Journal of Computer and System Sciences 77 (2011) 520–541

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Foundations of Semantic Web databases

Claudio Gutierrez[a,*], Carlos A. Hurtado[b], Alberto O. Mendelzon[†], Jorge Pérez[c]

[a] *Department of Computer Science, Universidad de Chile, Chile*
[b] *Faculty of Engineering and Sciences, Universidad Adolfo Ibáñez, Chile*
[c] *Department of Computer Science, Pontificia Universidad Católica de Chile, Chile*

ARTICLE INFO

Article history:
Received 7 October 2008
Received in revised form 17 September 2009
Available online 24 April 2010

Keywords:
Semantic Web
RDF model
Query language

ABSTRACT

The Semantic Web is based on the idea of a common and minimal language to enable large quantities of existing data to be analyzed and processed. This triggers the need to develop the database foundations of this basic language, which is the *Resource Description Framework* (*RDF*). This paper addresses this challenge by: 1) developing an abstract model and query language suitable to formalize and prove properties about the RDF data and query language; 2) studying the RDF data model, minimal and maximal representations, as well as normal forms; 3) studying systematically the complexity of entailment in the model, and proving complexity bounds for the main problems; 4) studying the notions of query answering and containment arising in the RDF data model; and 5) proving complexity bounds for query answering and query containment.

1. Introduction

The Semantic Web is a proposal to build an infrastructure of machine-readable semantics for the data on the Web. In 1999 the W3C issued a recommendation of a metadata model and language to serve as the basis for such infrastructure, the *Resource Description Framework* (*RDF*) [44]. As time passed, RDF evolved and increasingly gained attraction from both researchers and practitioners as a data model apt to represent the first layer of semantics on the Web [50].

RDF follows the W3C design principles of interoperability, extensibility, evolution and decentralization. Particularly, the RDF model was designed with the following goals: simple data model; formal semantics and provable inference; extensible URI-based vocabulary; allowing anyone to make statements about any resource. In the RDF model, the universe to be modeled is a set of *resources*, essentially anything that can have a *uniform resource identifier*, URI. The language to describe them is a set of *properties*, technically binary predicates. Descriptions are *statements* very much in the subject–predicate–object structure, where predicate and object are resources or strings. Both subject and object can be anonymous objects, known as *blank nodes*. In addition, the RDF specification includes a built-in vocabulary with a normative semantics (RDFS). This vocabulary deals with inheritance of classes and properties, as well as typing, among other features [46]. Good introductory references for the RDF model are [47] and [48]. Fig. 1 shows a simple example of RDF data. Simultaneously to the release of data model, the natural problem of querying RDF was raised. In fact, several languages for querying RDF were developed in parallel with RDF itself (see the studies [12] and [11] for detailed comparisons of RDF query languages). In 2008 the RDF Data Access Working Group (part of the Semantic Web Activity) released the standard of a query language for RDF, called SPARQL [49] which address the basic needs of querying RDF, leaving several issues open for the future: inclusion of RDFS vocabulary, paths, nesting, premises, etc.

[*] Corresponding author.
E-mail addresses: cgutierr@dcc.uchile.cl (C. Gutierrez), carlos.hurtado@uai.cl (C.A. Hurtado), jperez@ing.puc.cl (J. Pérez).
[†] Alberto O. Mendelzon passed away June 16, 2005.

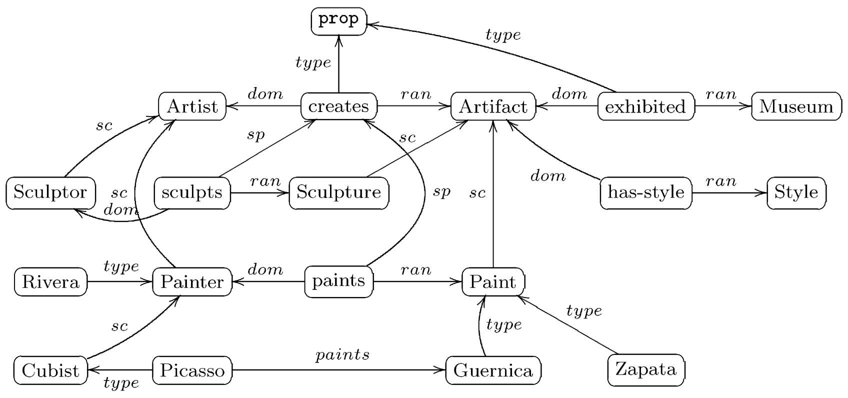

Fig. 1. An RDF graph specifying a schema to describe art resources. The relations *subclass* (*sc*), *subproperty* (*sp*), *type*, *domain* and *range* belong to the RDFS vocabulary. The triple (*Picasso, paints, Guernica*) shows that in RDF specifications, schemas and data can be described at the same level. Note that the set of arc labels and node labels may intersect, e.g. *paints* is both a node label and an arc label. There are arcs not depicted to avoid crowding the figure. Example taken from [6].

All these developments have triggered the need of a more systematic research on formal aspects of the RDF database model, that is, its data model and query language. Among the first formal studies of the characteristic of the RDF data model was the paper *Foundations of Semantic Web databases*, presented at the PODS conference in 2004 [22]. That paper presented an integrated analysis of fundamental database problems in the realm of RDF, including normal forms and redundancy elimination, minimal representation for data exchange, semantics of query languages, query containment and complexity of query processing.

The RDF data model allows several representations for the same information, which raises the question about the existence of normal forms and testing of equivalence among them. On the same lines, query language features deserve a systematic and integrated study. Traditional database notions of query containment do not translate directly to the RDF setting. They need to be reformulated to take into account the fact that RDF queries process logical specifications rather than plain data. Additionally, if one adds premises and constraints on queries, further complexity to the problem is added. Regarding query processing, the presence of predefined semantics and blank nodes in RDF introduce new problems. These include testing entailment of databases and query conditions for keeping RDF databases and query outputs as concise as possible.

The PODS 2004 paper introduced a simple and abstract version of RDF which captured the core aspects of the language, as well as a query language in a streamlined form to have a basic core to focus on the central aspects of the aforementioned problems. The abstract model was not intended for practical use, but designed to be simple enough to make it easy to formalize and prove results about its properties. The query language design addressed the basic features that arise in querying RDF graphs as opposed to standard databases: the presence of blank nodes, premises in queries, and the role played in this scenario by RDFS vocabulary with predefined semantics.

This paper is an extended, modified and updated version of [22]. Besides including the formal proofs that for space reasons were all absent in that conference version, in this paper we have extended several results on minimal representations and query containment. In the meantime, since the publication of [22], several results presented in the paper has been developed and improved by the RDF community. Of particular interest in this direction was the introduction of an abstract fragment to study RDF [31], which corrected the fragment presented in [22]. Thus, this paper incorporates these new improvements when necessary to our discussion, and points to other relevant developments of the area. We expect this paper to serve as a basic, self-contained and updated reference regarding the formal study of the RDF model on the lines of [22].

Related work. The RDF model was introduced in 1999 as a W3C recommendation [44]. In 2004 a standard semantics for the data model [45] was issued by the W3C. The first formal analysis on RDF from a database point of view was presented by Gutierrez el at. [22], where complexity bounds on entailment and computing cores were given. The RDF specification has been also object of several analysis in the W3C Committees and in the academic world. The studies of Marin [29] and ter Horst [26] formalized the notation and corrected minor problems. Ter Horst [26] also proved completeness and complexity results for an extension to some vocabulary of OWL. From a logical point of view, Yang and Kifer in [41] present an F-logic version of RDF. They define two notions of entailment for RDF graphs and concentrate mainly in the semantics of blank nodes and reification. De Bruijn et al. [8,9] present a logical analysis of the theory of RDF in a classical first order logic setting. In other direction, Muñoz et al. [31] study fragments of RDF and systematize the core fragment which was introduced in [22]. Extensions of the model, adding expressiveness leading to the realm of descriptive logics, can be found in the *Web Ontology Language*, OWL [37]. This line of development has rich developments which we will not survey here.

Languages for querying RDF have been developed in parallel with RDF itself. We can mention rdfDB [19], an influential simple graph-matching query language from which several other query languages evolved. Among them, SquishQL [30] is a graph-navigation query language that was designed to test some of the functionalities of an RDF query language. It adds constraints on the variables and returns a table as result. SquishQL has several implementations like RDQL and Inkling [30]. RQL [27] has a very different syntax based on OQL, but can perform similar sorts of queries. It is a typed language following a functional approach and supports generalized path expressions. Its new version is [6]. Other languages are Triple [39], a query and transformation language, QEL [32], a query-exchange language designed to work across heterogeneous repositories, and DQL [43], a language for querying DAML+OIL knowledge bases, that consider RDF data as a knowledge base, applying reasoning techniques to RDF querying. Good surveys are [38,28], and more recent ones [12,11]. Recent developments in RDF query languages include studies of the W3C standard SPARQL [49], its formal semantics and complexity [34] and expressive power [4], as well as extensions in different directions [3,14,35].

There are several ideas developed in the database community that are of interest to RDF. Ideas from the processing of semistructured data are of use in the RDF context, e.g. incomplete answers [13], and query rewriting [33]. Despite the apparent similarities of the models, aspects like blank nodes, graph-like structure, and semantics, make the problems studied in this paper somehow orthogonal to problems addressed in previous research on semistructured data. The notion of core has appeared in various contexts, e.g. graphs [25], data exchange [15,20,21]. Queries with premises (see Section 4.2) have been studied in the logic programming community, e.g. [16]. Their complexity aspects from a database point of view are studied in [7]. Premises also appear in the context of query languages for knowledge bases, e.g. DQL [43]. In SQL-like RDF query languages, this feature appears as a specification of a schema to be used when processing the query [6,30].

The paper is organized as follows. Section 2 gives the abstract formalization of RDF, including the semantics, a deductive system, and studies the complexity of entailment. Section 3 studies normal forms for RDF data. This section studies maximal representations (notions of closure), minimal representations (including notions such as core), and normal forms for RDF data. Section 4 studies RDF query languages. First we study the notion of answer in this context and then study query with premises. Section 5 deals with query containment. In this section we show that in the RDF context there are diverse notions of containment. Then we present results about query containment for queries with and without premises. In Section 6 we study the complexity of query answering for the models presented. Finally we present brief conclusions. To easy cross-referencing to readers, we enumerated theorems, propositions, corollaries, definitions, notes, etc. with a unique sequential numbering.

2. Formalization of abstract RDF

The RDF model is specified in a series of W3C documents [44–46,48]. In this section we introduce an abstract version of the RDF data model, which is both, a fragment following faithfully the original specification, and an abstract version more suitable to do formal analysis. What is left out are features of RDF directed to the implementations, such as detailed typing issues, some distinguish vocabulary which has no particular semantics, and all topics involved with the XML-based syntax and serialization. The original formulation of this fragment was introduced in [22] and enriched and corrected in [31], and we present it here to make this paper self-contained. The details can be found in [31].

The main objective of isolating and working over such a fragment is to have a simple and stable core over which to discuss theoretical issues dealing with RDF from a database point of view.

2.1. RDF graphs

Assume there is an infinite set U (RDF URI references) and an infinite set $B = \{N_j\colon\ j \in \mathbb{N}\}$ (blank nodes). A triple $(s, p, o) \in (U \cup B) \times U \times (U \cup B)$ is called an *RDF triple*.[1] In such a triple, s is called the *subject*, p the *predicate* and o the *object*. We often denote by *UB* the union of the sets U and B.

Definition 2.1. An *RDF graph* (just graph from now on) is a set of RDF triples. A *subgraph* is a subset of a graph. The *universe* of a graph G, universe(G), is the set of elements of *UB* that occur in the triples of G. The *vocabulary* of G, voc(G), is the set universe$(G) \cap U$. A graph is *ground* if it has no blank nodes. In general we will use uppercase letters $N, X, Y, \ldots$ to denote blank nodes, the initial lowercase letters $a, b, c, \ldots$ (and p) for URIs and final lowercase letters $v, w, x, \ldots$ (and s, o) to denote general elements in *UB*.

Graphically we represent RDF graphs as follows: each triple (s, p, o) is represented by $s \xrightarrow{p} o$. Note that the set of arc labels can have non-empty intersection with the set of node labels. Note that technically speaking, and "RDF graph" is not a graph in classical graph theoretic terms. The problem is that the set of nodes and arcs are not empty. For a further discussion of this issue see [24].

[1] Note that we are not considering *literals* as independent objects in this model. In [22] literals were considered, but they played no role in the development at that abstraction level. Thus, to simplify the model we simply disregarded them here. Nevertheless our results can be easily extended to consider *plain literals* [45]. When considering *typed literals* [45] several issues arise. For example, *XML typed literals* can be used to express *contradictory assertions* [45]. In this paper we do not consider typed literals in order to concentrate in the fundamental components of RDF.

We will need some technical definitions. A *map* is a function $\mu : UB \to UB$ preserving URIs, i.e., $\mu(u) = u$ for all $u \in U$. Given a graph G, we define $\mu(G)$ as the set of all $(\mu(s), \mu(p), \mu(o))$ such that $(s, p, o) \in G$. We say that the graph $\mu(G)$ is an *instance* of the graph G. An instance of G is *proper* if $\mu(G)$ has fewer blank nodes than G. This means that either μ sends a blank node to a URI, or identifies two blank nodes of G. We will overload the meaning of map and speak of a *map* $\mu : G_1 \to G_2$ if there is a map μ such that $\mu(G_1)$ is a subgraph of G_2.

Two RDF graphs G_1, G_2 are *isomorphic*, denoted $G_1 \cong G_2$, if there are maps μ_1, μ_2 such that $\mu_1(G_1) = G_2$ and $\mu_2(G_2) = G_1$.

We define two operations on graphs. The *union* of G_1, G_2, denoted $G_1 \cup G_2$, is the set theoretical union of their sets of triples. The *merge* of G_1, G_2, denoted $G_1 + G_2$, is the union $G_1 \cup G_2'$, where G_2' is an isomorphic copy of G_2 whose set of blank nodes is disjoint with that of G_1. Note that $G_1 + G_2$ is unique up to isomorphism.

2.2. RDFS vocabulary

The RDF specification includes a set of reserved words, the RDFS vocabulary (RDF schema [46]) designed to describe relationships between resources as well as to describe properties like attributes of resources (traditional attribute–value pairs).

Roughly speaking, this vocabulary can be divided conceptually in the following groups:

(a) A set of *properties* which are binary relations between subject resources and object resources: rdfs:subPropertyOf [we will denote it by `sp` in this paper], rdfs:subClassOf [`sc`], rdfs:domain [`dom`], rdfs:range [`range`] and rdf:type [`type`].

(b) A set of classes, that denote set of resources. Elements of a class are known as *instances* of that class. To state that a resource is an instance of a class, the property `type` may be used.

(c) Other functionalities, like a system of classes and properties to describe lists, a systems for doing reification.

(d) Utility vocabulary used to document, comment, etc. The complete vocabulary can be consulted in [46].

The groups in (b), (c) and (d) have a light semantics, essentially describing its internal function in the ontological design of the system of classes of RDFS. Their semantics is defined by “axiomatic triples” [45] which are relationships among these reserved words. All axiomatic triples are “structural”, in the sense that do not refer to external data but talk about themselves. Much of this semantics correspond to what in standard languages is captured via typing.

On the contrary, the group (a) is formed by predicates whose intended meaning is non-trivial and is designed to relate individual pieces of data external to the vocabulary of the language. Their semantics is defined by rules which involve variables (to be instantiated by real data). For example, rdfs:subClassOf [`sc`] is a binary property reflexive and transitive; when combined with rdf:type [`type`] specify that the type of an individual (a class) can be lifted to that of a superclass. This group (a) forms the core of the RDF language. From a theoretical point of view it has been shown to be a very stable core to work with. The detailed arguments supporting this choice are given in [31].

Thus, throughout the paper we will denote the rdfs-vocabulary by rdfs$V$,

$$\text{rdfs}V = \{\texttt{sp}, \texttt{sc}, \texttt{type}, \texttt{dom}, \texttt{range}\}.$$

2.3. Semantics of RDF graphs

In this section we present the formalization of the semantics of RDF following [45,31]. The normative semantics for RDF graphs given in [45] follows standard classical treatment in logic with the notions of model, interpretation, entailment, and so on. We follow here a simplification of the normative semantics proposed in [31]. The two semantics were shown to be equivalent when focusing on the fragment of the RDFS vocabulary that we are considering.

2.3.1. RDF model theory

We first present the notion of interpretation for RDF graphs following [31]. An RDF interpretation is a tuple $\mathcal{I} = (\mathit{Res}, \mathit{Prop}, \mathit{Class}, \mathit{PExt}, \mathit{CExt}, \mathit{Int})$ such that: (1) *Res* is a non-empty set of *resources*, called the *domain* or *universe* of $\mathcal{I}$; (2) *Prop* is a set of property names (not necessarily disjoint from Res); (3) $\mathit{Class} \subseteq \mathit{Res}$ is a distinguished subset of *Res* identifying if a resource denotes a class of resources; (4) $\mathit{PExt} : \mathit{Prop} \to 2^{\mathit{Res} \times \mathit{Res}}$, a mapping that assigns an *extension* to each property name; (5) $\mathit{CExt} : \mathit{Class} \to 2^{\mathit{Res}}$ a mapping that assigns a set of resources to every resource denoting a class; (6) $\mathit{Int} : U \to \mathit{Res} \cup \mathit{Prop}$, the *interpretation mapping*, a mapping that assigns a resource or a property name to each element of U.

Intuitively, a ground triple (s, p, o) in a graph G is true under the interpretation $\mathcal{I}$, if p is *interpreted* as a property name, s and o are *interpreted* as resources, and the interpretation of the pair (s, o) belongs to the extension of the property assigned to p. More formally, we say that $\mathcal{I}$ satisfies the ground triple (s, p, o) if $\mathit{Int}(p) \in \mathit{Prop}$ and $(\mathit{Int}(s), \mathit{Int}(o)) \in \mathit{PExt}(\mathit{Int}(p))$. An interpretation must also satisfy additional conditions induced by the usage of the rdfs-vocabulary. For example, an interpretation satisfying the triple $(c_1, \texttt{sc}, c_2)$ must interpret c_1 and c_2 as classes of resources and must assign to c_1 a subset of the set assigned to c_2. More formally, we say that $\mathcal{I}$ satisfies $(c_1, \texttt{sc}, c_2)$ if $\mathit{Int}(c_1), \mathit{Int}(c_2) \in \mathit{Class}$ and $\mathit{CExt}(c_1) \subseteq \mathit{CExt}(c_2)$.

Blank nodes work as existential variables. Intuitively the triple (X, p, o) would be true under $\mathcal{I}$ if there exists a resource s such that (s, p, o) is true under $\mathcal{I}$. When interpreting blank nodes an arbitrary element can be chosen, taking into account that the same blank node must always be interpreted as the same element.

To formally deal with blank nodes, an extension of the interpretation mapping *Int* is used in the following way. Let $A : B \to Res$ be a function between blank nodes and resources. Denote by $Int_A : UB \to Res$ the extension of function *Int* defined by $Int_A(x) = A(x)$ for $x \in B$, and $Int_A(x) = Int(x)$ for $x \notin B$.

We next formalize the notion of *model* for an RDF graph [45,31]. We say that the RDF interpretation $\mathcal{I} = (Res, Prop, Class, PExt, CExt, Int)$ *models* (is an interpretation for, is a model for) an RDF graph G, denoted by $\mathcal{I} \models G$, if every one of the following conditions holds:

Simple interpretation:

- there exists a function $A : B \to Res$ such that for each $(s, p, o) \in G$,

$$Int(p) \in Prop \quad \text{and} \quad \big(Int_A(s), Int_A(o)\big) \in PExt\big(Int(p)\big).$$

Properties and classes:

- $Int(\mathtt{sp}), Int(\mathtt{sc}), Int(\mathtt{type}), Int(\mathtt{dom}), Int(\mathtt{range}) \in Prop$,
- if $(x, y) \in PExt(Int(\mathtt{dom})) \cup PExt(Int(\mathtt{range}))$ then $x \in Prop$ and $y \in Class$.

Subproperty:

- $PExt(Int(\mathtt{sp}))$ is transitive and reflexive over $Prop$,
- if $(x, y) \in PExt(Int(\mathtt{sp}))$ then $x, y \in Prop$ and $PExt(x) \subseteq PExt(y)$.

Subclass:

- $PExt(Int(\mathtt{sc}))$ is transitive and reflexive over $Class$,
- if $(x, y) \in PExt(Int(\mathtt{sc}))$ then $x, y \in Class$ and $CExt(x) \subseteq CExt(y)$.

Typing:

- $(x, y) \in PExt(Int(\mathtt{type}))$ iff $y \in Class$ and $x \in CExt(y)$,
- if $(x, y) \in PExt(Int(\mathtt{dom}))$ and $(u, v) \in PExt(x)$ then $u \in CExt(y)$,
- if $(x, y) \in PExt(Int(\mathtt{range}))$ and $(u, v) \in PExt(x)$ then $v \in CExt(y)$.

Given G_1 and G_2 RDF graphs, we say that G_1 *entails* G_2, denoted by $G_1 \models G_2$, when for every interpretation $\mathcal{I}$, if $\mathcal{I} \models G_1$ then $\mathcal{I} \models G_2$. We say that two RDF graphs are *equivalent*, denoted $G_1 \equiv G_2$, if and only if $G_1 \models G_2$ and $G_2 \models G_1$. In [31], the authors showed that this entailment notion between RDF graphs is equivalent to the W3C normative notion of entailment [45], when one focuses on the fragment of the RDFS vocabulary that we consider in this paper.

The normative semantics of RDF graphs introduces a simplified notion of model to deal with graphs that do not mention vocabulary with predefined semantics. An interpretation $\mathcal{I}$ is a *simple model* of a graph G, if $\mathcal{I}$ satisfies the *simple interpretation* condition for G. Note that the *simple interpretation* condition is the only one in the formalization of models of RDF graphs that does not mention the rdfs-vocabulary. This discussion motivates the following definition.

Definition 2.2. A simple RDF graph is a graph that does not mention the rdfs-vocabulary, i.e. G is simple iff $\mathrm{rdfs}V \cap \mathrm{voc}(G) = \emptyset$.

Note 2.3 *(RDF versus standard first order semantics).* Probably the reader is asking [her/him]self why all these non-standard idiosyncrasies are needed to define a model theory for RDF. The problem is given by the double role of some elements both, as predicates and as objects. For example the triple $(a, \mathtt{type}, \mathtt{type})$ is a legal one in RDF. In a standard first order logic semantics it should be assigned a binary expression $\mathtt{type}(a, \mathtt{type})$, which does not type check.

2.3.2. Deductive system

In this section, we present a deductive system for the notion of entailment. This system follows the one presented in [31], and is based on a set of rules for $\models$ given in [45].

The system is arranged in five groups of rules. Group A describes the semantics of blank nodes, which is essentially the semantics of simple RDF graphs. Group B and Group C describe the semantics of `sp` and `sc`, respectively. Group D states the semantics of `dom` and `range`, the domain and range of a relation. Group E and Group F force the reflexivity conditions over `sp` and `sc`, respectively. In every rule, capital letters are variables representing elements in *UB*.

GROUP A (*Existential*). For a map $\mu : G' \to G$:

$$\frac{G}{G'} \tag{1}$$

GROUP B (*Subproperty*).

$$\frac{(A, \mathtt{sp}, B)(B, \mathtt{sp}, C)}{(A, \mathtt{sp}, C)} \tag{2}$$

$$\frac{(A, \mathtt{sp}, B)(X, A, Y)}{(X, B, Y)} \tag{3}$$

GROUP C (*Subclass*).

$$\frac{(A, \mathtt{sc}, B)(B, \mathtt{sc}, C)}{(A, \mathtt{sc}, C)} \tag{4}$$

GROUP D (*Typing*).

$$\frac{(A, \mathtt{sc}, B)(X, \mathtt{type}, A)}{(X, \mathtt{type}, B)} \tag{5}$$

$$\frac{(A, \mathtt{dom}, B)(C, \mathtt{sp}, A)(X, C, Y)}{(X, \mathtt{type}, B)} \tag{6}$$

$$\frac{(A, \mathtt{range}, B)(C, \mathtt{sp}, A)(X, C, Y)}{(Y, \mathtt{type}, B)} \tag{7}$$

GROUP E (*Subproperty reflexivity*).

$$\frac{(X, A, Y)}{(A, \mathtt{sp}, A)} \tag{8}$$

$$\frac{}{(p, \mathtt{sp}, p)} \quad \text{for } p \in \{\mathtt{sp}, \mathtt{sc}, \mathtt{dom}, \mathtt{range}, \mathtt{type}\} \tag{9}$$

$$\frac{(A, p, X)}{(A, \mathtt{sp}, A)} \quad \text{for } p \in \{\mathtt{dom}, \mathtt{range}\} \tag{10}$$

$$\frac{(A, \mathtt{sp}, B)}{(A, \mathtt{sp}, A)(B, \mathtt{sp}, B)} \tag{11}$$

GROUP F (*Subclass reflexivity*).

$$\frac{(X, p, A)}{(A, \mathtt{sc}, A)} \quad \text{for } p \in \{\mathtt{dom}, \mathtt{range}, \mathtt{type}\} \tag{12}$$

$$\frac{(A, \mathtt{sc}, B)}{(A, \mathtt{sc}, A)(B, \mathtt{sc}, B)} \tag{13}$$

Note 2.4. As noted in [29,26], the set of rules presented in [45] is not complete for $\models$. The problem is produced when a blank node X is implicitly used as standing for a property in triples like $(a, \mathtt{sp}, X)$, $(X, \mathtt{dom}, b)$, or $(X, \mathtt{range}, c)$. Here we solve the problem following the solution proposed by Marin [29] adding rules (6) and (7). These rules are not defined in [45].

An instantiation of a rule is a uniform replacement of the variables occurring in the triples of the rule by elements of *UB*, such that all the triples obtained after the replacement are well-formed RDF triples, that is, not assigning blank nodes to variables in predicate positions.

Definition 2.5 *(Proof).* Let G and H be RDF graphs. Define $G \vdash H$ iff there exists a sequence of graphs $P_1, P_2, \ldots, P_k$, with $P_1 = G$ and $P_k = H$, and for each j $(2 \leqslant j \leqslant k)$ one of the following cases hold:

- there exists a map $\mu : P_j \to P_{j-1}$ (rule (1)),
- there is an instantiation $\frac{R}{R'}$ of one of the rules (2)–(13), such that $R \subseteq P_{j-1}$ and $P_j = P_{j-1} \cup R'$.

The sequence of rules used at each step (plus its instantiation or map), is called a *proof* of H from G.

The soundness and completeness of this deductive was shown in [31]:

Theorem 2.6 *(Soundness and completeness). (See [31].) Let G and H be RDF graphs, then $G \models H$ iff $G \vdash H$.*

2.4. Characterizations and complexity of entailment

In this section we present some well-known results about the complexity of testing entailment between RDF graphs [22,9,26]. For the sake of completeness we present full proofs of some of these results.

We start by presenting a characterization of the semantic notions of entailment, via mappings between graphs. First we need to define the following notion of closure, similar versions of which has been considered in [45,29,26] using different sets of deductive rules. In Section 3.1 we will discuss in depth the notion of closure.

Definition 2.7 *(Closure* RDFS-cl*).* The graph RDFS-cl(G), the *closure* of G, is defined as the set of triples t which can be deduced from G using rules (2)–(13).

Note that RDFS-cl(G) is an RDF graph over universe(G) plus the rdfs-vocabulary. Because it consists of adding triples by a fixed set of rules starting from G, it is not difficult to check that for a given G, the closure RDFS-cl(G) is unique.

The following result appears in [45], in a slightly different formulation.

Theorem 2.8. *(See [45].) Let G_1 and G_2 be RDF graphs.*

1. $G_1 \models G_2$ *iff there is a map* $\mu : G_2 \to \text{RDFS-cl}(G_1)$.
2. *If G_1 and G_2 are simple graphs then $G_1 \models G_2$ iff there is a map $\mu : G_2 \to G_1$.*

Notice that, from the above theorem it follows directly that two simple RDF graphs G_1 and G_2 are equivalent if and only if there exist mappings $\mu_1 : G_1 \to G_2$ and $\mu_2 : G_2 \to G_1$.

To state the complexity results we use a simple encoding of a standard graph with an RDF graph. Let $H = (V, E)$ be a standard graph, with V a non-empty set of nodes and $E \subseteq V \times V$. Assume we have a set $B_V = \{X_v \mid v \in V\} \subseteq B$ in order to represent the nodes of H, and let e be a distinguished URI reference ($e \in U$). Then H is encoded by the RDF graph $G = \{(X_u, e, X_v) \mid (u, v) \in E\}$. We name G as enc(H).

With the above encoding, we obtain a straightforward connection between the classical notions of homomorphism and isomorphism of standard graphs, and the notions of mapping and isomorphism of RDF graphs. A *homomorphism* from a (standard) graph $H_1 = (V_1, E_1)$ to a graph $H_2 = (V_2, E_2)$, is a function $h : V_1 \to V_2$ such that $(h(u), h(v)) \in E_2$ whenever $(u, v) \in E_1$. When there is a homomorphism from H_1 to H_2 we say that H_1 is *homomorphic* to H_2. An *isomorphism* between $H_1 = (V_1, E_1)$ and $H_2 = (V_2, E_2)$ is a bijection $f : V_1 \to V_2$ such that $(f(u), f(v)) \in E_2$ if and only if $(u, v) \in E_1$. If there is an isomorphism from H_1 to H_2 we say that H_1 and H_2 are *isomorphic*. Given graphs $H_1 = (V_1, E_1)$ and $H_2 = (V_2, E_2)$ it is easy to prove that H_1 is homomorphic to H_2 if and only if there exists a map enc$(H_1) \to$ enc(H_2). Similarly, it holds that H_1 and H_2 are isomorphic if and only if enc$(H_1) \cong$ enc(H_2).

The following complexity results have appeared in several papers in different formulations [45,22,5,26,9]. These results belong to the *folklore* of RDF.

Theorem 2.9 *(Folklore I). Given G_1 and G_2 two simple RDF graphs.*

1. *Deciding if $G_1 \models G_2$ is NP-complete.*
2. *Deciding if $G_1 \equiv G_2$ is NP-complete.*

Proof.

1. Membership in NP follows taking the map as the witness. NP-hardness follows from an encoding of Graph Homomorphism problem, that asks given two graphs H and H' if there is a homomorphism from H to H'. Several NP-complete problems are restrictions of Graph Homomorphism. For example the Clique problem is the restriction when H is a complete graph. For our purpose, take two standard graph $H = (V, E)$ and $H' = (V', E')$ and their encodings as simple RDF graphs $G = \text{enc}(H)$ and $G' = \text{enc}(H')$. We know that H is homomorphic to H' if and only if there is a mapping $G \to G'$, and then we obtain that H is homomorphic to H' if and only if $G' \models G$.
2. Membership in NP follows taking the two maps as the witness. NP-hardness follows from an encoding of the problem of determining whether two graphs H and H' are *homomorphically equivalent*, i.e., whether H is homomorphic to H' and H' is homomorphic to H. Several NP-complete problems are restrictions of this problem. For example, if one restrict this problem to the case in which H is a triangle (K_3) then H is homomorphically equivalent to H' if and only if H' contains a triangle and is 3-colorable. For our purpose, take the standard graphs H and H' and their encodings as simple RDF graphs $G = \text{enc}(H)$ and $G' = \text{enc}(H')$. Now we know that $G \equiv G'$ if and only if there are mappings $G \to G'$ and $G' \to G$, and thus, $G \equiv G'$ if and only if H is homomorphic to H' and H' is homomorphic to H. □

There is also a straightforward connection between the problem of testing entailment of simple RDF graphs, and the problem of evaluating Boolean conjunctive queries. Given an RDF graph G consider for every $p \in \text{voc}(G)$ such that

$(s, p, o) \in G$, a relation name R_p. We associate to G a Boolean conjunctive query Q_G obtained by taking the conjunction of all the predicates $R_p(s, o)$ such that $(s, p, o) \in G$, and considering the blank nodes of G as existentially quantified variables in Q_G (the elements in voc(G) are considered as constants in Q_G). Similarly, we can associate a relational database D_G to every simple RDF graph G as follows. For every $p \in \text{voc}(G)$ such that $(s, p, o) \in G$ there is a relation R_p in D_G containing the set of tuples $\{(s, o) \mid (s, p, o) \in G\}$. Notice that the active domain of D_G is the set universe(G), thus blank nodes are allowed to appear in the tuples of the relations in D_G. It is straightforward that, given G_1 and G_2 simple RDF graphs, $D_{G_1} \models Q_{G_2}$ in the database sense if and only if there is a map $G_2 \to G_1$. Thus, $D_{G_1} \models Q_{G_2}$ if and only if $G_1 \models G_2$.

The above connection can be used to obtain polynomial-time upper bounds for the problem of testing entailment of simple RDF graphs, when one focuses on special classes of graphs. For instance, it is immediate that given a fixed RDF graph G_2, testing $G_1 \models G_2$ can be done in polynomial time. This is obtained as a direct consequence of the data-complexity version of the evaluation problem for conjunctive queries [42], that is, the complexity of the problem when the query is consider to be fixed and only the database is an input parameter. Another case on which testing $G_1 \models G_2$ can be done in polynomial time, and generalizes some results that appear in the *folklore* of RDF [22,26,9], is the case when G_2 has no *cycles induced by blank nodes*. More formally, a cycle induced by blank nodes in a simple RDF graph G, is a sequence $(x_1, x_2, \dots, x_n)$ of elements in universe(G) where $x_n = x_1$ and for every i such that $1 \leqslant i < n$, there exists $p \in \text{voc}(G)$ with $(x_i, p, x_{i+1}) \in G$ or $(x_{i+1}, p, x_i) \in G$, and $x_i, x_{i+1} \in B$. If a simple RDF graph G has no cycles induced by blank nodes, then the associated conjunctive query Q_G is an *acyclic conjunctive query* [40]. In [40] it was shown that evaluating an acyclic conjunctive query can be done in polynomial time. Thus, it follows directly that if G_2 has no cycles induced by blank nodes, then $G_1 \models G_2$ can be tested in polynomial time. Another notion that can be straightforwardly applied to obtain polynomial-time results for the entailment problem, is the notion of *tree-width* of conjunctive queries (see [10,18] for a definition of tree-width of conjunctive queries). It is well known that conjunctive queries of bounded tree-width can be evaluated in polynomial time [10,18]. The notion of bounded tree-width has been recently applied in the RDF context [36].

We end this section by stating the complexity of testing entailment in the presence of rdfs-vocabulary.

Theorem 2.10 *(Folklore II). Checking $G_1 \models G_2$ is NP-complete for general (not necessarily simple) RDF graphs.*

Proof. The hardness part of the proof follows from the fact that simple RDF graphs are special cases of RDF graphs. For the problem of deciding $G_1 \models G_2$ the witness is a proof of G_2 from G_1. In particular, the witness is composed by the graph RDFS-cl(G_1), a map $\mu : G_2 \to \text{RDFS-cl}(G_1)$, a sequence of graphs $G_1^1, G_1^2, \dots, G_1^k$, and a sequence $r_1, r_2, \dots, r_{k-1}$ of instantiations of rules (8)–(7), such that (1) $G_1^1 = G_1$, (2) $G_1^k = \text{RDFS-cl}(G_1)$, and (3) for $1 \leqslant i < k$, G_1^{i+1} is obtained from G_1^i by applying rule r_i. Notice that every G_1^i is a graph over universe(G_1) and then the size of G_1^i is at most cubic with respect to the size of G_1 (because $|G_1^i| \leqslant |\text{universe}(G_1)|^3 \leqslant |G_1|^3$). Moreover, the application of every rule adds at least one triple, and then $k \leqslant |G_1|^3$, implying that the witness is of polynomial size. The witness can be used to check entailment in polynomial time, first checking that G_1^{i+1} is obtained from G_1^i for every i, checking that RDFS-cl(G_1) is closed under application of rules r (8)–(7), and then using the mapping μ and the characterization of Theorem 2.8. □

3. Normal forms

For each RDF graph there exist many different equivalent RDF graphs. For several purposes, it is convenient to choose a distinguished representative of each such equivalence class, that is, a "normalized" version of an RDF graph.

In the normative document specifying the semantics of RDF [45] there are notions that point in this direction, although none completely satisfactory. Given an RDF graph, that document defines a notion of minimal graph, a *lean* graph. We prove that for each RDF graph G, there is a unique lean graph equivalent to it (modulo isomorphism), and following well-known notions in graph theory, we call it the *core* of G. On the other hand, the document defines a notion of maximal graph, that of closure (formalized in Definition 2.7) which we call RDFS closure in this paper.

In this section we discuss pros and cons of these notions, and propose a notion of normal form for RDF graphs. In Section 3.1, we study maximal representations, give a semantics definition of closure and show that it is equivalent to the RDFS closure. In Section 3.2, we study minimal representations, in particular the notion of core. Based on the notions of core and closure, we formalize the notion of normal form, show that it improves the RDFS closure in different aspects, and study the complexity of computing it.

3.1. Maximal representations

In this section we explore maximal representations of RDF graphs, where the standard mathematical notion is that of closure: a maximal (with respect to some metric) object of the same kind but equivalent to the original one. Thus a naive definition is the following:

Definition 3.1 *(Naive closure).* A *closure* of a graph G is a maximal set of triples G' over universe(G) plus the rdfs-vocabulary such that G' contains G and is equivalent to it.

Example 3.2 shows that with this definition, there could be more than one closure for a graph.

Example 3.2. There could be more than one closure of a graph. For example the graph

$$\begin{tikzcd}
& c \arrow[rd, "r"] & \\
a \arrow[ru, "p"] \arrow[r, "p"] \arrow[rd, "p"'] & X & d \\
& b \arrow[ru, "q"'] &
\end{tikzcd}$$

where p, q, r are different properties, has two different non-isomorphic closures, namely, either adding the triple (X, r, d) or the triple (X, q, d) (but not both).

A relation between the notions of closure of Definitions 2.7 and 3.1 is presented in the following lemma.

Lemma 3.3. *Let G' be any closure of G as defined in Definition* 3.1. *Then* RDFS-cl$(G) \subseteq G'$.

Proof. Let G' be a closure of G, we will show that RDFS-cl$(G) \cap G' =$ RDFS-cl(G). Let $G^\star =$ RDFS-cl$(G) \cap G'$, and suppose that $G^\star \neq$ RDFS-cl(G). Then the set RDFS-cl$(G) - G^\star$ is not empty. By the construction of RDFS-cl(G) and because $G \subseteq G^\star$ there exists $t \in$ RDFS-cl$(G) - G^\star$ such that $G^\star \vdash_r G^\star \cup \{t\}$ and then because $G^\star \subseteq G'$ we have that $G' \vdash_r G' \cup \{t\}$, and then $G' \equiv G' \cup \{t\}$. Note that $t \notin G'$ and then $G' \equiv G' \cup \{t\}$ is a contradiction with the maximality of G'. Finally $G^\star =$ RDFS-cl(G) and then RDFS-cl$(G) \subseteq G'$. □

What make the difference between the semantic notion of closure of Definition 3.1 and RDFS-cl are the presence and treatment of blank nodes. The following notion is motivated by this observation.

The notion of *Herbrand Model* of a structure, is, roughly speaking, a model in which each syntactic object (ground term) is represented as itself. When constructing Herbrand Models for sets of existential first order sentences, the idea of *Skolemization* comes into play. The Skolemization of an existential sentence consists simply in replacing every existentially quantified variable by a brand new constant. Following this idea we can construct a ground graph associated to every RDF graph. Formally, given an RDF graph G, define G^* as the RDF graph obtained by replacing each blank X in G by a fresh constant c_X. Conversely, H_* denotes the graph obtained from H after replacing each constant c_X by the blank X and deleting triples having blanks as predicates (which are not well-defined triples in the RDF specification).

From the definition of RDFS-cl follows without difficulty the next lemma:

Lemma 3.4. *Let G be an RDF graph, then* RDFS-cl$(G) = ($RDFS-cl$(G^*))_*$.

We are ready to present a robust semantic notion of closure, not depending on the set of rules as RDFS-cl, and not having the problems of Definition 3.1:

Definition 3.5. A closure of a graph G, denoted cl(G), is a graph G' that satisfies: if G is a ground graph, then G' is a maximal ground graph equivalent to G, otherwise, $G' = H_*$, where H is a closure of G^*.

Theorem 3.6. *(Cf. [31].) For each graph G:*

1. *The closure* cl(G) *is unique.*
2. cl(G) *coincides with the operational definition given in Hayes' document, that is,* cl$(G) =$ RDFS-cl(G).
3. cl(G) *has size $\Theta(|G|^2)$.*
4. *Deciding if $t \in$* cl(G) *can be computed in time $\mathcal{O}(|G| \log |G|)$.*

Proof. 1. Suppose that G' and G'' are closures of a ground graph G, that is, G' and G'' are two maximal ground RDF graphs equivalent to G. Then, by Theorem 2.8, $G' \subseteq$ RDFS-cl(G) and $G'' \subseteq$ RDFS-cl(G). But, since RDFS-cl$(G) \equiv G$, and G' and G'' are maximal, it must be the case that $G' =$ RDFS-cl$(G) = G''$. For non-ground graph uniqueness follows from the injective nature of the operators $(\cdot)^*$ and $(\cdot)_*$.

2. If G is a ground graph, the previous proof shows that cl$(G) =$ RDFS-cl(G). Otherwise, we have cl$(G^*) =$ RDFS-cl(G^*), and hence $($cl$(G^*))_* = ($RDFS-cl$(G^*))_*$. Thus, it is enough to prove that RDFS-cl$(G) = ($RDFS-cl$(G^*))_*$, which is Lemma 3.4.

Items 3 and 4 are Theorems 6 and 7 in [31], where the proofs can be found. □

3.2. Minimal representations

We study in this section the notion of *core* for RDF graphs which is related to the notion of *lean* presented in [45]. Similar notions have been investigated in other contexts [25,23,15]. For RDF graphs, this notion is closely bound to minimal representations.

Definition 3.7. A graph G is *lean* if there is no map μ such that $\mu(G)$ is a proper subgraph of G.

Example 3.8. Let p, q, r be different predicates and consider:

$$\begin{tikzcd} G_1 & a \arrow[r, "p"] \arrow[d, "p"'] & X & G_2 & a \arrow[r, "p"] \arrow[d, "p"'] & X \arrow[d, "q"] \\ & Y & & & Y \arrow[r, "r"'] & b \end{tikzcd}$$

Then G_1 is not lean, but G_2 is lean because there is no proper map of G_2 into itself.

Lemma 3.9. *Let G_1 and G_2 be two lean RDF graphs. Then $G_1 \cong G_2$ if and only if there are maps $G_1 \to G_2$ and $G_2 \to G_1$.*

Proof. The "only if" part is trivial by the definition of $\cong$. For the "if" part, suppose that there are two mappings $\mu_1 : G_1 \to G_2$ and $\mu_2 : G_2 \to G_1$. First, we know that $\mu_1(G_1) \subseteq G_2$ and $\mu_2(G_2) \subseteq G_1$ implying that $(\mu_1\mu_2)(G_2) \subseteq \mu_1(G_1) \subseteq G_2$. We have that $\mu_1\mu_2$ is a mapping such that $(\mu_1\mu_2)(G_2) \subseteq G_2$ and then $(\mu_1\mu_2)(G_2) = G_2$ because G_2 is lean. We have obtained that $G_2 = (\mu_1\mu_2)(G_2) \subseteq \mu_1(G_1) \subseteq G_2$, and then $\mu_1(G_1) = G_2$. Similarly we obtain that $\mu_2(G_2) = G_1$ and then $G_1 \cong G_2$. □

The following theorem will be the basis of the applications of the notion of core in the context of RDF graphs.

Theorem 3.10 *(Core). Each RDF graph G contains a unique (up to isomorphism) lean subgraph which is an instance of G. We will denote this unique subgraph by* $\text{core}(G)$.

Proof. First we show by induction on the size of G that every G contains a lean subgraph which is an instance of G. If G is lean there is nothing to prove. Assume that G is not lean, then by definition there is a map μ such that $\mu(G) \subsetneq G$. By induction hypothesis the graph $\mu(G)$ contains a lean subgraph which is an instance of $\mu(G)$, i.e. there is a map μ' such that $\mu'(\mu(G)) \subseteq \mu(G)$ and $\mu'(\mu(G))$ is lean, then $\mu'(\mu(G)) \subseteq G$ and $\mu'(\mu(G))$ is a lean instance of G, completing this part of the proof.

Now we must show that the lean sub-instance is unique up to isomorphism. Consider G and two lean subgraphs G_1 and G_2 that are instances of G. Because G_1 is an instance of G there is a map $G \to G_1$, and because $G_2 \subseteq G$ there is a map (the identity) $G_2 \to G$, and then there is a map $G_2 \to G_1$. Similarly there is a map $G_1 \to G_2$ and because G_1 and G_2 are lean graphs and applying Lemma 3.9 we obtain that $G_1 \cong G_2$. □

Note that by definition of core we have $G \vdash \text{core}(G)$ and $\text{core}(G) \vdash G$, and then every RDF graph is equivalent to its core, that is $G \equiv \text{core}(G)$.

For simple graph cores behave well: the concept of lean graph corresponds exactly to minimal representations, and allow to reduce logical equivalence to isomorphism of graphs, as the next theorem shows.

Theorem 3.11 *(Cores for simple RDF graphs). Let G, G_1, G_2 be simple RDF graphs. Then:*

1. $\text{core}(G)$ *is the unique (up to isomorphism) minimal (w.r.t. number of triples) graph equivalent to G.*
2. $G_1 \equiv G_2$ *if and only if* $\text{core}(G_1) \cong \text{core}(G_2)$.

Proof. (1) Suppose that G_m is another minimal graph such that $G \equiv G_m$. First note that G_m must be a lean graph, because, if it were not lean then $\text{core}(G_m) \subsetneq G_m$ and $\text{core}(G_m) \equiv G_m \equiv G$ and then G_m would not be a minimal (w.r.t. number of triples) graph equivalent to G. Now $G_m \equiv G \equiv \text{core}(G)$ and then $G_m \equiv \text{core}(G)$, and because they are both lean simple graphs, and by Theorem 2.8 and Lemma 3.9, it holds that $G_m \cong \text{core}(G)$.

(2) Let G_1 and G_2 be simple RDF graphs, then $G_1 \equiv G_2$ iff $G_1 \to G_2$ and $G_2 \to G_1$ iff $\text{core}(G_1) \to \text{core}(G_2)$ and $\text{core}(G_2) \to \text{core}(G_1)$ iff $\text{core}(G_1) \cong \text{core}(G_2)$ by Lemma 3.9. □

The bad news is that computing cores is hard:

Theorem 3.12. *Let G, G' be RDF graphs.*

1. *Deciding if G is lean is coNP-complete.*
2. *Deciding if $G' \cong \mathrm{core}(G)$ is DP-complete.*

Proof. Both proofs are based on encodings of graph theoretic problems dealing with the well-known notion of core: The core of a graph H is the smallest subgraph of H that is also a homomorphic image of H.

1. The proof is an encoding of the problem Core:
   Instance: A graph H.
   Question: Is there a homomorphism of H to a proper subgraph?
   This problem was shown to be NP-complete by Hell and Nesetril [25]. Encode the graph $H = (V, E)$ as the RDF graph $G = \mathrm{enc}(H)$. Now H has a homomorphism to a proper subgraph iff the RDF graph G has an instance that is a proper subgraph, i.e. iff G is not lean. We have show that deciding if G is not lean is NP-hard. Now for the membership in NP the certificate is the mapping μ such that $\mu(G)$ is a proper subset of G. The property $\mu(G) \subsetneq G$ can be checked in polynomial time.
2. The proof is an encoding of the problem Core Identification:
   Instance: Two graphs H and H'.
   Question: Is H' the graph theoretic core of H.
   This problem was shown to be DP-complete in [15]. Encode the graphs H and H' as RDF graphs $G = \mathrm{enc}(H)$ and $G' = \mathrm{enc}(H')$. H' is the graph theoretic core of H, iff H' is isomorphic to a subgraph of H that is an homomorphic image of H and has non-homomorphism to a proper subgraph, iff G' is isomorphic to a subgraph of G that is an instance of G and has no instance that is a proper subgraph, iff G' is isomorphic to a lean subgraph of G that is an instance of G, iff $G' \cong \mathrm{core}(G)$ by uniqueness of the core. Now for the membership in DP, one can split the problem in two, first checking if G' is isomorphic to a subgraph of G and an instance of G which both are NP (taking the maps as certificate), and then checking if G' is lean that we know is coNP. □

For the case of RDF graphs with RDFS vocabulary, things become more complex. Let us introduce a notion of minimal representation.

Definition 3.13. A *minimal representation* of a graph G is a minimal (w.r.t. number of triples) graph equivalent to G and contained in G.

We have seen that in the case of simple graphs core(G) is the (unique up to isomorphism) minimal representation of G. Unfortunately, for the case of general graphs we do not have such unique minimal representations in the general case. The semantics of the vocabulary plays a crucial role.

Example 3.14. There could be more than one reduction of a given graph. This follows from the transitive property of sc and sp and classical results on transitive reduction on graphs [1]. The standard example is:

$$\begin{tikzcd}
G & & a & \\
& b \arrow[ur, "\mathtt{sp}"] \arrow[rr, bend left, "\mathtt{sp}"] & & c \arrow[ul, "\mathtt{sp}"'] \arrow[ll, bend left, "\mathtt{sp}"]
\end{tikzcd}$$

The graphs obtained by deleting either $(b, \mathtt{sp}, a)$ or $(c, \mathtt{sp}, a)$, are two non-isomorphic reductions of G.

It is known that the transitive reduction of an acyclic graph is unique [1]. The class of graphs acyclic for the properties of sp and sc form a big and important class. In fact, in modeling, this is considered good practice [17]. Hence this could be a promising class with minimal representation. Unfortunately, we still have problems. Consider the following graph:

Example 3.15. Consider the graph $G = \{(a, \mathtt{sc}, b), (\mathtt{type}, \mathtt{dom}, a), (x, \mathtt{type}, a), (x, \mathtt{type}, b)\}$.

Even though it is acyclic w.r.t. subproperty and subclass, it has two non-isomorphic minimal representations:

$$G_1 = \{(a, \mathtt{sc}, b), (\mathtt{type}, \mathtt{dom}, a), (x, \mathtt{type}, a)\}$$

$$G_2 = \{(a, \mathtt{sc}, b), (\mathtt{type}, \mathtt{dom}, a), (x, \mathtt{type}, b)\}$$

In G_1 the missing triple can be obtained via rule (5), and in G_2 via rule (6) (replacing $A = \mathtt{type}$, $C = A$, $B = b$ and $X = x$).

The problem occurs because of the presence of vocabulary with predefined semantics in the subject or object positions in a triple. If we forbid these triples, we get an important subclass of RDF graphs with vocabulary for which there are minimal representations.

Theorem 3.16. *Let G be an RDF graph with no reserved vocabulary in the subject nor in the object position, and acyclic w.r.t. subproperty and subclass. Then G has a unique minimal representation.*

Proof. Consider the intersection of all minimal representations of G. We will prove that this graph *is* the unique minimal representation of G. We will prove that if t is a triple of G, either it is in all minimal representations, or it is in none of them (thus, by definition of minimal representation, it can be deduced in all of them).

Before we need some additional notions. Construct the graph $G_{\mathtt{sc}}$ built based on all triples $(a, \mathtt{sc}, b)$ of G as follows: vertices, all subject and object elements of such triples; a directed edge (a, b) if and only if the triple $(a, \mathtt{sc}, b)$ is in G. Similarly construct $G_{\mathtt{sp}}$.

Next, if c is a vertex in $G_{\mathtt{sp}}$, and if there is a triple (x, c, y) in G, mark in the graph $G_{\mathtt{sp}}$ the node c and all its descendants with the pair (x, y).

First, assume the graph G has no reflexive triples of the kind $(a, \mathtt{sc}, a)$ nor $(a, \mathtt{sp}, a)$. Assume G_1 and G_2 are minimal representations of G.

1. $(a, \mathtt{sc}, c)$. The only triples in all minimal representations are the ones in a transitive reduction of $G_{\mathtt{sc}}$, which is unique for acyclic graphs.
   Note that because of our assumptions (no reserved vocabulary in subject nor object positions) the rule (3) will not deduce triples of the kind $(a, \mathtt{sc}, c)$. Thus the only way to deduce such triples is using the transitivity rule (4).
2. $(a, \mathtt{sp}, c)$. Similar as before.
3. All triples of the form $(a, \mathtt{dom}, c)$ and $(a, \mathtt{range}, c)$ are preserved in any minimal representation, because there is no way of deducing them from other triples.
4. If $t = (a, b, c)$ with no vocabulary RDFS involved, then (a, c) is a label of node b in $G_{\mathtt{sp}}$ (which is the same graph as $(G_1)_{\mathtt{sp}}$ and $(G_2)_{\mathtt{sp}}$. In fact, one key point in the analysis is that these graphs do not change because there are no triples with $\mathtt{sc}$ nor $\mathtt{sp}$ allowed in object or subject positions). The *only* way to deduce (a, b, c) is the existence of node d in $G_{\mathtt{sp}}$ with label (a, c) which is an ancestor of b.
   Then if $b = d$ the triple (a, b, c) should be in all minimal representations. If $b \neq d$, then (a, b, c) can be safely ignored in any minimal representation.
5. $(a, \mathtt{type}, c)$. This kind of triple deserves careful analysis. Assume there is a triple $(a, \mathtt{type}, b)$ which is in G_1. With the restrictions imposed, note that a triple of the form $(a, \mathtt{type}, c)$ can be deduced only by rule (5) or rules (6) or (7). Hence a triple $(a, \mathtt{type}, c)$ is in a minimal representation if and only if is deduced by these rules. By induction, we already know that the antecedents of these rules must be the same in any minimal representation. Hence the result follows.

Now, if G has a reflexive triple of the kind $(a, \mathtt{sc}, a)$ or $(a, \mathtt{sp}, a)$, note that either, it can be eliminated in any minimal representation because it can be obtained by one of the rules (8)–(13), or it cannot be obtained by these rules, in which case is must be in any minimal representation. □

3.3. Normal form

Although the notion of closure allows us to reduce RDF entailment to the existence of a mapping between two RDF graphs (as Theorem 2.8 shows), it has some drawbacks, probably the most relevant is that it is syntax dependent, that is, for graphs G, H it is not necessarily the case that $\mathrm{cl}(G) \cong \mathrm{cl}(H)$.

Example 3.17. Consider the following equivalent graphs G and H (where N is a blank node):

$$\begin{tikzcd}
 & & & N \\
G: & a \arrow[r, "sc"'] & b \arrow[r, "sc"'] \arrow[ur, "sc"] & c
\end{tikzcd}$$

and

$$\begin{tikzcd}
H: & a \arrow[r, "sc"'] \arrow[rr, bend right, "sc"'] & b \arrow[r, "sc"'] & c
\end{tikzcd}$$

Then we have that even though $G \equiv H$, RDFS-$\mathrm{cl}(G) \ncong$ RDFS-$\mathrm{cl}(H)$ and $\mathrm{cl}(G) \ncong \mathrm{cl}(H)$. Moreover, also $\mathrm{core}(G) \ncong \mathrm{core}(H)$.

In fact, one would like a data representation, usually called normal form, with the following properties:

1. (Uniqueness). The normal form of a graph G, $\mathrm{nf}(G)$, is unique (up to isomorphism).

2. (Syntax-independence). For all graphs H, G, $G \equiv H$ if and only if $\mathrm{nf}(G) \cong \mathrm{nf}(H)$.

The maximal and minimal representations we studied do not have these properties as Example 3.17 shows. For simple RDF graphs, the core is really a normal form. But in general, it does not work because it is not unique. On the other hand, the closure is not syntax independent. Hence we will have to look for a compromise. The following definition introduces, a combination of the closure and the core, fulfill the desiderata.

Definition 3.18. For a graph G, define its *normal form*, denoted $\mathrm{nf}(G)$, as the core of the closure of G, that is, $\mathrm{nf}(G) = \mathrm{core}(\mathrm{cl}(G))$.

The next theorem shows that the notion of normal form meets our desiderata.

Theorem 3.19 *(Normal forms for RDF graphs). Let G and H be RDF graphs. Then*:

1. *The normal form is unique* (*up to isomorphism*).
2. *The normal form* $\mathrm{nf}(G)$ *is syntax independent, that is,* $G \equiv H$ *if and only if* $\mathrm{nf}(G) \cong \mathrm{nf}(H)$.

Proof. 1. Direct consequence of Theorem 3.6 and the uniqueness (up to isomorphism) of the core.

2. Proposition 3.6 and Theorem 2.8 imply that $G \models H$ if and only if $H \to \mathrm{cl}(G) \to \mathrm{core}(\mathrm{cl}(G)) = \mathrm{nf}(G)$. Hence, the statement follows using the fact that $\mathrm{nf}(G)$ and $\mathrm{nf}(H)$ are lean graphs applying Lemma 3.9. □

Note that the normal form for graphs G and H of Example 3.17 is H.

Unfortunately, computing the normal form is hard:

Theorem 3.20. *Let G, G' be graphs. The problem of deciding if G' is the normal form of G is DP-complete.*

Proof. The hardness part follows from the fact that if G is a simple graph deciding if G' is the normal form of G is equivalent to deciding if G' is the core of G. By Theorem 3.12 we know that this last problem is DP-hard. For the membership in DP, the problem is equivalent to test whether G' is the core of $\mathrm{cl}(G)$. We can split this problem in two, first checking if there is a map $\mathrm{cl}(G) \to G'$ which is NP, and then checking if G' is lean that we know is coNP. □

4. RDF query languages

Let V be a set of variables (disjoint from *UB*). Individual variables will be denoted $?X, ?Y, ?\mathit{Person}$, etc.

As query language, we will use the notion of *tableau* borrowed from the database literature (see for example [2]) but slightly extended to allow also a set of tuples in the head. A *tableau* is a pair (H, B) where H, B are RDF graphs with some elements of UBs replaced by variables in V, B has no blank nodes, and all variables of H occur also in B. We often write a tableau in the form $H \leftarrow B$ to indicate the similarity with logic programming and Datalog.

For example, a tableau such as

$$(?A, \text{creates}, ?Y) \leftarrow (?A, \text{type}, \mathit{Flemish}), (?A, \text{paints}, ?Y), (?Y, \text{exhibited}, \text{Uffizi})$$

where identifiers preceded by ? are variables, intuitively defines the artifacts created by Flemish artists being exhibited at Uffizi Gallery.

Definition 4.1. A *query* is a tableau (H, B) plus a set of premises P and a set of constraints C, where P is a graph over UB (i.e. with no variables) and C is a subset of the variables occurring in H. In other words, a query is a tuple (H, B, P, C).

When P is omitted we assume the premise is empty, i.e. write (H, B, C) instead of $(H, B, \emptyset, C)$. Similarly for the set of constraints C or both.

The set of constraints C gives the user the possibility to discriminate between blank and ground nodes in answers and plays the same role as IS NOT NULL in SQL. For example, the tableau above is a query with no constraints. We can add to it the constraint $\{?A\}$; intuitively, as we will formalize in the next subsection, this means that the $?A$ variable must be bound to a non-blank element in each answer to the query.

The premise P represents information the user supplies to the database to be queried in order to answer the query. For example, the query:

$$(?X, \text{relative}, \mathit{Peter}) \leftarrow (?X, \text{relative}, \mathit{Peter})$$

with premise $P = \{(\mathit{son}, \mathtt{sp}, \mathit{relative})\}$ ask for all relatives of Peter knowing that “son” is a subproperty of “relative”.

Note 4.2. The condition $var(H) \subseteq var(B)$ avoids the presence of free variables in the head of the query. The presence of blank nodes in the body of the query is unnecessary, because—as we will see—a variable plays exactly the same role in this position. However, we do allow blank nodes in the head of the query to permit some features which will clear in what follows.

4.1. Answers to a query

Let $q = (H, B, P, C)$ be a query, D a database, and V a set of variables. This section defines the semantics of the query q over the database D.

A *valuation* is a function $v : V \to UB$. For a set $C \subseteq V$ of variables, the valuation v satisfies the constraint C (denoted $v \models C$) if for all $x \in C$, $v(x)$ is not blank.[2] We define $v(B)$ as the graph obtained after replacing every occurrence of a variable x in B by $v(x)$.

A *matching* of the graph B in database D is a valuation v such that $v(B) \subseteq \mathrm{nf}(D)$. The matchings that interest us are those that satisfy the constraints C.

The semantics includes, for each blank node N occurring in H, a Skolem function $f_N : (UB)^k \to C$, where k is the number of distinct variables occurring in B and C a set of blank nodes disjoint with any appearing in the query or the database. For each valuation v, $v(H)$ is the graph obtained by replacing each variable $?X$ occurring in H by $v(?X)$ and each blank node N occurring in H by $f_N(v(?X_1), \dots, v(?X_k))$ where $?X_1, \dots, ?X_k$ are the variables occurring in B.

Definition 4.3. Let $q = (H, B, P, C)$ be a query and D a database. A *pre-answer* to q over D is the set

$$\mathrm{preans}(q, D) = \{ v(H) \mid v \text{ is a matching of } B \text{ in } D + P \text{ and } v \models C, \text{ and } v(H) \text{ is a well-formed RDF graph} \}.$$

A graph $v(H)$ in $\mathrm{preans}(q, D)$ is called a *single* answer of the query q over D.

Note 4.4. Some clarifications about the notion of matching are in order. We would like to preserve the semantics of answers under equivalence of datasets, that is if $D \equiv D'$, then the answer of q against D should be the same as the answers of q against D'.

For this, we need to query $\mathrm{nf}(D)$ instead of just D in order to deal with rdfs vocabulary because entailment in this case is characterized in terms of nf (cf. Theorem 3.19). Recall that using simply a closure D' of D instead of $\mathrm{nf}(D)$ would not give unique answers as shown by the RDF graphs in Example 3.17.

A more general definition of matching obtained by replacing "$v(B) \subseteq \mathrm{nf}(D)$" by "$D \models v(B)$" does not work properly because it could give infinite answers. For example, given $D = \{(a, b, c)\}$ and the query defined as $(?X, ?Y, ?Z) \leftarrow (?X, ?Y, ?Z)$, the answers would be D union all triples of the form (N, b, M) with N, M blank nodes.

A desirable property a query language for RDF should have is compositionality, i.e., the property that complex queries can be composed from simpler ones [6]. For this, we need to output results in the same format as input data. In our case, we can combine single answers in several different ways to obtain as final answer an RDF graph. We concentrate on two approaches to do this:

1. $\mathrm{ans}_\cup(q, D)$ is the *union* of all single answers. With this approach, queries properly capture the information carried by blank nodes inside D (in particular when blank nodes play the role of bridges between two single answers).
2. An alternative approach, $\mathrm{ans}_+(q, D)$, is to *merge* all single answers, which means to rename blank nodes if necessary to avoid name clashes.

Note that if there are no blank nodes in D, both approaches are the same.

The merge-semantics could be useful when querying several sources (e.g. several different files of metadata corresponding to different web pages). In this case we do not want clashes of blank nodes of different specifications. One important drawback of the merge-semantics is that there could be no data-independent query that retrieves the whole database. An approach similar to merge-semantics can be found in query languages for semistructured data [33].

The union-semantics is more intuitive. First, there exists an identity query (see Note 4.7 below). As another illustration, consider a database D which has a blank node N with several properties, i.e., there exist in D several triples $(N, p_1, z_1), (N, p_2, z_3), \dots$. If we follow the merge-semantics, we cannot retrieve the properties of N with a data-independent query. On the other hand, if we follow the union-semantics, the query $(?X, \mathit{feature}, ?Y) \leftarrow (?X, ?Y, ?Z)$ will do it.

Proposition 4.5. *Let $q = (H, B, C, P)$ be a query. Assume that when querying any database, the same Skolem function f_N is used for every blank node N in H. Then:*

[2] This constraint is called a *must-bind variable* in DQL [43].

1. *For both semantics, if* $D' \models D$ *then* $\mathrm{ans}(q, D') \models \mathrm{ans}(q, D)$.
2. *For all* D, $\mathrm{ans}_{\cup}(q, D) \models \mathrm{ans}_{+}(q, D)$.

Proof. 1. Note that from $D' \models D$ follows that $D' + P \models D + P$ and then by Theorem 3.19 there is map μ with $\mu(\mathrm{nf}(D+P)) \subseteq \mathrm{nf}(D' + P)$. In the proof we will use a map μ' equal to μ in the blanks of $D + P$ and the identity outside. The restriction on μ' to be the identity outside the blanks of D is to ensure that μ' do not change the value of any Skolem function used in the answer.

For merge semantics is enough to show that for every graph $G \in \mathrm{preans}(q, D)$, there is a graph G' in $\mathrm{preans}(q, D')$ such that $G' \models G$. Let $G \in \mathrm{preans}(q, D)$, then $G = v(H)$ for some valuation v that satisfies the conditions C and $v(B) \subseteq \mathrm{nf}(D + P)$. Note that the function $\mu' v : V \to UB$ is a valuation that satisfies the conditions C (because μ' is the identity over U) and $(\mu' v)(B) \subseteq \mu(\mathrm{nf}(D + P)) \subseteq \mathrm{nf}(D' + P)$, and then $(\mu' v)(H) \in \mathrm{preans}(q, D')$ because $(\mu' v)(N) = v(N)$ for any blank node N in H. Finally, let $G' = (\mu' v)(H) = \mu'(G)$, then $G' \models G$ and $G' \in \mathrm{preans}(q, D')$ completing this part of the proof.

For union semantics, let $t \in \mu'(\mathrm{ans}_{\cup}(q, D))$, then $t \in \mu'(v(H))$ for a valuation v that satisfies C and such that $v(B) \subseteq \mathrm{nf}(D + P)$. This last statement implies that $(\mu' v)(B) \subseteq \mathrm{nf}(D' + P)$, and because $\mu' v : V \to UB$ is a valuation that satisfies the conditions C and $(\mu' v)(N) = v(N)$ for any blank node N in H, we have that $(\mu' v)(H) \in \mathrm{preans}(q, D')$, and then $(\mu' v)(H) \subseteq \mathrm{ans}_{\cup}(q, D')$. Finally $t \in \mathrm{ans}_{\cup}(q, D')$ and then $\mu'(\mathrm{ans}_{\cup}(q, D)) \subseteq \mathrm{ans}_{\cup}(q, D')$ which implies that $\mathrm{ans}_{\cup}(q, D') \models \mathrm{ans}_{\cup}(q, D)$.

2. It follows from the general fact $G_1 \cup G_2 \models G_1 + G_2$. □

Theorem 4.6. *Let* $q = (H, B, C, P)$ *be a query, if* $D \equiv D'$ *then* $\mathrm{ans}(q, D) \cong \mathrm{ans}(q, D')$.

Proof. From $D' \equiv D$ follows that $D' + P \equiv D + P$ and then by Theorem 3.19 there $\mathrm{nf}(D + P) \cong \mathrm{nf}(D' + P)$. Consider an isomorphism $\mu : \mathrm{nf}(D + P) \to \mathrm{nf}(D' + P)$ such that μ is the identity outside the blanks of $D \cup D'$. Then μ witnesses the desired isomorphism. The restriction on μ to be the identity outside the blanks of $D \cup D'$ is to ensure that μ do not change the value of any Skolem function used in the answer. □

Note 4.7 *(The identity query).* The identity query is defined as (H, B) with $H = B = \{(?X, ?Y, ?Z)\}$.

Observe that this query works as identity *modulo equivalence* only with the union-semantics. Consider the database $D = \{(X, b, c), (X, b, d)\}$. Then $\mathrm{ans}_{\cup}(q, D) \equiv D$, but $\mathrm{ans}_{+}(q, D) = \{(X, b, c), (Y, b, d)\}$, which is not equivalent to D because there is no map from D to $\mathrm{ans}_{+}(q, D)$. This shows also that the converse of Proposition 4.5, item 2, does not hold.

In the sequel, unless stated otherwise, we will assume the union-semantics.

4.2. Premises

Having premises in queries extends classical querying in several aspects: The possibility of simulating if-then queries while still remaining within the expressiveness of the language; hypothetical analysis of information; and the ability to query incomplete information by supplying information not in the database. This is particularly relevant when querying sources which use external ontologies.

Our definition of premises differs from Bonner's [7] in that we have one fixed premise for the whole query. We also allow blank nodes, but not variables, in the premise.

It is important to remark that premises cannot be simulated with Datalog programs. For example consider the following query:

$$(?X, \text{relative}, \textit{Mary}) \leftarrow (?X, \text{relative}, \textit{Mary})$$

with premise $P = \{(\textit{son}, \mathtt{sp}, \textit{descendant})\}$.

It is not possible to write a data-independent Datalog-like query equivalent to it. The reason is that we do not know in advance the existence, in a given database, of triples like $(\textit{descendant}, \mathtt{sp}, \textit{relative})$ that could indirectly link "son" with "relative" via the transitive relation sp.

5. Query containment

In this section we explore different notions of query containment and their characterizations. In Section 5.1, we introduce two different notions of query containment for RDF queries. In Section 5.2 we give characterizations for the two notions in terms of mappings between the queries involved, for the case of queries without premises. Finally, in Sections 5.3 and 5.4, respectively, we study the containment problem under premises and constraints.

5.1. Notions of query containment

Any reasonable notion of query containment $q \sqsubseteq q'$ should embody the idea that $\mathrm{ans}(q', D)$ comprises all the information of $\mathrm{ans}(q, D)$. In relational databases, set-theoretical inclusion of tuples captures this requirement. When databases are viewed as knowledge bases having a notion of entailment (denoted in what follows by $\models$), the information comprised by a database is all that can be entailed from it. Hence the right notion of $q \sqsubseteq q'$ is $\text{ans}(q', D) \models \text{ans}(q, D)$ for all D. In the relational case both notions coincide. This is not the case in our context. In what follows we will discuss these two versions of containment.

Given sets S, S' of RDF graphs, we write that $S \subseteq_{iso} S'$ iff for each $G \in S$, there is $G' \in S'$ with $G' \cong G$.

Definition 5.1 *(Containment).* Let q, q' be queries.

1. (Standard containment). $q \sqsubseteq_p q'$ iff for all databases D, $\text{preans}(q, D) \subseteq_{iso} \text{preans}(q', D)$.
2. (Entailment-based containment). $q \sqsubseteq_m q'$ iff for all databases D, $\text{ans}(q', D) \models \text{ans}(q, D)$.

Proposition 5.2. $\sqsubseteq_p$ *implies* $\sqsubseteq_m$.

Proof. Let $q = (H, B, P, C)$ and $q' = (H', B', P', C')$ two queries. Suppose that $q \sqsubseteq_p q'$, then given a simple answer $v(H) \in \text{preans}(q, D)$, there must exist $v'(H') \in \text{preans}(q, D)$ with $v'(H') \cong v(H)$ via a map μ that preserves blank nodes of D. Otherwise we replace blanks of D by fresh constants obtaining a contradiction. Now the union of such maps $\bigcup \mu$ is a map from $\text{ans}(q, D)$ to $\text{ans}(q', D)$. $\square$

The converse of this proposition is not true as the following examples show.

Example 5.3. When working with rdfs vocabulary, containment characterizations are more complex. In the following queries, the head (not depicted) is assumed to be the same as the body.

B:

$$\begin{tikzcd}
 & ?Y \\
?X \arrow[r, "sc"] & ?Z \arrow[u, "sc"']
\end{tikzcd}$$

B':

$$\begin{tikzcd}
 & ?Y \\
?X \arrow[r, "sc"] \arrow[ur, "sc"] & ?Z \arrow[u, "sc"']
\end{tikzcd}$$

Clearly, $q' \sqsubseteq_m q$ and $q \sqsubseteq_m q'$. But $q \not\sqsubseteq_p q'$ nor $q' \not\sqsubseteq_p q$.

Even if we do not allow rdfs vocabulary in queries, the two notions are not equivalent. Consider two queries $q = (H, B)$ and $q' = (H', B')$, where $B = B'$, and the heads are as follows:

H:

$$\begin{tikzcd}
c \\
?X \arrow[u, "q"]
\end{tikzcd}$$

H':

$$\begin{tikzcd}
Y \\
?X \arrow[u, "q"]
\end{tikzcd}$$

where Y is a blank node, and $?X$ is a variable. Clearly, $q' \sqsubseteq_m q$ but $q' \not\sqsubseteq_p q$.

For queries without rdfs vocabulary and blank nodes we can still find examples for which the two containment notions disagree. Consider two queries $q = (H, B)$ and $q' = (H', B')$, where $B = B'$, and the heads are as follows:

H:

$$\begin{tikzcd}
 & ?Y \\
?X \arrow[r, "q"] & ?Z \arrow[u, "p"']
\end{tikzcd}$$

H':

$$\begin{tikzcd}
?Y \\
?Z \arrow[u, "p"']
\end{tikzcd}$$

In this case, $q' \sqsubseteq_m q$ but $q' \not\sqsubseteq_p q$.

We prove next that blank nodes in databases do not play any role in the containment problem, when queries do not have constraints. We need the following auxiliary notions.

Let $\mathcal{D}$ be a set of databases. We write that $q \sqsubseteq_p q'$ in $\mathcal{D}$ if and only if for all databases $D \in \mathcal{D}$, $\text{preans}(q, D) \subseteq_{iso} \text{preans}(q', D)$. We write that $q \sqsubseteq_m q'$ in $\mathcal{D}$ iff for all databases $D \in \mathcal{D}$, $\text{ans}(q', D) \models \text{ans}(q, D)$.

Proposition 5.4. *Let q, q' be queries without constraints and let $\mathcal{G}$ be the set of ground databases. Then*:

1. *$q' \sqsubseteq_m q$ in $\mathcal{G}$ if and only if $q' \sqsubseteq_m q$.*
2. *$q' \sqsubseteq_p q$ in $\mathcal{G}$ if and only if $q' \sqsubseteq_p q$.*

Proof.

1. The "if" direction is trivial. For "only if", assume it is not true. Then there is a database D such that $\text{ans}(q', D) \not\models \text{ans}(q, D)$ (*). Consider the database $D_g = \mu(D)$, where μ is the map sending blank nodes N of D to constants c_N. Then $D_g \models D$, and $\text{ans}(q', D_g) \models \text{ans}(q, D_g)$, that is, there is $\delta : \text{ans}(q, D_g) \to \text{ans}(q', D_g)$. Then we can build a map $\theta : \text{ans}(q, D) \to \text{ans}(q', D)$ defined as follows: $\theta(x) = x$ if x is a blank node in D, and $\theta(x) = \delta(x)$ elsewhere (URIs or blank nodes generated by Skolem functions). It is not difficult to show that θ is a map, yielding a contradiction with (*).
2. The "if" direction is trivial. For "only if", assume it is not true. Then there is a database D such that $\text{ans}(q', D) \not\subseteq \text{ans}(q, D)$. Consider the database $D_g = \mu(D)$, where μ is the map sending blank nodes N of D to constants c_N. Then $\text{ans}(q', D_g) \subseteq \text{ans}(q, D_g)$. Now, we replace each constant c_N in D_g with N, and it can be easily verified that $\text{ans}(q', D) \subseteq \text{ans}(q, D)$, a contradiction. □

5.2. Testing containment

In this section we study the notions of containment for queries without premises and without constraints.

Testing standard containment of queries without premises resembles containment of conjunctive relational queries. A characterization of entailment-based containment is more subtle. The next theorem gives characterizations for both notions.

Define $G_1 \models G_2$ for graphs G_1, G_2 containing variables, as $v(G_1) \models v(G_2)$, where v is a valuation sending the variables to fresh constants.

Theorem 5.5. *Consider the queries $q = (H, B)$ and $q' = (H', B')$. Then*:

1. *$q \sqsubseteq_p q'$ if and only if there is a substitution of variables θ such that* (a) $\theta(B') \subseteq \text{nf}(B)$ *and* (b) $\theta(H') \cong H$.
2. *$q \sqsubseteq_m q'$ if and only if there are substitutions $\theta_1, \ldots, \theta_n$ (of variables) such that* (a) $\theta_j(B') \subseteq \text{nf}(B)$ *and* (b) $\bigcup_j \theta_j(H') \models H$.

Proof.

1. (**If**) Let D be a database, and v a substitution with $v(H) \in \text{preans}(q, D)$, that is, $v(B) \subseteq \text{nf}(D)$. Hence $\text{nf}(v(B)) \subseteq \text{nf}(D)$. But it can be easily verified (by structural induction on derivation rules) that $v(\text{nf}(B)) \subseteq \text{nf}(v(B))$. Therefore, we obtain $v(\text{nf}(B)) \subseteq \text{nf}(D)$. Now, let $\mu = v(\theta(\ ))$, using condition (a) of the theorem, we obtain $v(\theta(B')) \subseteq \text{nf}(D)$, that is $\mu(H') \in \text{preans}(q', D)$. But condition (b) of the theorem states that $\theta(H') \cong H$. Then $v(\theta(H')) \cong v(H)$, hence, $\mu(H') \cong v(H)$. Therefore $q \sqsubseteq_p q'$.
   (**Only if**) Assume $q \sqsubseteq_p q'$. Consider the database $D = \mu(B)$, where $\mu(x) = x$ if x is a constant and $\mu(x) = c_x$ if x is a variable. Clearly, $\mu(H) \in \text{preans}(q, D)$, then there is $\mu'(H') \in \text{preans}(q', D)$ such that $\mu(H) \cong \mu'(H')$. Therefore, $\mu'(B') \subseteq \text{nf}(D)$. Now, let $\theta = \mu^{-1}(\mu'(\ ))$. It can be easily verified that θ satisfies conditions (a) and (b) of the theorem.
2. Because of Proposition 5.4, we need to show the statement for $q \sqsubseteq_m q'$ in $\mathcal{G}$, for $\mathcal{G}$ being the set of ground databases.
   (**If**) Let D be a ground database and $\text{ans}(q, D) = \bigcup v_i(H)$ and $\text{ans}(q', D) = \bigcup u_i(H')$ (the $v_i(H)$ and $u_i(H)$ are pre-answers), we will prove that there is a map $\omega : \text{ans}(q, D) \to \text{ans}(q', D)$. Since D is a ground database the pre-answers do not share blank nodes and therefore it is enough to prove that there are maps $\omega_i : v_i(H) \to \text{ans}(q', D)$, for each $v_i(H) \in \text{preans}(q, D)$. Consider the maps $\theta_1, \ldots, \theta_j$. For each j, $\theta_j(B') \subseteq \text{nf}(B)$, hence $v_i(\theta_j(B')) \subseteq v_i(\text{nf}(B)) \subseteq \text{nf}(D)$. Hence, $v_i(\theta_j(H')) \in \text{preans}(q', D)$. Therefore, $\bigcup_j v_i(\theta_j(H')) \subseteq \text{ans}(q', D)$. Let α be the map of condition (b), that is $\alpha(H) \subseteq \bigcup_j \theta_j(H')$. Notice that α only replace blank nodes, but the variables are preserved. Now let define ω_i as follows: $\omega_i(x) = \alpha(x)$ for each blank node x in $v(H)$, and $\omega_i(x) = x$ for each constant. From condition (b), it follows that $\omega_i(v_i(H)) \subseteq \bigcup_j v_i(\theta_j(H'))$, and hence $\omega_i : v_i(H) \to \text{ans}(q', D)$.
   (**Only if**) Consider the database $D_B = v(B)$, where v is the 1–1 valuation assigning x to a fresh constant a_x. By hypothesis, we have $\text{ans}(q', D_B) \models v(H)$. So, there are maps $v'_1, \ldots, v'_n : B' \to \text{nf}(D_B)$, such that $\bigcup_j v'_j(H') \models v(H)$. Now, applying v^{-1} to both sides of the expression, and using that (a) v^{-1} is 1–1 and works only over ground elements, and (b) considering variables resulting from the application of v^{-1} as ground elements, we have that $v^{-1}(\bigcup_j v'_j(H')) \models H$. Then $\bigcup_j v^{-1}(v'_j(H')) \models H$. Thus, let $\theta_j = v^{-1} \circ v'_j$, and we have conditions (1) and (2) of the theorem. □

We end the section by giving complexity bounds for the containment problem (for queries without premises).

Theorem 5.6. *Consider the queries $q = (H, B)$ and $q' = (H', B')$. Then*:

1. *Testing whether $q \sqsubseteq_p q'$ is NP-complete.*
2. *Testing whether $q \sqsubseteq_m q'$ is NP-complete.*

Proof.

1. NP-hardness: we can encode the problem of deciding $G \models G'$, where G, G' are simple graphs. From Theorem 2.9 this problem is NP-complete. We just construct the queries $q : (a, b, c) \leftarrow B$ and $q' : (a, b, c) \leftarrow B'$, where a, b, c are constants, and B, B' are obtained from G, G', respectively, by replacing blanks with variables. From Theorems 2.8 and 5.5 it can be easily verified that the two problems are equivalent, that is $q \subseteq_p q'$ (i.e., there is a substitution of variables θ such that $\theta(B') \subseteq B$) iff $G \models G'$ (i.e., there is a map from G' to G). Membership in NP follows directly from Theorem 5.5: just use θ as a certificate.
2. NP-hardness: notice that, for the queries q, q' given in the item 1 of this proof, Theorem 5.5(2) becomes: $q \subseteq_m q'$ iff there is a substitution θ_j such that $\theta_j(B') \subseteq \mathrm{nf}(B)$ (because q, q' have the same head). Therefore, for q, q' the two containment problems are equivalent, and we can use the encoding from the previous proof. For proving membership in NP just notice that, from the proof of Theorem 5.5(2), the number of maps θ_j is at most the number of triples of H. Then, the witness is the set $\{\theta_j\}$ and the map that witness the entailment of condition (b) of Theorem 5.5(2). □

5.3. Containment of queries with constraints

The results of Section 5.2 can be easily generalized to handle constraints. Next, we extend Theorem 5.5 to handle constraints.

Theorem 5.7. *Consider the queries $q = (H, B)$ and $q' = (H', B')$. Then:*

1. *$q \sqsubseteq_p q'$ if and only if there is a substitution of variables θ such that* (a) $\theta(B') \subseteq \mathrm{nf}(B)$ *and* (b) $\theta(H') \cong H$, *and* (c) $\theta(C') \subseteq C$.
2. *$q \sqsubseteq_m q'$ if and only if there are substitutions $\theta_1, \dots, \theta_n$ (of variables) such that* (a) $\theta_j(B') \subseteq \mathrm{nf}(B)$ *and* (b) $\bigcup_j \theta_j(H') \models H$, *and* (c) $\theta_j(C') \subseteq C$.

Proof. Given a query $q = (H, B, C)$, we define q_c as the query $(H, B_c, \emptyset)$, where B_c is obtained from B by adding a triple $(?X, \mathtt{is}, \mathtt{ground})$ (which we will refer to as a constraint triple) for each $?X \in C$.

Now, for a database D, we define D_c as D union a set including a triple $(c, \mathtt{is}, \mathtt{ground})$ for each constant c that appears in D. We define $\mathcal{D}_c$ as the set containing D_c for all databases D.

Notice that for all queries q: $\mathrm{ans}(q, D) = \mathrm{ans}(q_c, D_c)$ and $\mathrm{preans}(q, D) = \mathrm{ans}(q_c, D_c)$. Therefore we have that: $q \sqsubseteq_p q'$ iff $q_c \sqsubseteq_p q'_c$ in $\mathcal{D}_c$ and $q \sqsubseteq_m q'$ iff $q_c \sqsubseteq_m q'_c$ in $\mathcal{D}_c$.

Now, Theorem 5.5 also applies for $q_c \sqsubseteq_p q'_c$ in $\mathcal{D}_c$ and $q_c \sqsubseteq_m q'_c$ in $\mathcal{D}_c$. The new conditions (c) of both parts of the theorem make explicit that the maps of Theorem 5.5 should include the new constraint triples. □

5.4. Containment of queries with premises

In this section, we present a first study of containment for queries with premises. We tackle the containment problem in the realm of RDF graphs over non-interpreted vocabulary, that is, rdfs graphs are treated as simple graphs wherever they appear, that is in bodies, heads, and premises and databases. We refer to this class of queries as *simple queries*.

We prove that simple queries with premises can be transformed into union of queries without premises, which yields a characterization and complexity bounds for testing entailment. This result does not hold if the semantics of rdfs vocabulary is considered, even if premises are simple RDF graphs. The results here can also be extended to include constraints using the trick in the proof of Theorem 5.7, but for the sake of simplicity, we omit constraints in this section.

We start by proving that, when testing $q \sqsubseteq_p q'$ or testing $q \sqsubseteq_m q'$, a mild extension of Theorem 5.5 can handle a premise in q'.

Theorem 5.8. *Consider the simple queries $q = (H, B, \emptyset)$ and $q' = (H', B', P')$. Then:*

1. *$q \sqsubseteq_p q'$ if and only if there is a substitution (of variables and blank nodes) θ such that* (a) $\theta(B') \subseteq P' + B$ *and* (b) $\theta(H') \cong H$.
2. *$q \sqsubseteq_m q'$ if and only if there are substitutions $\theta_1, \dots, \theta_n$ (of variables) such that* (a) $\theta_j(B') \subseteq P' + B$ *and* (b) $\bigcup_j \theta_j(H') \models H$.

Proof. The proof mirrors closely that of Theorem 5.5 and we write it down for the sake of completeness.

1. (**If**) Let D be a database, and v a substitution with $v(H) \in \mathrm{preans}(q, D)$, that is, (a) $v(B) \subseteq D$. Let $\mu = v(\theta(\,))$. From (a) and (1) we obtain $v(\theta_1(B')) \subseteq P' + D$, and hence $\mu(H') \in \mathrm{preans}(q', D)$. But (2) states that $\theta_1(H') \cong H$. Then $v(\theta_1(H')) \cong v(H)$, hence, $\mu(H') \cong v(H)$. Therefore $q \sqsubseteq_p q'$.
(**Only if**) Assume $q \sqsubseteq_p q'$. Consider the database $D = \mu(B)$, where $\mu(x) = x$ if x is a constant and $\mu(x) = c_x$ if x is a variable. Clearly, $\mu(H) \in \mathrm{preans}(q, D)$, then there is $\mu'(H') \in \mathrm{preans}(q', D)$ such that $\mu(H) \cong \mu'(H')$. Therefore, $\mu'(B') \subseteq P' + D$. Now, let $\theta = \mu^{-1}(\mu'(\,))$. It can be easily verified that θ satisfies conditions (a) and (b) of the theorem.

2. Because of Proposition 5.4, we need to show the statement for $q \sqsubseteq_m q'$ in $\mathcal{G}$, for $\mathcal{G}$ being the set of ground databases.
(**If**) Let D be a ground database and $\text{ans}(q, D) = \bigcup v_i(H)$ and $\text{ans}(q', D) = \bigcup u_i(H')$ (the $v_i(H)$ and $u_i(H)$ are pre-answers), we will prove that there is a map $\omega : \text{ans}(q, D) \to \text{ans}(q', D)$. Since D is a ground database the pre-answers do not share blank nodes and therefore it is enough to prove that there are maps $\omega_i : v_i(H) \to \text{ans}(q', D)$, for each $v_i(H) \in \text{preans}(q, D)$. Consider the maps $\theta_1, \dots, \theta_j$. For each j, $\theta_j(B') \subseteq P' + B$, hence $v_i(\theta_j(B')) \subseteq P' + v_i(B) \subseteq P' + D$. Hence, $v_i(\theta_j(H')) \in \text{preans}(q', D)$. Therefore, $\bigcup_j v_i(\theta_j(H')) \subseteq \text{ans}(q', D)$. Let α be the map of condition (b), that is $\alpha(H) \subseteq \bigcup_j \theta_j(H')$. Notice that α only replace blank nodes, but the variables are preserved. Now let define ω_i as follows: $\omega_i(x) = \alpha(x)$ for each blank node x in $v(H)$, and $\omega_i(x) = x$ for each constant. From condition (b), it follows that $\omega_i(v_i(H)) \subseteq \bigcup_j v_i(\theta_j(H'))$, and hence $\omega_i : v_i(H) \to \text{ans}(q', D)$.
(**Only if**) Consider the database $D_B = v(B)$, where v is the 1–1 valuation assigning to each variable X a fresh constant a_X. By hypothesis, we have $\text{ans}(q', D_B) \models v(H)$. So, there are maps $v'_1, \dots, v'_n : B' \to \text{nf}(D_B)$, such that $\bigcup_j v'_j(H') \models v(H)$. Now, applying v^{-1} to both sides of the expression, and using that (a) v^{-1} is 1–1 and works only over ground elements, and (b) considering variables resulting from the application of v^{-1} as ground elements, we have that $v^{-1}(\bigcup_j v'_j(H')) \models H$. Then $\bigcup_j v^{-1}(v'_j(H')) \models H$. Thus, let $\theta_j = v^{-1} \circ v'_j$, and we have conditions (1) and (2) of the theorem. □

Next, we prove that, if the semantics rdfs vocabulary is not taken into account, a query with premises can be transformed into a union of queries without premises.

Proposition 5.9. *Let $q = (H, B, P)$ be a simple query. And let Ω_q be the set of queries defined as:*

$$\Omega_q = \{q_\mu = (\mu(H), \mu(B - R), \emptyset) \mid \mu : R \to P \text{ and } R \subseteq B \text{ and } \mu(B - R) \text{ has no blanks}\}.$$

Then, $q = \bigcup_{q_\mu \in \Omega_q} q_\mu$, that is, for all databases D, $\text{ans}(q, D) = \bigcup_{q_\mu \in \Omega_q} \text{ans}(q_\mu, D)$.

Proof. ($\subseteq$) Consider a database D, and $v(H) \in \text{preans}(q, D)$. Then $v(B) \subseteq D + P$. Now, we can split B into R and $B - R$, such that $v(R) \subseteq P$ and $v(B - R) \subseteq D$. Now let μ be the map obtained by restricting the domain of v to the variables in R. And let w be the map obtained by restricting the domain of v to variables in $B - R$. If $\mu(B - R)$ has a blank, then there is a variable x such that $\mu(x)$ is a blank in P and also $\mu(x)$ is a blank in D, contradicting the restriction that blanks of premises are disjoint with blanks of databases. Therefore q_μ is a query in the set given in the theorem. In addition, it can be easily verified that $w(\mu(B - R)) \in D$ and hence $w(\mu(H)) \in \text{preans}(q_\mu, D)$, and hence $v(H) \in \text{preans}(q_\mu, D)$.

($\supseteq$) Consider a query q_u, a database D and a pre-answer $v(\mu(H)) \in \text{preans}(q_u, D)$. Then $v(\mu(B - R)) \in D$ and also from the definition of q_μ we have $\mu(R) \in P$. Then, we have $v(\mu(R)) \in P$ (because $v(\mu(R)) = \mu(R)$). Therefore, $v(\mu(B)) \subseteq P + D$, hence $v(\mu(H)) \in \text{preans}(q, D)$. We can encode containment of classical tableau into queries without premises. □

Example 5.10. As an example, the answer returned by the query q defined by:

q: $(?X, p, ?Y) \leftarrow (?X, q, ?Y), (?Y, t, s)$, with $P = \{(a, t, s), (b, t, s)\}$

is the same as the union of the answers returned by the following queries:

q_1: $(?X, p, a) \leftarrow (?X, q, a)$, with $P_1 = \emptyset$,
q_1: $(?X, p, b) \leftarrow (?X, q, b)$, with $P_2 = \emptyset$,
q_3: $(?X, p, ?Y) \leftarrow (?X, q, ?Y), (?Y, t, s)$, with $P_3 = \emptyset$.

Proposition 5.11. *Let q_1, q_2 be queries without premises, and let $q' = (H', B', P')$. Then:*

1. *$(q_1 \cup q_2) \sqsubseteq_p q'$ iff $q_1 \sqsubseteq_p q'$ and $q_2 \sqsubseteq_p q'$.*
2. *$(q_1 \cup q_2) \sqsubseteq_m q'$ iff $q_1 \sqsubseteq_m q'$ and $q_2 \sqsubseteq_m q'$.*

Proof.

1. It follows directly from the definition of $\sqsubseteq_p$.
2. (If) Then $q_1 \sqsubseteq_m q'$ in $\mathcal{G}$ and $q_2 \sqsubseteq_m q'$ in $\mathcal{G}$, where $\mathcal{G}$ is the set of ground databases. Then for all constant databases D, we have maps: $\mu_1 : \text{ans}(q_1, D) \to \text{ans}(q', D)$ and $\mu_2 : \text{ans}(q_2, D) \to \text{ans}(q', D)$. But, because D is constant $\text{ans}(q_1, D)$ and $\text{ans}(q_2, D)$ do not share blanks, and hence $\mu_1 \cup \mu_2 : (\text{ans}(q_1, D) \cup \text{ans}(2, D)) \to \text{ans}(q', D)$. Hence, $(q_1 \cup q_2) \sqsubseteq_m q'$ in $\mathcal{G}$. The conclusion follows from Proposition 5.4.
The “only if” part is direct. □

Propositions 5.9 and 5.11 yield the following algorithm for testing the containment $q \sqsubseteq_p q'$ (respectively $q \sqsubseteq_m q'$). First, we transform q into Ω_q and then for each query $q_\mu \in \Omega_q$ we use Theorem 5.8 to test if $q_\mu \sqsubseteq_p q'$ (respectively $q_\mu \sqsubseteq_m q'$). We end this section by given complexity bounds for the containment problems.

Theorem 5.12. *Let $q = (H, B, P)$ and $q' = (H', B', P')$ be simple queries.*

1. *Testing whether $q \sqsubseteq_p q'$ is NP-hard and in $\Pi_2 P$.*
2. *Testing whether $q \sqsubseteq_m q'$ is NP-hard and in $\Pi_2 P$.*
3. *If $P = \emptyset$ then both testing $q \sqsubseteq_p q'$ and testing $q \sqsubseteq_m q'$ are NP-complete.*

Proof.

1. NP-hardness: follows from the same encoding as in the proof of Theorem 5.6(1). From Proposition 5.9 and Theorem 5.5, this can be solved by an *NP* machine with an *NP* oracle. The certificate is a query q_μ in the set given by Proposition 5.9, we use the oracle to test whether $q_\mu \not\sqsubseteq_p q'$. Hence the complement problem is in $\mathrm{NP}^{\mathrm{NP}}$, and the problem is in $\mathrm{coNP}^{\mathrm{NP}}$.
2. Similar to the previous proof.
3. Follows directly from Theorem 5.8: just use θ as a certificate to prove membership in NP. For NP-hardness the encoding given in the proof of Theorem 5.6(1) works here. □

6. Complexity of query answering

6.1. Computing matchings

In order to understand the complexity of computing the set of matchings for a query over a database, we consider the simpler problem of testing emptiness of the query answer set.

1. Query complexity version: For a fixed database D, given a query q, is $q(D)$ non-empty?
2. Data complexity version: For a fixed query q, given a database D, is $q(D)$ non-empty?

Theorem 6.1. *The evaluation problem is NP-complete for the query complexity version, and polynomial for the data complexity version.*

Proof. Reduction of 3SAT to the problem of evaluating a conjunctive query over a database. Membership in NP follows immediately.

Data complexity version: This follows from the fact that the number of potential matchings of the body of q in $\mathrm{nf}(D)$ is bounded by the number of subgraphs of $\mathrm{nf}(D)$ of size q, and the fact that the size of $\mathrm{nf}(D)$ is polynomial in D. □

From the proof of Theorem 6.1 follows that the size of the set of answers of a query q issued against a database D is bounded by $|D|^{|q|}$, where $|D|$ is the size of the normal form of the database (number of triples) and $|q|$ is the number of symbols in the query.

6.2. Redundancy elimination

We give some observations on redundancies in queries, databases and set of answers.

It is desirable and possible to have queries with lean heads. Otherwise, the answer generated will have redundancies which could have been avoided.

On the contrary, it is not always possible to have lean graphs in body of queries. For example, consider the query $q = (H, B, \emptyset)$, where $H = (?Course, \text{related}, \text{“DB”})$ and $B = (?Dept, \text{offers}, \text{“DB”}), (?Dept, \text{offers}, ?Course)$. B is not lean and is equivalent to the lean graph $B' = (?Dept, \text{offers}, \text{“DB”})$. It turns out that there is no query equivalent to q with body B' (using any notion of equivalence).

Even having lean databases and queries with lean heads and bodies does not avoid redundancies in the answer set. Consider the lean graph G_2 in Example 3.8, and the query $(?Z, p, ?U) \leftarrow (?Z, p, ?U)$. The answer set is G_1 which is not lean.

Answers to queries in RDF usually have redundancies. Ideally, the answer set $\mathrm{ans}(q, D)$ should reduce these redundancies to the minimum, i.e. to an equivalent lean graph. The naive approach to eliminate redundancy in answers is to compute: (1) $\mathrm{ans}(q, D)$, and (2) a lean equivalent to $\mathrm{ans}(q, D)$. The next theorem shows that in the worst case there is no better approach.

Theorem 6.2. *Given a lean database D and a query q, to decide whether $\mathrm{ans}_\cup(q, D)$ is lean is coNP-complete (in the size of D).*

The theorem follows from the fact that there is a query that computes the identity and from Theorem 3.12.

For merge-semantics redundancy elimination can be done much more efficiently:

Theorem 6.3. *Given a lean database D and a query q, deciding whether* $\mathrm{ans}_+(q, D)$ *is lean can be done in polynomial time in the size of D.*

Proof. Let $A = \mathrm{ans}_+(q, D)$ and let us refer to maps from single answers to A as *single maps*. The key observation is that, because single answers do not share variables in merge-semantics, maps $\mu : A \to A$ are exactly unions of single maps $\mu_j : G_j \to A$ for each G_j single answer. (Note that in the case of union-semantics the union of the μ_j would not be a function.)

Thus an algorithm for finding a proper map $\mu : A \to A$ only needs to compute single maps and check whether (1) at least a single map is proper, or (2) two of them share a blank node in their range. This can be done in time polynomial on the set of single maps, whose size is polynomial in the size of D. Thus the complete test can be done in polytime. □

7. Conclusions and future work

We have shown in this paper that the RDF data model poses new challenges to the development of query languages, by formalizing and studying some fundamental problems introduced by this model. The model presented establishes a good common base to formally study and compare functionalities, features and limitations of RDF query languages. We think that this streamlined formalization also establishes a solid ground from where to study theoretical properties of extensions of the model. In fact, there are several extension either proposed or currently in use in working query languages for RDF which deserve theoretical analysis. Among the most relevant we can mention connectedness, reachability, paths, recursion, composition, subquerying, extended constraints, aggregation and views.

Acknowledgments

The authors were supported by: Fondecyt 1070348 (Gutierrez); Fondecyt 1080672 (Hurtado); Conicyt Ph.D. Scholarship (Pérez).

References

[1] A.V. Aho, M.R. Garey, J.D. Ullman, The transitive reduction of a directed graph, SIAM J. Comput. 1 (1972) 131–137.
[2] S. Abiteboul, R. Hull, V. Vianu, Foundations of Databases, Addison–Wesley Publishing Co., 1995.
[3] F. Alkhateeb, Querying RDF(S) with regular expressions, Ph.D. thesis, Université Joseph Fourier, Grenoble, France, 2008.
[4] R. Angles, C. Gutierrez, The expressive power of SPARQL, in: ISWC 2008, in: Lecture Notes in Comput. Sci., vol. 5318, 2008, pp. 114–129.
[5] J. Baget, RDF entailment as graph homomorphism, in: ISWC 2005, pp. 82–96.
[6] J. Broeskstra, A. Kampman, SeRQL: A second generation RDF query language, in: SWAD-Europe Workshop on Semantic Web Storage and Retrieval, 2003, pp. 13–14.
[7] A. Bonner, Hypothetical Datalog: Complexity and expressibility, in: Proceedings of the Second International Conference on Database Theory, ICDT 1988, pp. 144–160.
[8] J. de Bruijn, S. Tessaris, E. Franconi, Logical reconstruction of RDF and ontology languages, in: Proc. PPSWR-05, 2005, pp. 65–71.
[9] J. de Bruijn, E. Franconi, S. Tessaris, Logical reconstruction of normative RDF, in: Proceedings of the Workshop on OWL Experiences and Directions, OWLED 2005.
[10] Ch. Chekuri, A. Rajaraman, Conjunctive query containment revisited, in: ICDT 1997, pp. 56–70.
[11] T. Furche, B. Linse, F. Bry, D. Plexousakis, G. Gottlob, RDF querying: Language constructs and evaluation methods compared, in: Reasoning Web, 2006, pp. 1–52.
[12] P. Haase, J. Broekstra, A. Eberhart, R. Volz, A comparison of RDF query languages, in: International Semantic Web Conference, 2004, pp. 502–517.
[13] Y. Kanza, W. Nutt, Y. Sagiv, Queries with incomplete answers over semistructured data, in: PODS 1999, pp. 227–236.
[14] K. Kochut, M. Janik, SPARQLeR: Extended SPARQL for semantic association discovery, in: ESWC 2007, in: Lecture Notes in Comput. Sci., vol. 4519, 2007, pp. 145–159.
[15] R. Fagin, Ph.G. Kolaitis, L. Popa, Data exchange: getting to the core, ACM Trans. Database Syst. 30 (1) (2005) 174–210.
[16] D.M. Gabbay, U. Reyle, N-Prolog: an extension of Prolog with hypothetical implications. I, J. Logic Program. 1 (4) (1984) 319–355.
[17] A. Gomez-Perez, M.C. Suarez-Figueroa, Results of taxonomic evaluation of RDF(S) and DAML+OIL ontologies using RDF(S) and DAML+OIL validation tools and ontology platforms import services, in: 2nd Workshop on Evaluation of Ontology-based Tools, EON2003.
[18] M. Grohe, T. Schwentick, L. Segoufin, When is the evaluation of conjunctive queries tractable?, in: STOC 2001, pp. 657–666.
[19] R.V. Guha, rdfDB query language, http://www.guha.com/rdfdb/query.html.
[20] G. Gottlob, Computing cores for data exchange: New algorithms and practical solutions, in: PODS 2005, pp. 148–159.
[21] G. Gottlob, A. Nash, Data exchange: Computing cores in polynomial time, in: PODS 2006, pp. 40–49.
[22] C. Gutierrez, C. Hurtado, A. Mendelzon, Foundations of Semantic Web databases, in: PODS 2004, pp. 95–106.
[23] C. Gutierrez, Normal forms for connectedness in categories, Ann. Pure Appl. Logic 108 (2001) 237–247.
[24] J. Hayes, C. Gutierrez, Bipartite graphs as intermediate model for RDF, in: ISWC 2004, pp. 47–61.
[25] P. Hell, J. Nesetril, The core of a graph, Discrete Math. 109 (1992) 117–126.
[26] H.J. ter Horst, Completeness, decidability and complexity of entailment for RDF schema and a semantic extension involving the OWL vocabulary, J. Web Sem. 3 (2–3) (2005) 79–115.
[27] G. Karvounarakis, S. Alexaki, V. Christophides, D. Plexousakis, M. Scholl, RQL: A declarative query language for RDF, in: WWW 2002, pp. 592–603.
[28] A. Magkanaraki, et al., Ontology storage and querying, Technical Report No. 308, Foundation for Research and Technology Hellas, Institute of Computer Science, Information System Laboratory, April 2002.

[29] D. Marin, RDF formalization, Technical Report, TR/DCC-2006-8, Universidad de Chile, Santiago de Chile, http://www.dcc.uchile.cl/~cgutierr/ftp/draltan.pdf, 2004.
[30] L. Miller, A. Seaborne, A. Reggiori, Three implementations of SquishQL, a simple RDF query language, in: ISWC 2002, pp. 399–403.
[31] S. Muñoz, J. Pérez, C. Gutierrez, Minimal deductive systems for RDF, in: ESWC 2007, pp. 53–67.
[32] M. Nilsson, W. Siberski (Eds.), RDF query exchange language (QEL), http://edutella.jxta.org/spec/qel.html.
[33] Y. Papakonstantinou, V. Vassalos, Query rewriting for semistructured data, in: SIGMOD 1999, pp. 455–466.
[34] J. Pérez, M. Arenas, C. Gutierrez, Semantics and complexity of SPARQL, in: ISWC 2006, in: Lecture Notes in Comput. Sci., vol. 4273, 2006, pp. 30–43.
[35] J. Pérez, M. Arenas, C. Gutierrez, nSPARQL: A navigational language for RDF, in: ISWC 2008, in: Lecture Notes in Comput. Sci., vol. 5318, 2008, pp. 66–81.
[36] R. Pichler, A. Polleres, F. Wei, S. Woltran, dRDF: Entailment for domain-restricted RDF, in: ESWC 2008, pp. 200–214.
[37] M. Dean, G. Schreiber (Eds.), OWL web ontology language reference, W3C candidate recommendation, 18 August 2003.
[38] E. Prud'hommeaux, B. Grosof, RDF query rules: A framework and survey, http://www.w3.org/2001/11/13-RDF-Query-Rules/.
[39] M. Sintek, S. Decker, TRIPLE—A query, inference, and transformation language for the Semantic Web, in: ISWC 2002, pp. 364–378.
[40] M. Yannakakis, Algorithms for acyclic database schemes, in: VLDB 1981, pp. 82–94.
[41] G. Yang, M. Kifer, On the semantics of anonymous identity and reification, in: ODBASE 2002, pp. 1047–1066.
[42] M. Vardi, The complexity of relational query languages (extended abstract), in: STOC 1982, pp. 137–146.
[43] R. Fikes, P. Hayes, I. Horrocks (Eds.), DAML query language (DQL), abstract specification, DAML Joint Committee, April 2003.
[44] O. Lassila, R. Swick (Eds.), Resource description framework (RDF) model and syntax specification, working draft, W3C, 1998.
[45] Patrick Hayes (Ed.), RDF semantics, W3C recommendation, 10 February 2004.
[46] Dan Brickley, R.V. Guha (Eds.), RDF vocabulary description language 1.0: RDF schema, W3C working draft, 23 January 2003.
[47] G. Klyne, J.J. Carroll (Eds.), RDF concepts and abstract syntax, W3C recommendation, 10 February 2004.
[48] F. Manola, E. Miller (Eds.), RDF primer, W3C recommendation, 10 February 2004.
[49] E. Prud'hommeaux, A. Seaborne (Eds.), SPARQL query language for RDF, W3C recommendation, 15 January 2008, http://www.w3.org/TR/rdf-sparql-query/.
[50] W3C Semantic Web Activity, http://www.w3.org/2001/sw/.