RDFS with Attribute Equations via SPARQL Rewriting

Stefan Bischof[1,2] and Axel Polleres[1]

[1] Siemens AG Österreich, Siemensstraße 90, 1210 Vienna, Austria
[2] Vienna University of Technology, Favoritenstraße 9, 1040 Vienna, Austria

Abstract. In addition to taxonomic knowledge about concepts and properties typically expressible in languages such as RDFS and OWL, implicit information in an RDF graph may be likewise determined by arithmetic equations. The main use case here is exploiting knowledge about functional dependencies among numerical attributes expressible by means of such equations. While some of this knowledge can be encoded in rule extensions to ontology languages, we provide an arguably more flexible framework that treats attribute equations as first class citizens in the ontology language. The combination of ontological reasoning and attribute equations is realized by extending query rewriting techniques already successfully applied for ontology languages such as (the DL-Lite-fragment of) RDFS or OWL, respectively. We deploy this technique for rewriting SPARQL queries and discuss the feasibility of alternative implementations, such as rule-based approaches.

1 Introduction

A wide range of literature has discussed completion of data represented in RDF with implicit information through ontologies, mainly through taxonomic reasoning within a hierarchy of concepts (classes) and roles (properties) using RDFS and OWL. However, a lot of implicit knowledge within real world RDF data does not fall into this category: a large amount of emerging RDF data is composed of numerical attribute-value pairs assigned to resources which likewise contains a lot of implicit information, such as functional dependencies between numerical attributes expressible in the form of simple mathematical equations. These dependencies include unit conversions (e.g. between Fahrenheit and Celsius), or functional dependencies, such as the population density that can be computed from total population and area. Such numerical dependencies between datatype properties are not expressible in standard ontology languages such as RDFS or OWL. Rule based approaches also fail to encode such dependencies in the general case.

Example 1. Sample RDF data about cities, aggregated from sources such as DBPedia or Eurostat,[3] may contain data of various levels of completeness and using numerical attributes based on different units like

```
:Jakarta :tempHighC 33 .          :New_York :tempHighF 84 .
:New_York :population 8244910 .   :New_York :area_mile2 468.5 .
:Vienna :population 1714142 .     :Vienna :area_km2 414.6 .
:Vienna :populationDensity 4134 .    ...
```

[3] cf. `http://dbpedia.org/`, `http://eurostat.linked-statistics.org/`

Users familiar with SPARQL might expect to be able to ask for the population density, or for places with temperatures over 90°F with queries like

```
SELECT ?C ?P WHERE { ?C :populationDensity ?P }  or
SELECT ?C WHERE { ?C :tempHighF ?TempF FILTER(?TempF > 90) }
```

However, implicit answers from mathematical knowledge such as the following equations would not be returned by those queries:

$$tempHighC = (tempHighF - 32) \cdot 5/9$$
$$populationDensity = population \div area_{km2}$$

One might ask why such equations cannot be directly added to the terminological knowledge modeled in ontologies? We aim to show that it actually can; further, we present an approach how to extend the inference machinery for SPARQL query answering under ontologies to cater for such equations. Inspired by query rewriting algorithms for query answering over DL-Lite [3], we show how similar ideas can be deployed to extend a DL-Lite fragment covering the core of RDFS with so-called equation axioms.

We focus on query rewriting techniques rather than e.g. rule-based approaches such as SWRL [13], where the equations from Example 1 could encoded as

$$tempHighC(X, C) \Leftarrow tempHighF(X, F), C = (F - 32) \cdot 5/9 \quad (1)$$
$$populationDensity(X, PD) \Leftarrow population(X, P), area_{km2}(X, A), PD = P \div A \quad (2)$$

given respective arithmetic built-in support in a SWRL reasoner. However, note that these rules are not sufficient: (i) rule (1) is in the "wrong direction" for the query in Example 1, that is, we would need different variants of the rule for converting from $tempHighC$ to $tempHighF$ and vice versa; (ii) the above rules are not *DL safe* (i.e., we do not suffice to bind values only to explicitly named individuals, as we want to compute *new* values) which potentially leads to termination problems in rule-based approaches (and as we will see it actually does in existing systems). Our approach addresses both these points in that (i) equations are added as first class citizens to the ontology language, where variants are considered directly in the semantics, (ii) the presented query rewriting algorithm always terminates and returns finite answers; we also discuss reasonable completeness criteria.

In the remainder of this paper, we first define our ontology language DL^{E}_{RDFS} which extends the RDFS fragment of DL-Lite by simple equations (Sect. 2). In Sect. 3 we define SPARQL queries over DL^{E}_{RDFS} and present our query rewriting algorithm along with a discussion of considerations on soundness and completeness. Alternative implementation approaches with DL reasoners and rules are discussed briefly in Sect. 4, followed by the discussion of a use case experiment in Sect. 5. We wrap up with a discussion of related and future work as well as conclusions (Sects. 6 and 7).

2 Extending Description Logics by Equations

We herein define a simple, restricted form of arithmetic equations and extend a lightweight fragment of DL-Lite by such equations.

Definition 1. *Let $\{x_1, \ldots, x_n\}$ be a set of variables. A* simple equation *E is an algebraic equation of the form $x_1 = f(x_2, \ldots, x_n)$ such that $f(x_2, \ldots, x_n)$ is an arithmetic*

expression over numerical constants and variables $x_2, \ldots, x_n$ where f uses the elementary algebraic operators $+, -, \cdot, \div$ and contains each x_i exactly once. $\mathrm{vars}(E)$ *is the set of variables $\{x_1, \ldots, x_n\}$ appearing in E.*

That is, we allow non-polynomials for f – since divisions are permitted – but do not allow exponents (different from ± 1) for any variable; such equations can be solved uniquely for each x_i by only applying elementary transformations, assuming that all x_j for $j \neq i$ are known: i.e., for each x_i, such that $2 \leq i \leq n$, an equivalent equation E' of the form $x_i = f'(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ is uniquely determined. Note that since each variable occurs only once, the standard procedure for solving single variable equations can be used, we write $\mathrm{solve}(x_1 = f(x_2, \ldots, x_n), x_i)$ to denote E'.[4]

2.1 The Description Logic $\mathbf{DL}^{\boldsymbol{E}}_{\boldsymbol{RDFS}}$

When we talk about Description Logics (DL), we consider a fragment of DL-Lite$_\mathcal{A}$ [18] with basic concepts, existential quantification, attributes over concrete value domains, role/attribute inclusions, and inverse roles which we extend by simple attribute equations. We we call this fragment DL^E_{RDFS}, i.e., it is just expressive enough to capture (the DL fragment of) the RDFS semantics [11] extended with equations. In contrast to DL-Lite$_\mathcal{A}$, DL^E_{RDFS} leaves out role functionality, as well as concept and role negation, and we restrict ourselves to a single value domain for attributes, the set of rational numbers $\mathbb{Q}$.[5]

Definition 2. *Let A be an atomic concept name, P be an atomic role name, and U be an atomic attribute name. As usual, we assume the sets of atomic concept names, role name, and attribute names to be disjoint. Then DL* concept expressions *are defined as* $C ::= A \mid \exists P \mid \exists P^- \mid \exists U$

In the following, let Γ be an infinite set of constant symbols (which, in the context of RDF(S) essentially equates to the set I of IRIs).

Definition 3. *A DL^E_{RDFS} knowledge base (KB) $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ consists of a finite set of terminological axioms $\mathcal{T}$ (TBox) and assertions $\mathcal{A}$ (ABox). For A, P_i, U_i and C denoting atomic concepts, roles, attributes, and concept expressions, resp., $\mathcal{T}$ can contain:*

$C \sqsubseteq A$	(*concept inclusion axiom*)
$P_1 \sqsubseteq P_2$	(*role inclusion axiom*)
$U_1 \sqsubseteq U_2$	(*attribute inclusion axiom*)
$U_0 = f(U_1, \ldots, U_n)$	(*equation axiom*)

A set of role (attribute, resp.) inclusion axioms is called a role hierarchy *(*attribute hierarchy*, resp.). For $a, b \in \Gamma$, and $q \in \mathbb{Q}$, an* ABox *is a set of* concept assertions $C(a)$, role assertions $R(a, b)$, *and* attribute assertions $U(a, q)$. *Finally, by $\Gamma_\mathcal{K}$ (and $\Gamma_A, \Gamma_P, \Gamma_U$, resp.), we denote the (finite) sets of constants from Γ appearing in $\mathcal{K}$ (as concepts, roles, and attributes, resp.).*

[4] in analogy to notation used by computer algebra systems (such as Mathematica or Maxima)

[5] Note that since we only consider this single type of attributes, we also do not introduce value-domain expressions from [18]. Further, instead of $\delta(U)$ in [18] we just write $\exists U$.

Rows 1–6 of Table 1 show the obvious correspondence between DL^{E}_{RDFS} syntax and the essential RDFS terminological vocabulary. As for line 7, we can encode equation axioms in RDF by means of a new property definedByEquation and write the respective arithmetic expressions $f(U_1, \ldots, U_n)$ as plain literals (instead of e.g. breaking down the arithmetic expressions into RDF triples). ABox assertions are covered in rows 8–10, where we note that we use datatype literals of the type owl:rational from OWL2 for rational numbers (which however subsumes datatypes such as xsd:integer, xsd:decimal more commonly used in real world RDF data).

As mentioned before in the context of Definition 1, we consider equations that result from just applying elementary transformations as equivalent. In order to define the semantics of equation axioms accordingly, we will make use of the following definition.

Definition 4. *Let* $E\colon U_0 = f(U_1, \ldots, U_n)$ *be an equation axiom then, for any* U_i *with* $0 \leq i \leq n$ *we call the equation axiom* $\mathrm{solve}(E, U_i)$ *the* U_i-variant *of* E.

Note that the DL defined herein encompasses the basic expressivity of RDFS (subproperty, subclassOf, domain, range)[6] and in fact, rather than talking about a restriction of DL-Lite$_{\mathcal{A}}$, we could also talk about an extension of DL-Lite$_{RDFS}$ [1].[7]

Definition 5 (Interpretation). *An* interpretation $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$ *consists of a non-empty set* $\Delta^{\mathcal{I}}$ *called the object domain, and an interpretation function* $\cdot^{\mathcal{I}}$ *which maps*

- *each atomic concept* A *to a subset of the domain* $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$,
- *each atomic role* P *to a binary relation over the domain* $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$,
- *each attribute* U *to a binary relation over the domain and the rational numbers* $U^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \mathbb{Q}$, *and*
- *each element of* Γ *to an element of* $\Delta^{\mathcal{I}}$.

For concept descriptions the interpretation function is defined as follows:

- $(\exists R)^{\mathcal{I}} = \left\{ x \in \Delta^{\mathcal{I}} \,\middle|\, \exists y.(x, y) \in R^{\mathcal{I}} \right\}$

[6] leaving out subtleties such as e.g. those arising from non-standard use [2] of the RDF vocabulary

[7] DL-Lite$_{RDFS}$ actually also allows to write axioms of the form $P_1 \sqsubseteq P_2^-$ which we do not allow since these in fact are beyond the basic expressivity of RDFS.

Table 1. DL^{E}_{RDFS} axioms in RDFS

	DL^{E}_{RDFS}	RDFS
1	$A_1 \sqsubseteq A_2$	$\mathsf{A_1}$ rdfs:subClassOf $\mathsf{A_2}$
2	$\exists P \sqsubseteq A$	P rdfs:domain A
3	$\exists P^- \sqsubseteq A$	P rdfs:range A
4	$\exists U \sqsubseteq A$	U rdfs:domain A
5	$P_1 \sqsubseteq P_2$	$\mathsf{P_1}$ rdfs:subPropertyOf $\mathsf{P_2}$
6	$U_1 \sqsubseteq U_2$	$\mathsf{U_1}$ rdfs:subPropertyOf $\mathsf{U_2}$
7	$U_0 = f(U_1, \ldots, U_n)$	$\mathsf{U_0}$ definedByEquation "$\mathsf{f(U_1, \ldots, U_n)}$"
8	$A(x)$	x rdf:type A
9	$R(x, y)$	x R y
10	$U(x, q)$	x U "q"^^owl:rational

- $(\exists R^-)^{\mathcal{I}} = \{y \in \Delta^{\mathcal{I}} \mid \exists x.(x,y) \in R^{\mathcal{I}}\}$
- $(\exists U)^{\mathcal{I}} = \{x \in \Delta^{\mathcal{I}} \mid \exists q \in \mathbb{Q}.(x,q) \in U^{\mathcal{I}}\}$

Definition 6 (Model). *An interpretation $\mathcal{I}$ satisfies an axiom of the form*

- $C \sqsubseteq A$ *if* $C^{\mathcal{I}} \subseteq A^{\mathcal{I}}$
- $P_1 \sqsubseteq P_2$ *if* $P_1^{\mathcal{I}} \subseteq P_2^{\mathcal{I}}$
- $U_1 \sqsubseteq U_2$ *if* $U_1^{\mathcal{I}} \subseteq U_2^{\mathcal{I}}$
- $U_0 = f(U_1, \ldots, U_n)$ *if*
  $\forall x, y_1, \ldots, y_n (\bigwedge_{i=1}^{n} (x, y_i) \in U_i^{\mathcal{I}}) \wedge \mathrm{defined}(f(U_1/y_1, \ldots, U_n/y_n)$
  $\Rightarrow (x, \mathrm{eval}(f(U_1/y_1, \ldots, U_n/y_n)) \in U_0^{\mathcal{I}}$

where, by $\mathrm{eval}(f(U_1/y_1, \ldots, U_n/y_n))$ *we denote the actual value in* $\mathbb{Q}$ *from evaluating the arithmetic expression* $f(U_1, \ldots, U_n)$ *after substituting each* U_i *with* y_i*, and by* $\mathrm{defined}(f(U_1/y_1, \ldots, U_n/y_n))$ *we denote that this value is actually defined (i.e., does not contain a division by zero). Analogously, $\mathcal{I}$ satisfies an ABox assertion of the form*

- $C(a)$ *if* $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- $P(a,b)$ *if* $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in P^{\mathcal{I}}$
- $U(a,q)$ *if* $(a^{\mathcal{I}}, q) \in U^{\mathcal{I}}$

Finally, an interpretation $\mathcal{I}$ is called a model *of a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, written $\mathcal{I} \models \mathcal{K}$, if $\mathcal{I}$ satisfies all (role, attribute and concept) inclusion axioms in $\mathcal{T}$, all* variants *of equation axioms in $\mathcal{T}$, and all assertions in $\mathcal{A}$.*

Finally, we define conjunctive queries (with assignments) over DL^{E}_{RDFS}.

Definition 7. *A* conjunctive query *(CQ) is an expression of the form*

$$q(\boldsymbol{x}) \leftarrow \exists \boldsymbol{y}.\phi(\boldsymbol{x}, \boldsymbol{y})$$

where $\boldsymbol{x}$ is a sequence of variables called distinguished variables*, $\boldsymbol{y}$ is a sequence of variables called* non-distinguished variables*, and ϕ is a conjunction of **class**, **role** or **attribute atoms** of the forms $C(x), P(x,y)$, and $U(x,z)$, respectively, and **assignments** of the form $x_0 = f(x_1, \ldots, x_n)$ representing simple equations, where x, y are constant symbols from Γ or variables (distinguished or non-distinguished), and z is either a value from $\mathbb{Q}$ or a variable, and the x_i are variables such that for all $i \geq 1$, x_i appears in an atom of the form $U(x, x_i)$ within ϕ. A set of queries with the same head $q(\boldsymbol{x})$ is a* union of conjunctive queries *(UCQ).*

For an interpretation $\mathcal{I}$, we denote by $q^{\mathcal{I}}$ the set of tuples $\boldsymbol{a}$ of domain elements and elements of $\mathbb{Q}$ which makes ϕ true[8] when $\boldsymbol{a}$ is assigned to distinguished variables $\boldsymbol{x}$ in q.

Definition 8. *For a conjunctive query q and a KB $\mathcal{K}$ the* answer to q over $\mathcal{K}$ *is the set* $\mathrm{ans}(q, \mathcal{K})$ *consisting of tuples $\boldsymbol{a}$ of constants from $\Gamma_{\mathcal{K}} \cup \mathbb{Q}$ such that $\boldsymbol{a}^{\mathcal{M}} \in q^{\mathcal{M}}$ for every model $\mathcal{M}$ of the KB $\mathcal{K}$.*

Note that, as opposed to most DL-Lite variants (such as [3]), $\mathrm{ans}(q, \mathcal{K})$ in our setting is not necessarily finite, as shown by the following example.

Example 2. Let $\mathcal{K}_1 = (\mathcal{T}_1, \mathcal{A}_1)$ with $\mathcal{A}_1 = u_1(o_1, 1), u_2(o_1, 1), u_3(o_1, 1), \mathcal{T}_1 = \{e\colon u_1 = u_2 + u_3\}$ and $q(x) \leftarrow u_1(o_1, x)$ then $\mathrm{ans}(q, \mathcal{K})$ contains any value from $\mathbb{N}$.

[8] We mean true in the sense of first-order logic, where we assume that the interpretation of arithmetic expressions is built-in with the usual semantics for arithmetics over the rational numbers $\mathbb{Q}$, and that equality "=" is false for expressions that yield division by zero on the RHS.

3 SPARQL over $\mathbf{DL}^{E}_{RDFS}$

The semantics of SPARQL is defined as usual based on matching of basic graph patterns (BGPs), more complex patterns are defined as per the usual SPARQL algebra and evaluated on top of basic graph pattern matching, cf. for instance [16, 19]. In order to remain compatible with the notion of CQs in DL^{E}_{RDFS}, we only allow restricted BGPs.[9]

Definition 9. *Let V be an infinite set of variables, I be the set of IRIs, $I_{RDF} = \{\mathsf{rdfs:subClassOf}, \mathsf{rdfs:subPropertyOf}, \mathsf{rdfs:domain}, \mathsf{rdfs:range}, \mathsf{rdf:type}, \mathsf{definedByEquation}\}$, and $I' = I \setminus I_{RDF}$, then* basic graph patterns (BGPs) *are sets of RDF triple patterns (s, p, o) from $((I' \cup V) \times I' \times (I' \cup \mathbb{Q} \cup V)) \cup ((I' \cup V) \times \{\mathsf{rdf:type}\} \times I')$*

More complex graph patterns can be defined recursively on top of basic graph patterns, i.e., if P_1 and P_2 are graph patterns, $v \in V$, $g \in I \cup V$, R is a filter expression, and $Expr$ an arithmetic expression over constants and variables in V, then (i) $\{\{P_1\}\{P_2\}\}$ (conjunction), (ii) $\{P_1\}$ UNION $\{P_2\}$ (disjunction), (iii) P_1 OPTIONAL $\{P_2\}$ (left-outer join), (iv) P_1 FILTER(R) (filter), and (v) P_1 BIND $(Expr$ AS $v)$ (assignment) are graph patterns; as a syntactic restriction we assume that $v \notin \mathrm{vars}(P_1)$. The evaluation semantics of complex patterns builds up on basic graph pattern matching,[10] which we define in our setting simply in terms of conjunctive query answering over the underlying DL.

Following the correspondence of Table 1 and the restrictions we have imposed on BGPs, any BGP P can trivially be mapped to a (non-distinguished-variable-free) conjunctive query of the form $q_P\colon q(\mathrm{vars}(P)) \leftarrow \phi(P)$, where $\mathrm{vars}(P)$ is the set of variables occurring in P.

Example 3. Within the SPARQL query

```
SELECT ?X WHERE { { :o1 :u1 ?X } FILTER ( ?X > 1 ) }
```

the BGP `{ :o1 :u1 ?X }` corresponds to the CQ from Example 2. FILTERs and other complex patterns are evaluated on top of BGP matching:

Definition 10 (Basic graph pattern matching for $\mathbf{DL}^{E}_{RDFS}$). *Let G be an RDF representation of a DL^{E}_{RDFS} KB (cf. Table 1) $\mathcal{K}$. Then, the solutions of a BGP P for G, denoted (analogously to [16]) as $[\![P]\!]_G = \mathrm{ans}(q_P, \mathcal{K})$.*

Note that here we slightly abused notation using $\mathrm{ans}(q_P, \mathcal{K})$ synonymous for what would be more precisely "the set of SPARQL variable mappings corresponding to $\mathrm{ans}(q_P, \mathcal{K})$". As for the semantics of more complex patterns, we refer the reader to [16, 19] for details, except for the semantics of BIND which is newly introduced in SPARQL 1.1 [10], which we define as:

$$[\![P \text{ BIND } (Expr \text{ AS } v)]\!]_G = \{\mu \cup \{v \to \mathrm{eval}(\mu(Expr))\} \mid \mu \in [\![P]\!]_G\}$$

Here, by $\mathrm{eval}(\mu(Expr))$ we denote the actual value in $\mathbb{Q}$ from evaluating the arithmetic expression $Expr$ after applying the substitutions from μ.

[9] We note though, that soundness of our query rewriting approach would not be affected if we allowed arbitrary BGPs.

[10] For simplicity we leave our GRAPH graph patterns or other new features except BIND introduced in SPARQL1.1.

3.1 Adapting PerfectRef to $\mathbf{DL}^{\boldsymbol{E}}_{\boldsymbol{RDFS}}$

Next, we extend the PerfectRef algorithm [3] which reformulates a conjunctive query to directly encode needed TBox assertions in the query. The algorithm $\mathsf{PerfectRef}_\mathsf{E}$ in Algorithm 1 extends the original PerfectRef by equation axioms and conjunctive queries containing assignments, as defined before, following the idea of query rewriting by "expanding" a conjunctive query (CQ) Q to a union of conjunctive queries (UCQ) Q_0 that is translated to a regular SPARQL 1.1 query which is executed over an RDF Store.

$\mathsf{PerfectRef}_\mathsf{E}$ first expands atoms using inclusion axioms (lines 6–8) as in the original PerfectRef algorithm. Here, an DL^{E}_{RDFS} inclusion axiom I is *applicable* to a query atom g if the function gr (Table 2) is defined.[11] The only new thing compared to [3] in Table 2 is the "adornment" $\mathrm{adn}(g)$ of attribute atoms which we explain next, when turning to the expansion of equation axioms.

[11] With DL^{E}_{RDFS} we cover only a very weak DL, but we expect that our extension is applicable to more complex DLs such as the one mentioned in [3], which we leave for future work.

Algorithm 1: Rewriting algorithm $\mathsf{PerfectRef}_\mathsf{E}$

Input: Conjunctive query q, TBox $\mathcal{T}$
Output: Union (set) of conjunctive queries

1 $P := \{q\}$
2 **repeat**
3 $\quad P' := P$
4 $\quad$ **foreach** $q \in P'$ **do**
5 $\quad\quad$ **foreach** *g in q* **do** // expansion
6 $\quad\quad\quad$ **foreach** *inclusion axiom I in $\mathcal{T}$* **do**
7 $\quad\quad\quad\quad$ **if** *I is applicable to g* **then**
8 $\quad\quad\quad\quad\quad P := P \cup \{q[g/\,\mathrm{gr}(g, I)]\}$
9 $\quad\quad\quad$ **foreach** *equation axiom E in $\mathcal{T}$* **do**
10 $\quad\quad\quad\quad$ **if** *$g = U^{\mathrm{adn}(g)}(x, y)$ is an (adorned) attribute atom and* $\mathrm{vars}(E) \cap \mathrm{adn}(g) = \emptyset$ **then**
11 $\quad\quad\quad\quad\quad P := P \cup \{q[g/\,\mathrm{expand}(g, E)]\}$
12 **until** $P' = P$
13 **return** P

Table 2. Semantics of $\mathrm{gr}(g, I)$ of Algorithm 1

g	I	$\mathrm{gr}(g/I)$
$A(x)$	$B \sqsubseteq A$	$B(x)$
	$\exists P \sqsubseteq A$	$P(x, \_)$
	$\exists P^- \sqsubseteq A$	$P(\_, x)$
	$\exists U \sqsubseteq A$	$U(x, \_)$
$P_1(x, y)$	$P_2 \sqsubseteq P_1$	$P_2(x, y)$
$U_1^{\mathrm{adn}(g)}(x, y)$	$U_2 \sqsubseteq U_1$	$U_2^{\mathrm{adn}(g)}(x, y)$

The actually new part of $\mathsf{PerfectRef_E}$ that reformulates attribute atoms in terms of equation axioms is in lines 9–11. In order to avoid infinite expansion of equation axioms during the rewriting, the algorithm "adorns" attribute atoms in a conjunctive query by a set of attribute names. That is, given an attribute atom $U(x, z)$ and a set of attribute names $\{U_1, \ldots, U_k\}$ we call $g = U^{U_1,\ldots,U_k}(x, z)$ an *adorned attribute atom* and write $\mathrm{adn}(g) = \{U_1, \ldots, U_k\}$ to denote the set of adornments. For an unadorned $g = U(x, z)$, obviously $\mathrm{adn}(g) = \emptyset$. Accordingly, we call an *adorned conjunctive query* a CQ where adorned attribute atoms are allowed.

The function $\mathrm{expand}(g, E)$ returns for $g = U^{\mathrm{adn}(g)}(x, y)$ and $E' \colon U = f(U_1, \ldots, U_n)$ being the U-variant of E the following conjunction:

$$U_1^{\mathrm{adn}(g)\cup\{U\}}(x, y_1) \wedge \ldots \wedge U_n^{\mathrm{adn}(g)\cup\{U\}}(x, y_n) \wedge y = f(y_1, \ldots, y_n)$$

where $y_1, \ldots, y_n$ are fresh variables. Here, the condition $\mathrm{vars}(E) \cap \mathrm{adn}(g) = \emptyset$ ensures that U is not "re-used" during expansion to compute its own value recursively. The adornment thus prohibits infinite recursion.

We note that we leave out the *reduction* step of the original PerfectRef algorithm from [3][Fig.2, step (b)], since it does not lead to any additional applicability of inclusion axioms in the restricted Description Logic DL^E_{RDFS}. As we may extend $\mathsf{PerfectRef_E}$ to more expressive DLs as part of future work, this step may need to be re-introduced accordingly.

Finally, just as before we have defined how to translate a SPARQL BGP P to a conjunctive query, we translate the result of $\mathsf{PerfectRef_E}(q_P, \mathcal{T})$ back to SPARQL by means of a recursive translation function $\mathrm{tr}(\mathsf{PerfectRef_E}(q_P, \mathcal{T}))$. That is, for $\mathsf{PerfectRef_E}(q_P, \mathcal{T}) = \{q_1, \ldots q_m\}$ and each q_i being of the form $\bigwedge_{j=0}^{k_i} atom_j$, we define tr as follows:

$\mathrm{tr}(\{q_1, \ldots q_m\})$	{ $\mathrm{tr}(q_1)$ } UNION ... UNION { $\mathrm{tr}(q_m)$ }
$\mathrm{tr}(\bigwedge_j = 0^{k_i} atom_j)$	$\mathrm{tr}(atom_1) \ldots . \mathrm{tr}(atom_{k_i})$
$\mathrm{tr}(A(x))$	$\mathrm{tr}(x)$ rdf : type A
$\mathrm{tr}(P(x, y))$	$\mathrm{tr}(x)$ P $\mathrm{tr}(y)$
$\mathrm{tr}(U(x, y))$	$\mathrm{tr}(x)$ U $\mathrm{tr}(y)$
$\mathrm{tr}(y = f(y_1, \ldots, y_n))$	BIND(f(tr(y_1),...,tr(y_n)) AS $\mathrm{tr}(y)$)
$\mathrm{tr}(x)$, for $x \in V$	$?x$
$\mathrm{tr}(x)$, for $x \in \Gamma$	x
$\mathrm{tr}(x)$, for $x \in \mathbb{Q}$	"x"^^owl:rational

The following proposition follows from the results in [3], since (a) $\mathsf{PerfectRef_E}$ is a restriction of the original PerfectRef algorithm as long as no equation axioms are allowed, and (b) any DL^E_{RDFS} KB is consistent.

Proposition 1. *Let q be a conjunctive query without assignments and $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL^E_{RDFS} KB without equation axioms. Then* $\mathsf{PerfectRef_E}$ *is sound and complete, i.e.*

$$\mathrm{ans}(q, \mathcal{K}) = \mathrm{ans}(\mathsf{PerfectRef_E}(q, \mathcal{T}), \langle \emptyset, \mathcal{A} \rangle)$$

The following corollary follows similarly.

Corollary 1. *Let q be a conjunctive query without assignments and without attribute axioms and let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be an arbitrary DL^E_{RDFS} KB. Then* $\mathsf{PerfectRef_E}$ *is sound and complete.*

As for arbitrary DL^{E}_{RDFS} knowledge bases, let us return to Example 2.

Example 4. Given the knowledge base $\mathcal{K}_1 = \langle \mathcal{T}_1, \mathcal{A}_1 \rangle$ and query q from Example 2. The query $\mathsf{PerfectRef}_{\mathsf{E}}(q, \mathcal{T})$ is

$$\{\, q(x) \leftarrow u_1(o_1, x), q(x) \leftarrow u_2^{u_1}(o_1, x_2), u_3^{u_1}(o_1, x_3), x = x_2 + x_3 \}$$

which only has the certain answers $x = 1$ and $x = 2$, showing that $\mathsf{PerfectRef}_{\mathsf{E}}$ is incomplete in general. As a variant of $\mathcal{K}_1$, lets consider $\mathcal{K}_2 = \langle \mathcal{T}_1, \mathcal{A}_2 \rangle$ with the modified ABox $\mathcal{A}_2 = \{u_1(o_1, 2), u_2(o_1, 1), u_3(o_1, 1)\}$. In this case, notably $\mathsf{PerfectRef}_{\mathsf{E}}$ delivers complete results for $\mathcal{K}_2$, i.e., $\mathrm{ans}(q, \mathcal{K}_2) = \mathrm{ans}(\mathsf{PerfectRef}_{\mathsf{E}}(q, \mathcal{T}_1), \langle \emptyset, \mathcal{A}_2 \rangle)$ with the single certain answer $x = 2$. Finally, the rewritten version of the SPARQL query in Example 3 is

```
SELECT ?X WHERE {
    { { :o1 :u1 ?X } UNION
      { :o1 :u2 ?X2 . :o1 :u3 ?X3 . BIND(?X2+?X3 AS ?X ) } }
    FILTER ( ?X > 1 )  }
```

In order to capture a class of DL^{E}_{RDFS} KBs, where completeness can be retained, we will use the following definition.

Definition 11. *An ABox $\mathcal{A}$ is* data-coherent *with $\mathcal{T}$, if there is no pair of ground atoms $U(x, d'), U(x, d)$ with $d \neq d'$ entailed by $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$*

The following result is obvious.

Lemma 1. *Whenever $\mathcal{A}$ is* data-coherent *with $\mathcal{T}$, any conjunctive query has a finite number of certain answers.*

Proof (Sketch). Assume that the certain answers to q are infinite. From Corollary 1 we can conclude that infiniteness can only stem from distinguished variables that occur as attribute value y in some attribute atom $U(x, y)$ in the query. However, that would in turn mean that there is at least one x with an infinite set of findings for y, which contradicts the assumption of data-coherence.

The following stronger result (which for our particular use case of BGP matching in SPARQL we only consider for non-distinguished-variable-free conjunctive queries) states that data-coherence in fact implies completeness.

Theorem 1. *If $\mathcal{A}$ is* data-coherent *with $\mathcal{T}$, the for any non-distinguished-variable-free conjunctive query q $\mathsf{PerfectRef}_{\mathsf{E}}$ is sound and complete.*

Proof (Sketch). The idea here is that whenever $\mathcal{A}$ is *data-coherent* with $\mathcal{T}$ for any fixed x any certain value y for $U(x, y)$ will be returned by $\mathsf{PerfectRef}_{\mathsf{E}}$: assuming the contrary, following a shortest derivation chain $U(x, y)$ can be either (i) be derived by only atoms $U_i(x, y_i)$ such that any U_i is different from U, in which case this chain would have been "expanded" by $\mathsf{PerfectRef}_{\mathsf{E}}$, or (ii) by a derivation chain that involves an instance of $U(x, z)$. Assuming now that $z \neq y$ would violate the assumption of data-coherence, whereas if $z = y$ then $U(x, y)$ was already proven by a shorter derivation chain.

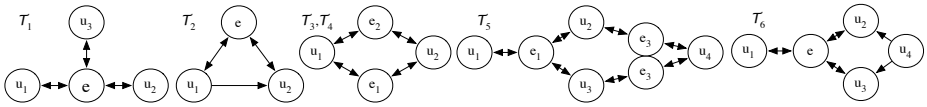
In what follows, we will define a fragment of DL^{E}_{RDFS} KBs where data-coherence can be checked efficiently. First, we note that a data-coherent ABox alone, such as for instance in $\mathcal{K}_2$ in Example 4 above, is in general not a guarantee for data-coherence. To show this, let us consider the following additional example.

Example 5. Consider the TBox $\mathcal{T}_2 = \{e\colon u_1 = u_2 + 1, u_2 \sqsubseteq u_1\}$ As easily can be seen, any ABox containing an attribute assertion for either u_1 or u_2 is data-incoherent with $\mathcal{T}_2$.

The example also shows that considering equation axioms only is not sufficient to decide data-coherence, but we also need to consider attribute inclusion axioms. Following this intuition, we define a dependency graph over $\mathcal{T}$ as follows.

Definition 12. *A TBox dependency graph is $G_{\mathcal{T}} = \langle N, E\rangle$ is constructed from nodes for each attribute and each equation axiom $N = \{e \mid e \text{ is an equation axiom in } \mathcal{T}\} \cup \Gamma_U$. There exist edges (e, v) and (v, e) between every equation e and its variables $v \in \mathrm{vars}(e)$. Furthermore there exists an edge (u, v) for each attribute inclusion axiom $u \sqsubseteq v$. If G contains no (simple) cycle with length greater than 2, then we call $\mathcal{T}$* attribute-acyclic.

Example 6. Given $\mathcal{T}_1, \mathcal{T}_2$ from Examples 2 and 5, let further $\mathcal{T}_3 = \{e_1\colon u_1 = u_2 + 1, e_2\colon u_2 = u_1 + 1\}$, $\mathcal{T}_4 = \{e_1\colon u_1 = u_2 + 1, e_2\colon u_2 = u_1 - 1\}$, and $\mathcal{T}_5 = \{e_1\colon u_1 = u_2 - u_3, e_2\colon u_4 = u_2\, e_3\colon u_4 = u_3\}$ $\mathcal{T}_6 = \{e\colon u_1 = u_2 - u_3, u_4 \sqsubseteq u_2\, u_4 \sqsubseteq u_3\}$ then the resp. dependency graphs are as follows where the graphs for $\mathcal{T}_2$–$\mathcal{T}_5$ are cyclic.



Notably, since e_2 is a variant of e_1 in $\mathcal{T}_4$, $\mathcal{T}_4$ is actually equivalent to an acyclic TBox (removing either e_1 or e_2), whereas this is not the case for $\mathcal{T}_3$; more refined notions of acyclicity, which we leave for future work, might capture this difference. Therefore, as shown in Examples 2 and 4 for $\mathcal{T}_1, \mathcal{T}_2$. Further, let us point out the subtle difference between $\mathcal{T}_5$ and $\mathcal{T}_6$. In $\mathcal{T}_5$, when e_1–e_3 are viewed as equation system, partially solving this system would result in the new equation $u_1 = 0$, independent of the ABox. Since $\mathsf{PerfectRef_E}$ does not solve any equation systems (but only instantiates equations with values from the ABox), it would not detect this. On the contrary, in $\mathcal{T}_6$, only when a concrete "witness" for u_4 is available in the ABox, this constrains the value of u_1 to be 0, which could be correctly detected by means of $\mathsf{PerfectRef_E}$: for attribute-acyclic TBoxes, data-coherence indeed (finitely) depends on the ABox and we can define a procedure to check data-coherence (and thus completeness) by means of $\mathsf{PerfectRef_E}$ itself.

Proposition 2. *Let $\mathcal{T}$ be an attribute-acyclic TBox, and $\Gamma_U = \{u_1, \ldots, u_m\}$. The following SPARQL query $Q^{\mathcal{T}}_{check}$*

$$\begin{aligned}&\mathsf{ASK}\ \{\ \{\ \mathrm{tr}(\mathsf{PerfectRef_E}(q_{P_1}, \mathcal{T}))\ \mathsf{FILTER}(\ ?Y_1\ !{=}\ ?Z_1)\ \}\\ &\qquad \mathsf{UNION}\ldots\ \mathsf{UNION}\\ &\qquad \{\ \mathrm{tr}(\mathsf{PerfectRef_E}(q_{P_m}, \mathcal{T}))\ \mathsf{FILTER}(\ ?Y_1\ !{=}\ ?Z_1)\ \}\ \}\end{aligned}$$

where $P_i = \{\ ?X\ u_i\ ?Y_1\ .\ ?X\ u_i\ ?Z_2\ \}$ determines data-coherence in the following sense: an ABox $\mathcal{A}$ is data-coherent with $\mathcal{T}$ if Q returns "no".

The idea here is that since $\mathcal{T}$ is attribute-acyclic, and due to the restriction that variable occurs at most once in simple equations, finite witnesses for data-incoherences can be acyclically derived from the ABox, and thus would be revealed by $\mathsf{PerfectRef}_{\mathsf{E}}$.

4 Discussion of Alternative Implementation Approaches

Our approach relies on standard SPARQL1.1 queries and runs on top of any off-the-shelf SPARQL1.1 implementation by first extracting the TBox and then rewriting BGPs in each query according to the method described in the previous section. In order to compare this rewriting to alternative approaches, we have looked into DL reasoners as well as rule-based reasoners, namely, Racer, Pellet, and Jena Rules. We discuss the feasibility of using either of these for query answering under DL^{E}_{RDFS} separately.

Racer [8] provides no SPARQL interface but uses its own functional query language *new Racer Query Language* (nRQL). The system allows for modeling some forms of equation axioms, cf. examples modeling unit conversions in [9], but Racer only uses these for satisfiability testing and not for query answering (which is orthogonal to our approach, as due to the lack of negation there is no inconsistency in DL^{E}_{RDFS}).

SWRL [12, 13] implementations like Pellet [26] allow to handle DL-safe rules [15], that is, rules where each variable appears in at least one non-DL-Atom. We discussed potential modeling of equation axioms as SWRL rules already in Example 1: as mentioned there, rules for each variant of each equation axiom must be added to enable query answering for DL^{E}_{RDFS}. Taking this approach, experiments with Pellet showed that queries over certain data-coherent ABoxes were answered correctly (despite – to our reading – rules like (1)+(2) are not DL-safe in the strict sense), but we still experienced termination problems for e.g. the data and query mentioned in Example 1, since strictly speaking, the data for `:Vienna` is not data-coherent (due to rounding errors). Due to the finite nature of our rewriting, our approach always terminates and is thus robust even for such – strictly speaking – incoherent data. Sect. 5 will give more details.

Jena[12] provides rule-based inference on top of TDB in a proprietary rule language with built-ins, with SPARQL querying on top. Similar to SWRL, we can encode all variants of equation axioms. Jena allows to execute rules in backward and forward mode, where backward execution does not terminate due to its recursive nature (including empty ABoxes). Forward execution suffers from similar non-termination problems as mentioned above for incoherent data as in Example 1, whereas forward execution for data-coherent ABoxes terminates. Jena offers a hybrid rule based reasoning where pure RDFS inferencing is executed in a backward-chaining manner, but still can be combined with forward rules; this approach was incomplete in our experiments, because property inclusion axioms did not "trigger" the forward rules modeling equation axioms correctly.

5 A Practical Use Case and Experiments

For a prototypical application to compare and compute base indicators of cities – as its needed for studies like Siemens' Green City Index[13] – we collected open data about cities

[12] `http://jena.apache.org/documentation/inference/index.html`
[13] `http://www.siemens.com/entry/cc/en/greencityindex.htm`

from several sources (DBPedia, Eurostat, ...) from several years. When aggregating these sources into a joint RDF dataset, different kinds of problems such as incoherences, incomplete data, incomparable units along the lines of the extract in Example 1 occurred. Most indicators (such as demography, economy, or climate data) comprise numeric values, where functional dependencies modeled as equation axioms are exploitable to arrive at more complete data from the sparse raw values.

For an initial experiment to test the feasibility of the query answering approach presented in this paper, we assembled a dataset containing ABox 254,081 triples for a total of 3162 city contexts (i.e., when we speak of a "city" sloppily, we actually mean one particular city in a particular year) along with the following (attribute-acyclic) TBox:

$$\text{e1}: \mathsf{tempHighC} = (:\mathsf{tempHighF} - 32) \cdot 5 \div 9$$
$$\text{e2}: \mathsf{populationRateMale} = :\mathsf{populationMale} \div :\mathsf{population}$$
$$\text{e3}: \mathsf{populationRateFemale} = :\mathsf{populationFemale} \div :\mathsf{population}$$
$$\text{e4}: \mathsf{area\_km2} = :\mathsf{area\_m2} \div 1000000$$
$$\text{e5}: \mathsf{area\_km2} = :\mathsf{area\_mile2} \div 2.589988110336$$
$$\text{e6}: \mathsf{populationDensity} = :\mathsf{population} \div :\mathsf{area\_km2}$$

$$:\mathsf{City} \sqsubseteq :\mathsf{Location} \quad \mathsf{foaf}:\mathsf{name} \sqsubseteq \mathsf{rdfs}:\mathsf{label} \quad \mathsf{dbpedia}:\mathsf{name} \sqsubseteq \mathsf{rdfs}:\mathsf{label}$$

We use the following queries for our experiments:

Q1. Return the population density of all cities:

```
SELECT ?C ?P
WHERE { ?C rdf:type :City . ?C :populationDensity ?P . }
```

Q2. Select cities with a maximum annual temperature above 90°F.

```
SELECT ?C
WHERE { ?C rdf:type :City . ?C rdfs:label ?L .
        ?C :tempHighF ?P . FILTER(?F > 90) }
```

Q3. Select locations with a label that starts with "W" and a population over 1 million:

```
SELECT ?C
WHERE { ?C rdf:type :Location . ?C rdfs:label ?L .
        ?C :population ?P .
        FILTER(?P > 1000000 && STRSTARTS(?L,"W")) }
```

Q4. Select places with a higher female than male population rate.

```
SELECT ?C
WHERE { ?C :populationRateFemale ?F .
        ?C :populationRateMale ?M . FILTER( ?F > ?M ) }
```

Experimental results are summarized in Table 3. For the reasons given in Sect. 4, we compare our approach only to Jena Rules. Experiments were run on the dataset using Jena and ARQ 2.9.2 (without a persistent RDF Store). For Jena Rules, first we encoded the essential RDFS rules plus all variants of equation axioms in a straightforward manner as forward rules, leading to the expected non-termination problems with incoherent data. To avoid this, we created a coherent sample of our dataset (253,114 triples) by removing triples leading to possible incoherences, however still reaching a timeout of 10min for all 4 queries. As an alternative approach, we used Jena's negation-as-failure built-in `noValue` which returns sound but incomplete results, in that it fires a rule only if no value exists for a certain attribute (on the inferences so far or in the data);

similar to our approach, this returns complete results for data-coherent datasets and always terminates. As an example of encoding the variants of an axiom in Jena Rules, we show the encoding of equation e6 (which is identical to the naive encoding except the `noValue` predicates). Possible divisions by 0, which we do not need to care about in our SPARQL rewriting, since BIND just filters them out as errors, are caught by `notEqual(`$Quotient$`,0)` predicates.

```
[  (?city :area ?ar) (?city :population ?p)
   notEqual(?ar, 0) quotient(?p, ?ar, ?pd)
   noValue(?city, :populationDensity)
   -> (?city :populationDensity ?d)]
[  (?city :area ?ar) (?city :populationDensity ?pd)
   product(?ar, ?pd, ?p) noValue(?city, :population)
   -> (?city :population ?p)]
[  (?city :populationDensity ?pd) (?city :population ?p)
   notEqual(?pd, 0) quotient(?p, ?pd, ?ar) noValue(?city, :area)
   -> (?city :area ?ar)]
```

Overall, while this experiment was mainly meant as a feasibility study of our query-rewriting approach, the results as shown in Table 3 are promising: we clearly outperform the only rule-based approach we could compare to. However, looking further into alternative implementation strategies and optimizations remains on our agenda.

As a final remark, we observed during our experiments that single Web sources tend to be coherent in the values they report for a single city, thus data-incoherences, i.e. ambiguous results in our queries for one city typically stem from the combination of different sources considered for computing values through equations. As a part of future work, we aim to further investigate this, building up on our earlier results for combining inferences in SPARQL with conveying provenance information in the results, cf. [27].

6 Further Related Work and Possible Future Directions

OWL ontologies for measurements and units such as QUDT [20], OM [21] provide means to describe units and – to a certain extent – model conversion between these units, though without the concrete machinery to execute these conversions in terms of arbitrary SPARQL queries. Our approach is orthogonal to these efforts in that (a) it provides not only a modeling tool for unit conversions, but integrates attribute equations as axioms in the ontology language, and (b) allows for a wider range of use cases, beyond conversions

Table 3. Query response times in seconds

	Coherent Sample of our Dataset			Full Dataset		
#	Our System	Jena naive	Jena noValue	Our System	Jena naive	Jena noValue
Q1	6.5	>600	30.7	7.3	–	30.1
Q2	5.8	>600	32.7	5.7	–	31.3
Q3	7.8	>600	32.5	8.2	–	29.0
Q4	6.9	>600	34.3	7.9	–	32.4

between pairs of units only. It would be interesting to investigate whether ontologies like QUDT and OM can be mapped to the framework of DL^{E}_{RDFS} or extensions thereof.

Moreover, in the realm of DL-Lite query rewriting, following the PerfectRef algorithm [3] which we base on, there have been a number of extensions and alternative query rewriting techniques proposed [7, 14, 17, 22, 23] which could likewise serve as a basis for extensions by attribute equations. Another obvious direction for further research is the extension to more expressive ontology languages than DL^{E}_{RDFS}. Whereas we have deliberately kept expressivity to a minimum in this paper, apart from further DL-Lite fragments we are particularly also interested in lightweight extensions of RDFS such as OWL LD [6] which we aim to consider for future work.

Apart from query answering, this work opens up research in other reasoning tasks such as query containment of SPARQL queries over DL^{E}_{RDFS}. While containment and equivalence in SPARQL are a topic of active research [4,16,25] we note that containment could in our setting depends not only on the BGPs, but also on FILTERs. E.g., intuitively query Q4 in our setting would be equivalent (assuming `:population` > 0) to

```
SELECT ?C WHERE { ?C :populationFemale ?F .
      ?C :populationMale ?M . FILTER( ?F > ?M ) }
```

While we leave closer investigation for future work, we note another possible connection to related work [24] on efficient query answering under FILTER expression also based in constraint-based techniques.

Lastly, we would like to point out that our approach could be viewed as rather related to Constraint-handling-rules [5] than to mainstream semantic Web rules approaches such as SWRL, etc.; we aim to further look into this.

7 Conclusions

We have presented a novel approach to model mathematical equations as axioms in an ontology, along with a practical algorithm for query answering using SPARQL over such enriched ontologies. To the best of our knowledge, this is the first framework that combines ontological reasoning in RDFS, inferencing about functional dependencies among attributes formulated as generic equations, and query answering for SPARQL. Experimental results compared to rule-based reasoning are encouraging. Given the increasing amount of published numerical data in RDF on the emerging Web of data, we strongly believe that this topic deserves increased attention within the Semantic Web reasoning community.

Acknowledgements. Stefan Bischof has been partially funded by the Vienna Science and Technology Fund (WWTF) through project ICT12-015.

References

1. Arenas, M., Botoeva, E., Calvanese, D., Ryzhikov, V., Sherkhonov, E.: Representability in DL-Lite$_R$ knowledge base exchange. In: 25th Int'l DL Workshop, vol. 846, pp. 4–14 (2012)
2. de Bruijn, J., Heymans, S.: Logical foundations of (e)RDF(S): Complexity and reasoning. In: Aberer, K., et al. (eds.) 6th ISWC. LNCS, vol. 4825, pp. 86–99. Springer (2007)

3. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The DL-Lite family. Journal of Automated Reasoning 39(3), 385–429 (2007)
4. Chekol, M.W., Euzenat, J., Genevès, P., Layaïda, N.: Sparql query containment under shi axioms. In: 26th AAAI Conf. (2012)
5. Frühwirth, T.W.: Constraint handling rules: the story so far. In: 8th PPDP, pp. 13–14 (2006)
6. Glimm, B., Hogan, A., Krötzsch, M., Polleres, A.: OWL: Yet to arrive on the web of data? In: WWW2012 Workshop on Linked Data on the Web (2012)
7. Gottlob, G., Schwentick, T.: Rewriting ontological queries into small nonrecursive datalog programs. In: 13th Int'l KR Conf. (2012)
8. Haarslev, V., Moeller, R.: Racer system description. In: Goré, R., et al. (eds.) Automated Reasoning (IJCAR). LNCS, vol. 2083, pp. 701–705. Springer (2001)
9. Haarslev, V., Möller, R.: Description logic systems with concrete domains: Applications for the semantic web. In: 10th Int'l KRDB Workshop (2003)
10. Harris, S., Seaborne, A.: SPARQL 1.1 query language. W3C proposed rec., W3C (2012)
11. Hayes, P.: RDF semantics. W3C rec., W3C (2004),
12. Horrocks, I., Patel-Schneider, P.F.: A proposal for an owl rules language. In: 13th Int'l Conf. on World Wide Web (WWW2004), pp. 723–731. ACM (2004)
13. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosof, B., Dean, M.: SWRL: A semantic web rule language combining OWL and RuleML. W3C member subm., W3C (2004)
14. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to ontology-based data access. In: 22nd IJCAI, pp. 2656–2661. (2011)
15. Motik, B., Sattler, U., Studer, R.: Query answering for OWL-DL with rules. Journal of Web Semantics (JWS) 3(1), 41–60 (2005)
16. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. ACM Transactions on Database Systems 34(3) (2009)
17. Pérez-Urbina, H., Motik, B., Horrocks, I.: Tractable query answering and rewriting under description logic constraints. Journal of Applied Logic 8(2), 186–209 (2010)
18. Poggi, A., Lembo, D., Calvanese, D., Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. In: Spaccapietra, S. (ed.) JODS X, LNCS, vol. 4900, pp. 133–173. Springer (2008)
19. Prud'hommeaux, E., Seaborne (eds.), A.: SPARQL Query Language for RDF. W3C rec., W3C (2008)
20. Ralph Hodgson, P.J.K.: Qudt - quantities, units, dimensions and data types in owl and xml (2011), `http://www.qudt.org/`
21. Rijgersberg, H., van Assem, M., Top, J.: Ontology of units of measure and related concepts. Semantic Web Journal (SWJ), accepted (2012)
22. Rosati, R.: Prexto: Query rewriting under extensional constraints in dl - lite. In: Simperl, E., et al. (eds.) 9th ESWC, LNCS, vol. 7295, pp. 360–374. Springer (2012)
23. Rosati, R., Almatelli, A.: Improving query answering over dl-lite ontologies. In: 12th Int'l KR Conf. (2010)
24. le Clément de Saint-Marcq, V., Deville, Y., Solnon, C., Champin, P.A.: Castor: A constraint-based sparql engine with active filter processing. In: Simperl, E., et al. (eds.) 9th ESWC, LNCS, vol. 7295, pp. 391–405. Springer (2012)
25. Schmidt, M., Meier, M., Lausen, G.: Foundations of sparql query optimization. In: ICDT (2010)
26. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Journal of Web Semantics (JWS) 5(2), 51–53 (2007)
27. Zimmermann, A., Lopes, N., Polleres, A., Straccia, U.: A general framework for representing, reasoning and querying with annotated semantic web data. Journal of Web Semantics (JWS) 12, 72–95 (2012)