical first-order logic. It can be proved by induction on the structure of ψ that $\pi_{2,l}(\psi)$ and $\pi_2(\psi)$ are sets of formulas such that, for any interpretation $\mathcal{I}$,

- $\mathcal{I} \models \bigvee \pi_{2,l}(\psi)$ iff $\mathcal{I} \models \psi$,
- $\mathcal{I} \models \bigwedge \pi_2(\psi)$ iff $\mathcal{I} \models \psi$.

Consequently, for any interpretation $\mathcal{I}$ and any DL TBox axiom or RBox axiom φ, $\mathcal{I} \models \pi_3(\varphi)$ iff $\mathcal{I} \models \pi(\varphi)$. By definition, $\mathcal{I} \models \varphi$ iff $\mathcal{I} \models \pi(\varphi)$. Therefore, $\pi_3(\mathcal{T})$ is equivalent to $\mathcal{T}$.

It remains to show that $\pi_3(\mathcal{T})$ is an eDatalog$^\neg$ program. In the following, let α denote an atomic formula. We define the families of $l\psi_\pm$, $l\psi$ and $r\psi$ as follows (by using BNF grammar for $l\psi_\pm$ and $r\psi$):

$$\begin{aligned}l\psi_\pm &:= \alpha \mid \neg\alpha \mid r^-(t,t') \mid l\psi_\pm \wedge l\psi_\pm \mid l\psi_\pm \vee l\psi_\pm\\ l\psi &:= l\psi_\pm \text{ with the safeness condition}\\ r\psi &:= \alpha \mid r^-(t,t') \mid r\psi \wedge r\psi \mid l\psi \rightarrow r\psi\end{aligned}$$

where a formula ψ of the $l\psi_\pm$ family satisfies the safeness condition if translating ψ to the conjunctive normal form by using the distributive laws of $\wedge$ and $\vee$ results in $\psi_1 \vee \ldots \vee \psi_k$ (where each ψ_i does not contains $\vee$) such that every variable occurring in some ψ_i occurs (among others) in some positive atom of ψ_i.

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 ψ of the $l\psi$ family such that translating ψ to the conjunctive normal form by using the distributive laws of $\wedge$ and $\vee$ results in $\psi_1 \vee \ldots \vee \psi_k$ (where each ψ_i does not contains $\vee$) such that variable x occurs in each ψ_i,
- if C is a concept of the rC family then $\pi_{(x)}(C)$ is a formula of the $r\psi$ 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 ψ is a formula of the $l\psi$ family then $\pi_{2,l}(\psi)$ is a set of formulas of the $l\psi$ family without the connective $\vee$ and atoms of the form $r^-(t,t')$,

- if φ is a DL TBox axiom or an RBox axiom then $\pi(\varphi)$ is a set of formulas of the $r\psi$ family such that every variable occurring in a formula from $\pi(\varphi)$ occurs (among others) in some positive atom of the formula in the left hand side of some $\rightarrow$,
- if φ is a DL TBox axiom or an RBox axiom and $\psi \in \pi(\varphi)$ then $\pi_2(\psi)$ is a set of eDatalog$^\neg$ program clauses.

Therefore, $\pi_3(\mathcal{T})$ is an eDatalog$^\neg$ program. □

3.3 The Well-Founded Semantics of WORL

The *flattened version* of a WORL knowledge base KB is the WORL knowledge layer denoted by *flatten*(KB) and defined as follows:

- if KB is a layer then *flatten*(KB) $= KB$,
- else if $KB = \langle \mathcal{L}, \{KB_1, \ldots, KB_k\}\rangle$, $\mathcal{L} = \langle \mathcal{T}, \mathcal{A}\rangle$ and *flatten*(KB_i) $= \langle \mathcal{T}_i, \mathcal{A}_i\rangle$ for $1 \leq i \leq k$, then

$$\textit{flatten}(KB) = \langle \mathcal{T} \cup \mathcal{T}_1 \cup \ldots \cup \mathcal{T}_k, \mathcal{A} \cup \mathcal{A}_1 \cup \ldots \cup \mathcal{A}_k\rangle.$$

Given a WORL knowledge base KB with *flatten*(KB) $= \langle \mathcal{T}, \mathcal{A}\rangle$, the *well-founded (Herbrand) model* of KB, denoted by WF_{KB}, is defined to be the well-founded model of the eDatalog$^\neg$ knowledge base $KB' = \langle \pi_3(\mathcal{T}), \mathcal{A}\rangle$. A (correct) *answer* to a query φ w.r.t. that KB and the *well-founded semantics* is an answer to φ w.r.t. that KB' and the well-founded semantics of eDatalog$^\neg$.

The *data complexity* of WORL w.r.t. the well-founded semantics is the complexity of the problem of finding all answers to a query φ w.r.t. a WORL knowledge base KB and the well-founded semantics, measured w.r.t. the sum of the sizes of all ABoxes used in KB when assuming that DPreds, φ and all the TBoxes used in KB are fixed and checking whether a ground atom of an external checkable predicate is true or false can be done in polynomial time. The following theorem immediately follows from Proposition 2.3.

Theorem 3.3. *The data complexity of WORL w.r.t. the well-founded semantics is in* PTime. □

4 Stratified WORL

A TBox $\mathcal{T}$ is said to be *stratifiable* if $\pi_3(\mathcal{T})$ is a stratified eDatalog$^\neg$ program. In the appendix we present a direct method for checking stratifiability of a TBox without using translation.

A WORL knowledge layer $\langle \mathcal{T}, \mathcal{A}\rangle$ is called a *SWORL knowledge layer* if $\mathcal{T}$ is a stratifiable TBox. A WORL knowledge base is called a *SWORL knowledge base* if it is either a SWORL knowledge layer or a pair $\langle \mathcal{L}, \{KB_1, \ldots, KB_k\}\rangle$ where $\mathcal{L}$ is a SWORL knowledge layer and each KB_i is a SWORL knowledge base.

Note that flattening a SWORL knowledge base $\langle \mathcal{L}, \{KB_1, \ldots, KB_k\}\rangle$ may result in a WORL knowledge layer that is not stratifiable.

4.1 The Standard Semantics of SWORL

Let KB be a SWORL knowledge base. The *standard Herbrand model* of KB, denoted by $\mathcal{H}_{KB}$, is defined as follows:

- If KB is a SWORL knowledge layer $\langle \mathcal{T}, \mathcal{A}\rangle$ then $\mathcal{H}_{KB}$ is the standard Herbrand model of the stratified eDatalog$^\neg$ knowledge base $\langle \pi_3(\mathcal{T}), \mathcal{A}\rangle$.
- If $KB = \langle \mathcal{L}, \{KB_1, \ldots, KB_k\}\rangle$ and $\mathcal{L} = \langle \mathcal{T}, \mathcal{A}\rangle$ then $\mathcal{H}_{KB}$ is the standard Herbrand model of the stratified eDatalog$^\neg$ knowledge base $\langle \pi_3(\mathcal{T}), \mathcal{A} \cup \mathcal{H}_{KB_1} \cup \ldots \cup \mathcal{H}_{KB_k}\rangle$.

The *standard model* of a SWORL knowledge base KB is defined to be the traditional interpretation corresponding to $\mathcal{H}_{KB}$ and is denoted by $\mathcal{M}_{KB}$.

The notion of *answer* to a query w.r.t. a SWORL knowledge base and the data complexity of SWORL are defined as usual:

- Given a SWORL knowledge base KB and a query φ, a (correct) *answer* to φ w.r.t. KB and the *standard semantics* is a ground substitution θ for all the variables of φ such that $\mathcal{M}_{KB} \models \varphi\theta$, where $\models$ is the satisfaction relation defined in the usual way.
- The *data complexity* of SWORL w.r.t. the standard semantics is the complexity of the problem of finding all answers to a query φ w.r.t. a SWORL knowledge base KB and the standard semantics, measured w.r.t. the sum of the sizes of all ABoxes used in KB when assuming that DPreds, φ, the structure of KB and all the TBoxes used in KB are fixed and checking whether a ground atom of an external checkable predicate is true or false can be done in polynomial time.

Theorem 4.1. *The data complexity of SWORL w.r.t. the standard semantics is in* PTime.

Proof. Let KB be a SWORL knowledge base and n be the sum of the sizes of all ABoxes used in KB. We prove by induction on the structure of KB that the standard Herbrand model $\mathcal{H}_{KB}$ of KB can be computed in polynomial time and has polynomial size in n:

- If KB is a SWORL knowledge layer $\langle \mathcal{T}, \mathcal{A} \rangle$ then $\mathcal{H}_{KB}$ is the standard Herbrand model of the stratified eDatalog$^\neg$ knowledge base $\langle \pi_3(\mathcal{T}), \mathcal{A} \rangle$, and by Lemma 2.5, $\mathcal{H}_{KB}$ can be computed in polynomial time and has polynomial size in n.
- If $KB = \langle \langle \mathcal{T}, \mathcal{A} \rangle, \{KB_1, \ldots, KB_k\} \rangle$ then:
  - By the inductive assumption, $\mathcal{H}_{KB_1}$, ..., $\mathcal{H}_{KB_k}$ can be computed in polynomial time and have polynomial size in n.
  - $\mathcal{H}_{KB}$ is the standard Herbrand model of the stratified eDatalog$^\neg$ knowledge base $\langle \pi_3(\mathcal{T}), \mathcal{A} \cup \mathcal{H}_{KB_1} \cup \ldots \cup \mathcal{H}_{KB_k} \rangle$, and as stated in the proof of Proposition 2.8, $\mathcal{H}_{KB}$ can be computed in polynomial time and has polynomial size in the size of $\mathcal{A} \cup \mathcal{H}_{KB_1} \cup \ldots \cup \mathcal{H}_{KB_k}$.
  - Hence, $\mathcal{H}_{KB}$ can be computed in polynomial time and has polynomial size in n.

As a consequence, the data complexity of SWORL w.r.t. the standard semantics is in PTime. □

The standard semantics of SWORL coincides with the well-founded semantics when restricting to SWORL knowledge bases that are single layers and to queries of the form $(\varphi_1 \wedge \ldots \wedge \varphi_h \wedge \xi_1 \wedge \ldots \wedge \xi_l \wedge \neg\zeta_1 \wedge \ldots \wedge \neg\zeta_m)$, where φ_1, ..., φ_h are atoms of predicates from DPreds and ξ_1, ..., ξ_l, ζ_1, ..., ζ_m are atoms of predicates from ECPreds.

5 Example: Apartment Renting

To illustrate introduced ideas, we discuss apartment renting, a common activity that is often tedious and time-consuming. The example is based on the one of [2]. The difference is that we use SWORL instead of defeasible logic.

We begin by presenting the potential renter's requirements:

- Carlos is looking for an apartment of at least 45 m^2 with at least two bedrooms. If it is on the third floor or higher, the house must have an elevator. Also, pet animals must be allowed.
- Carlos is willing to pay \$300 for a centrally located 45 m^2 apartment, and \$250 for a similar flat in the suburbs. In addition, he is willing to pay an extra \$5 per m^2 for a larger apartment, and \$2 per m^2 for a garden.

– He is unable to pay more than $400 in total. If given the choice, he would go for the cheapest option. His second priority is the presence of a garden; his lowest priority is additional space.

We use the following predicates to describe properties of apartments:

– $\mathit{hasSize}(X, Y)$: Y is the size of apartment X,
– $\mathit{bedrooms}(X, Y)$: apartment X has Y bedrooms,
– $\mathit{hasPrice}(X, Y)$: Y is the rent price of apartment X,
– $\mathit{floor}(X, Y)$: apartment X is on the Y^{th} floor,
– $\mathit{garden}(X, Y)$: apartment X has a garden of size Y,
– $\mathit{withLift}(X)$: there is an elevator in the house of X,
– $\mathit{allowsPets}(X)$: pets are allowed in apartment X,
– $\mathit{central}(X)$: apartment X is centrally located.

The predicates *hasSize*, *bedrooms*, *hasPrice*, *floor* and *garden* are data role names, while the predicates *withLift*, *allowsPets* and *central* are concept names. These predicates are specified by ABox assertions.

We define a number of predicates. The first one is *withGarden*, specified by:

$$\mathit{garden}(X, Y) \rightarrow \mathit{withGarden}(X) \tag{5}$$

We use predicate $\mathit{offers}(X, N, Y, Z)$ defined as follows:

$$[\mathit{hasSize}(X, Y) \wedge \mathit{central}(X) \wedge \neg \mathit{withGarden}(X)] \rightarrow \mathit{offers}(X, 1, Y, 0) \tag{6}$$
$$[\mathit{hasSize}(X, Y) \wedge \mathit{central}(X) \wedge \mathit{garden}(X, Z)] \rightarrow \mathit{offers}(X, 2, Y, Z) \tag{7}$$
$$[\mathit{hasSize}(X, Y) \wedge \neg \mathit{central}(X) \wedge \neg \mathit{withGarden}(X)] \rightarrow \mathit{offers}(X, 3, Y, 0) \tag{8}$$
$$[\mathit{hasSize}(X, Y) \wedge \neg \mathit{central}(X) \wedge \mathit{garden}(X, Z)] \rightarrow \mathit{offers}(X, 4, Y, Z) \tag{9}$$

The predicate $\mathit{offers}(X, N, Y, Z)$ means Carlos is willing to pay $f(N, Y, Z)$ dollars for apartment X, where $f(N, Y, Z)$ is defined as

$$f(N, Y, Z) = \begin{cases} 300 + 5(Y - 45) & \text{if } N = 1 \\ 300 + 5(Y - 45) + 2.Z & \text{if } N = 2 \\ 250 + 5(Y - 45) & \text{if } N = 3 \\ 250 + 5(Y - 45) + 2.Z & \text{if } N = 4 \end{cases}$$

This function is used only to specify the external checkable predicate

$$\mathit{tooExpensive}(N, Y, Z, P) \equiv (f(N, Y, Z) < P)$$

which in turn is used in the following program clause:

$$\begin{array}{l}[\, \mathit{offers}(X, N, Y, Z) \wedge \mathit{hasPrice}(X, P) \wedge \\ \quad \mathit{tooExpensive}(N, Y, Z, P)\,] \rightarrow \mathit{excluded}_0(X)\end{array} \tag{10}$$

Thus, $\mathit{excluded}_0(X)$ means apartment X is unacceptable. Apartments acceptable to Carlos are defined by the following DL TBox axiom:

$$\begin{array}{l}[\, \exists \mathit{hasSize}.(\geq 45) \sqcap \exists \mathit{bedrooms}.(\geq 2) \sqcap (\exists \mathit{floor}.(\leq 2) \sqcup \mathit{withLift}) \sqcap \\ \quad \mathit{allowsPets} \sqcap \neg \mathit{excluded}_0 \sqcap \exists \mathit{hasPrice}.(\leq 400)\,] \sqsubseteq \mathit{acceptable}\end{array} \tag{11}$$

In the above axiom, (≥ 45), (≥ 2), (≤ 2) and (≤ 400) are unary external checkable predicates.

Among the acceptable apartments, the cheapest ones are preferable:

$$\begin{array}{l}[\, \mathit{acceptable}(X) \wedge \mathit{hasPrice}(X, Y) \wedge \\ \mathit{acceptable}(X') \wedge \mathit{hasPrice}(X', Y') \wedge Y < Y'\,] \rightarrow \mathit{excluded}_1(X')\end{array} \tag{12}$$

$$acceptable(X) \wedge \neg excluded_1(X) \rightarrow preferable_1(X) \tag{13}$$

Among the cheapest apartments that are acceptable, the ones with a garden are more preferable:

$$\begin{array}{c}[\, preferable_1(X) \wedge \neg withGarden(X) \wedge \\ preferable_1(X') \wedge withGarden(X')\,] \rightarrow excluded_2(X)\end{array} \tag{14}$$

$$preferable_1(X) \wedge \neg excluded_2(X) \rightarrow preferable_2(X) \tag{15}$$

Among those apartments, Carlos will rent a largest one:

$$\begin{array}{c}[\, preferable_2(X) \wedge hasSize(X, Y) \wedge \\ preferable_2(X') \wedge hasSize(X', Y') \wedge Y < Y'\,] \rightarrow excluded_3(X)\end{array} \tag{16}$$

$$preferable_2(X) \wedge \neg excluded_3(X) \rightarrow mayRent(X) \tag{17}$$

In the program clauses (12) and (16), '$<$' is a binary external checkable predicate.

Let $\mathcal{T} = \{(5), \ldots, (17)\}$. It is a stratifiable TBox. Only (11) is a DL TBox axiom, while the other axioms are eDatalog$^\neg$ program clauses. The program clauses (5), (13), (15) and (17) can also be expressed as DL TBox axioms, treating *withGarden*, *acceptable*, $excluded_1$, $preferable_1$, $excluded_2$, $preferable_2$, $excluded_3$ and *mayRent* as concept names.

Translating the TBox $\mathcal{T}$ to a stratified eDatalog$^\neg$ program $\mathcal{P} = \pi_3(\mathcal{T})$, the DL TBox axiom (11) is replaced by the following eDatalog$^\neg$ program clauses:

$$\begin{array}{c}[\, hasSize(X, Y_1) \wedge Y_1 \geq 45 \wedge bedrooms(X, Y_2) \wedge Y_2 \geq 2 \wedge \\ floor(X, Y_3) \wedge Y_3 \leq 2 \wedge allowsPets(X) \wedge \neg excluded_0(X) \wedge \\ hasPrice(X, Y_4) \wedge Y_4 \leq 400\,] \rightarrow acceptable(X)\end{array} \tag{18}$$

$$\begin{array}{c}[\, hasSize(X, Y_1) \wedge Y_1 \geq 45 \wedge bedrooms(X, Y_2) \wedge Y_2 \geq 2 \wedge \\ withLift(X) \wedge allowsPets(X) \wedge \neg excluded_0(X) \wedge \\ hasPrice(X, Y_4) \wedge Y_4 \leq 400\,] \rightarrow acceptable(X)\end{array} \tag{19}$$

A possible stratification of $\mathcal{P}$ is: {(5)}, {(6), (7), (8), (9), (10)}, {(18), (19), (12)}, {(13), (14)}, {(15), (16)}, {(17)}.

Let $\mathcal{A}$ be the ABox consisting of the ground atoms of predicates *bedrooms*, *hasSize*, *central*, *floor*, *withLift*, *allowsPets*, *garden* and *hasPrice* that reflect the information contained in the following table:

flat	bedrooms	size	central	floor	lift	pets	garden	price
a1	1	50	yes	1	no	yes		300
a2	2	45	yes	0	no	yes		335
a3	2	65	no	2	no	yes		350
a4	2	55	no	1	yes	no	15	330
a5	3	55	yes	0	no	yes	15	350
a6	2	60	yes	3	no	no		370
a7	3	65	yes	1	no	yes	12	375

For example, $bedrooms(a1, 1)$, $hasSize(a1, 50)$, $central(a1)$, $floor(a1, 1)$, $allowsPets(a1)$ and $hasPrice(a1, 300)$ are the atoms of $\mathcal{A}$ that involve apartment $a1$. As ABoxes contain only positive information, only atom $withLift(a4)$ of predicate *withLift* occurs in $\mathcal{A}$.

The pair $KB = \langle \mathcal{T}, \mathcal{A} \rangle$ is a SWORL knowledge layer (and a SWORL knowledge base). The standard Herbrand model $\mathcal{H}_{KB}$ contains atoms $acceptable(X)$ only for $X \in \{a3, a5, a7\}$ and atoms $preferable_1(X)$ only for $X \in \{a3, a5\}$. Only atom $preferable_2(a5)$ of predicate $preferable_2$ and atom $mayRent(a5)$ of predicate *mayRent* occur in $\mathcal{H}_{KB}$.

6 Conclusions

We have developed the Web ontology rule languages WORL and SWORL together with the well-founded semantics for WORL and the standard semantics for SWORL. These rule languages have PTime data complexity and, as demonstrated by the example given in Section 5, are expressive for formalizing practical problems.

As WORL can be translated into eDatalog$^\neg$ and SWORL can be translated into stratified eDatalog$^\neg$, the languages WORL and SWORL are not more expressive than eDatalog$^\neg$ and stratified eDatalog$^\neg$, respectively. However, WORL and SWORL allow using also syntax of description logic (and hence also OWL). This has the same benefits as in the case OWL 2 RL compared to eDatalog, and is very useful for applications of the Semantic Web. As Web ontology rule languages, WORL and SWORL have the advantage of using efficient computational methods of Datalog$^\neg$ (extended for eDatalog$^\neg$).

Using nonmonotonic semantics for negation in concept inclusion axioms is a novelty of our approach. Modularity of SWORL is also worth mentioning.

Acknowledgements

This work was supported by Polish National Science Centre grants 2011/01/B/ST6/02769 and 2011/01/B/ST6/02759.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison Wesley (1995)
2. Antoniou, G., van Harmelen, F.: A Semantic Web Primer. MIT Press (2004)
3. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope. In: Proceedings of IJCAI'2005. pp. 364–369. Morgan-Kaufmann Publishers (2005)
4. Baader, F., Brandt, S., Lutz, C.: Pushing the EL envelope further. In: Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions (2008)
5. Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. J. Autom. Reasoning 39(3), 385–429 (2007)
6. Cao, S., Nguyen, L., Szałas, A.: On the Web ontology rule language OWL 2 RL. In: Proceedings of ICCCI 2011. LNCS, vol. 6922, pp. 254–264. Springer (2011)
7. Cao, S., Nguyen, L., Szalas, A.: WORL: a Web ontology rule language. In: Proceedings of KSE'2011. pp. 32–39. IEEE Computer Society (2011)
8. Donini, F., Lenzerini, M., Nardi, D., Schaerf, A.: AL-log: Integrating Datalog and description logics. J. Intell. Inf. Syst. 10(3), 227–252 (1998)
9. Drabent, W., Maluszynski, J.: Well-founded semantics for hybrid rules. In: Proceedings of RR'2007. LNCS, vol. 4524, pp. 1–15. Springer (2007)
10. Eiter, T., Ianni, G., Lukasiewicz, T., Schindlauer, R.: Well-founded semantics for description logic programs in the Semantic Web. ACM Trans. Comput. Log. 12(2), 11 (2011)
11. Grosof, B., Horrocks, I., Volz, R., Decker, S.: Description logic programs: combining logic programs with description logic. In: Proceedings of WWW'2003. pp. 48–57 (2003)
12. Horrocks, I., Kutz, O., Sattler, U.: The even more irresistible SROIQ. In: Proceedings of KR'2006. pp. 57–67. AAAI Press (2006)
13. Horrocks, I., Patel-Schneider, P., Bechhofer, S., Tsarkov, D.: OWL rules: A proposal and prototype implementation. J. Web Sem. 3(1), 23–40 (2005)
14. Hustadt, U., Motik, B., Sattler, U.: Reasoning in description logics by a reduction to disjunctive Datalog. J. Autom. Reasoning 39(3), 351–384 (2007)
15. Knorr, M., Alferes, J., Hitzler, P.: A coherent well-founded model for hybrid MKNF knowledge bases. In: Proceedings of ECAI'2008. Frontiers in Artificial Intelligence and Applications, vol. 178, pp. 99–103. IOS Press (2008)
16. Levy, A., Rousset, M.C.: Combining Horn rules and description logics in CARIN. Artif. Intell. 104(1-2), 165–209 (1998)
17. Motik, B., Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C., Calvanese, D., Carroll, J., Giacomo, G.D., Hendler, J., Herman, I., Parsia, B., Patel-Schneider, P., Ruttenberg, A., Sattler, U., Schneider, M.: OWL 2 RL. `http://www.w3.org/TR/owl2-profiles/#OWL_2_RL` (2009)
18. Motik, B., Rosati, R.: Reconciling description logics and rules. J. ACM 57(5) (2010)

19. Motik, B., Sattler, U., Studer, R.: Query answering for OWL-DL with rules. J. Web Sem. 3(1), 41–60 (2005)
20. Nguyen, L.: A bottom-up method for the deterministic Horn fragment of the description logic $\mathcal{ALC}$. In: Proceedings of JELIA 2006. LNAI, vol. 4160, pp. 346–358. Springer (2006)
21. Nguyen, L.: Horn knowledge bases in regular description logics with PTime data complexity. Fundam. Inform. 104(4), 349–384 (2010)
22. Ortiz, M., Rudolph, S., Simkus, M.: Worst-case optimal reasoning for the Horn-DL fragments of OWL 1 and 2. In: Proceedings of KR'2010. AAAI Press (2010)
23. Ricca, F., Gallucci, L., Schindlauer, R., Dell'Armi, T., Grasso, G., Leone, N.: OntoDLV: An ASP-based system for enterprise ontologies. J. Log. Comput. 19(4), 643–670 (2009)
24. Rosati, R.: DL+log: Tight integration of description logics and disjunctive Datalog. In: Proceedings of KR'2006. pp. 68–78. AAAI Press (2006)

A Checking Stratifiability of TBoxes

We specify a dependency relation between the predicates occurring in a TBox for deciding whether the TBox is stratifiable.

For φ being either a concept of the lC family but not of the form $\geq n\, R.C$, or an expression of the form R, $R_1 \circ \ldots \circ R_k$, $\top$, σ or $\exists\sigma$, let $Preds_{-}(\varphi)$ be the set of the concept names that occur in φ under negation, and let $Preds_{+}(\varphi)$ be the set of the predicates from DPreds that occur in φ but do not belong to $Preds_{-}(\varphi)$.

For a concept C belonging to the rC family, define $LPreds_{+}(C)$, $LPreds_{-}(C)$ and $RPreds(C)$ as follows:[7]

- case $C = A$:
  $LPreds_{+}(C) = LPreds_{-}(C) = \emptyset$, $RPreds(C) = \{A\}$;
- case $C = D_1 \sqcap D_2$:
  $LPreds_{+}(C) = LPreds_{+}(D_1) \cup LPreds_{+}(D_2)$,
  $LPreds_{-}(C) = LPreds_{-}(D_1) \cup LPreds_{-}(D_2)$,
  $RPreds(C) = RPreds(D_1) \cup RPreds(D_2)$;
- case $C = \forall r.D$ or $C = \forall r^{-}.D$:
  $LPreds_{+}(C) = \{r\} \cup LPreds_{+}(D)$, $LPreds_{-}(C) = LPreds_{-}(D)$,
  $RPreds(C) = RPreds(D)$;
- case $C = \exists r.\{a\}$ or $C = \exists r^{-}.\{a\}$:
  $LPreds_{+}(C) = LPreds_{-}(C) = \emptyset$, $RPreds(C) = \{r\}$;
- case $C = \forall\sigma.DR$:
  $LPreds_{+}(C) = \{\sigma\}$, $LPreds_{-}(C) = \emptyset$,
  $RPreds(C)$ is the set of all data types occurring in DR;
- case $C = \exists\sigma.\{d\}$:
  $LPreds_{+}(C) = LPreds_{-}(C) = \emptyset$, $RPreds(C) = \{\sigma\}$;
- case $C = \leq 1\, r.D$ or $C = \leq 1\, r^{-}.D$:
  $LPreds_{+}(C) = \{r\} \cup Preds_{+}(D)$, $LPreds_{-}(C) = Preds_{-}(D)$,
  $RPreds(C) = \{\text{`}=\text{'}\}$;
- case $C = \leq 1\, r.\top$ or $C = \leq 1\, r^{-}.\top$:
  $LPreds_{+}(C) = \{r\}$, $LPreds_{-}(C) = \emptyset$, $RPreds(C) = \{\text{`}=\text{'}\}$.

Let $LPreds_{+}(R) = LPreds_{-}(R) = \emptyset$ and $RPreds(r) = RPreds(r^{-}) = \{r\}$. Let $LPreds_{+}(\sigma) = LPreds_{-}(\sigma) = \emptyset$ and $RPreds(\sigma) = \{\sigma\}$.

It can be proved that a TBox $\mathcal{T}$ is stratifiable if $\mathcal{T}$ does not use the concept constructor $\geq n\, R.C$ and there exists a function f from DPreds to positive natural numbers such that:

- for every eDatalog$^{\neg}$ program clause φ in $\mathcal{T} \cup EqAxioms$, if q is the predicate of the head of φ and p is a predicate from DPreds that occurs in the body of φ then $f(p) \leq f(q)$, and additionally, if p occurs under negation in φ then $f(p) < f(q)$;
- for every axiom of the form $\varphi \sqsubseteq \psi$ in $\mathcal{T}$ and for every $q \in RPreds(\psi)$:
  - for every $p \in Preds_{+}(\varphi) \cup LPreds_{+}(\psi)$, $f(p) \leq f(q)$;
  - for every $p \in Preds_{-}(\varphi) \cup LPreds_{-}(\psi)$, $f(p) < f(q)$;
- for every axiom of the form $\varphi = \psi$ in $\mathcal{T}$, all the predicates occurring in $\varphi = \psi$ have the same f value;
- for every axiom $\mathsf{Key}(C, R_1, \ldots, R_h, \sigma_1, \ldots, \sigma_k)$ in $\mathcal{T}$: $Preds_{-}(C) = \emptyset$ and, for every predicate p belonging to $Preds_{+}(C)$ or $\{\sigma_1, \ldots, \sigma_k\}$ or occurring in $R_1, \ldots, R_h$, $f(p) = f(\text{`}=\text{'})$;
- for every axiom of the form $\mathsf{Func}(R)$ or $\mathsf{InvFunc}(R)$ in $\mathcal{T}$, where $R = r$ or $R = r^{-}$, we have that $f(r) = f(\text{`}=\text{'})$.

[7] where L stands for "left of $\rightarrow$" and R stands for "right of $\rightarrow$"

To check whether a TBox $\mathcal{T}$ is stratifiable one can construct a graph of dependencies between the predicates occurring in $\mathcal{T}$. The condition $f(p) \leq f(q)$ (resp. $f(p) < f(q)$) is expressed by an edge with mark $+$ (resp. $-$) from vertex p to vertex q. The TBox is stratifiable if that graph does not contain any cycle with an edge marked by $-$.