Habilitation à Diriger des Recherches

Spécialité : ***Informatique***

Université Paris-Sud

Knowledge Representation meets Databases

for the sake of ontology-based data management

par

François Goasdoué

Marraine :

Marie-Christine Rousset, Professeur, Université de Grenoble

Rapporteurs :

Alexander Borgida, Professeur, Rutgers University, États-Unis d'Amérique
Vassilis Christophides, Professeur, ICS-FORTH Crète, Grèce
Pierre Marquis, Professeur, Université de Lens

Examinateurs :

Serge Abiteboul, Directeur de recherche, INRIA Saclay Île-de-France
Michel Chein, Professeur, Université de Montpellier (Président du jury de soutenance)
Philippe Dague, Professeur, Université Paris-Sud

Abstract

This Habilitation thesis outlines my research activities carried out as an Associate Professor at Université Paris-Sud and Inria Saclay Île-de-France. During this period, from 2003 to early 2012, my work was – and still is – at the interface between *Knowledge Representation* and *Databases*. I have mainly focused on *ontology-based data management* using the Semantic Web data models promoted by W3C: the *Resource Description Framework (RDF)* and the *Web Ontology Language (OWL)*. In particular, my work has covered (i) the design, (ii) the optimization, and (iii) the decentralization of ontology-based data management techniques in these data models. This thesis briefly reports on the results obtained along these lines of research.

Contents

Chapter 1

Introduction

Data management is a longstanding research topic in *Knowledge Representation* (KR), a prominent discipline of *Artificial Intelligence* (AI), and – of course – in *Databases* (DB).

Till the end of the 20^{th} century, there have been few interactions between these two research fields concerning data management, essentially because they were addressing it from different perspectives. KR was investigating data management according to human cognitive schemes for the sake of intelligibility, e.g., using *Conceptual Graphs* [CM08] or *Description Logics* [BCM$^+$03], while DB was focusing on data management according to simple mathematical structures for the sake of efficiency, e.g., using the *relational model* [AHV95] or the *eXtensible Markup Language* [AMR$^+$12].

In the beginning of the 21^{st} century, these ideological stances have changed with the new era of *ontology-based data management* [Len11]. Roughly speaking, ontology-based data management brings data management one step closer to end-users, especially to those that are not computer scientists or engineers. It basically revisits the traditional architecture of database management systems by decoupling the models with which data is exposed to end-users from the models with which data is stored. Notably, ontology-based data management advocates the use of conceptual models from KR as human intelligible front-ends called *ontologies* [Gru09], relegating DB models to back-end storage.

The *World Wide Web Consortium* (W3C) has greatly contributed to ontology-based data management by providing *standards* for handling data through ontologies, the two *Semantic Web* data models. The first standard, the *Resource Description Framework* (RDF) [W3Ca], was introduced in 1998. It's a graph data model coming with a very simple ontology language, *RDF Schema*, strongly related to description logics. The second standard, the *Web Ontology Language* (OWL) [W3Cd], was introduced in 2004. It's actually a family of well-established description logics with varying expressivity/complexity tradeoffs.

The advent of RDF and OWL has rapidly focused the attention of academia and industry on *practical* ontology-based data management. The research community has undertaken this challenge at the highest level, leading to pioneering and compelling contributions in top venues on Artificial Intelligence (e.g., AAAI, ECAI, IJCAI, and KR), on Databases (e.g., ICDT/EDBT, ICDE, SIGMOD/PODS, and VLDB), and on the Web (e.g., ESWC, ISWC, and WWW). Also, open-source and commercial software providers are releasing an ever-growing number of tools allowing effective RDF and OWL data management (e.g., Jena, ORACLE 10/11g, OWLIM, Protégé, RDF-3X, and Sesame).

Last but not least, large societies have promptly adhered to RDF and OWL data management (e.g., library and information science, life science, and medicine), sustaining and begetting further efforts towards always more convenient, efficient, and scalable ontology-based data management techniques.

This HDR thesis outlines my contributions to ontology-based data management using RDF and OWL. The reported results were obtained from 2003 to early 2012, as an Associate professor at Université Paris-Sud (LRI, UMR CNRS 8623, Artificial Intelligence and Inference Systems team) and Inria (Saclay – Île-de-France, GEMO team 2003–2009, LEO team 2009–2012, and OAK team 2012–now). During this period, I have mainly investigated the *design*, the *optimization*, and the *decentralization* of ontology-based data management *techniques* for RDF and OWL. Each of these three lines of research has a dedicated Chapter in this thesis, summarizing a main contribution as well as ongoing work with promising preliminary results. For each, I briefly report on the context, motivations, and contributions of the study, before providing some intuitions and examples about the devised techniques. More specifically, the thesis is organized as follows.

Chapter 2 (Preliminaries). I introduce the basics of RDF and of the DL-lite family of description logics which underlies the OWL dialect dedicated to the management of large datasets: OWL2 QL. I describe for each the *data model*, *query language*, and prevalent *techniques* for the traditional data management tasks of *consistency checking* and *query answering*.

Chapter 3 (Design). The main contribution I present is *robust module-based data management in DL-lite*. Module-based data management amounts to handling data using an ontology – a *module* – that derives from that of a preexisting ontology-based data management application. I summarize the results from [GR10, GR12], which introduce the novel notion of *robust* module and show how to use it to enhance data integrity and to complement the answers to queries in ontology-based data management applications.

The ongoing work I present next is the design of *RDF query answering techniques that are robust to graph*[1] *updates*. I summarize the results from [GMR12b, GMR12a], which build on the prevalent saturation-based query answering technique and on the alternative reformulation-based query answering technique. A saturation maintenance technique is designed for the former to limit the necessary re-computation efforts upon graph updates, while the latter – de facto robust to updates – is extended to a larger fragment of RDF than those investigated in the literature.

Chapter 4 (Optimization). The main contribution I present is *view selection for efficient RDF query answering*. It amounts to tuning an RDF data management system to users' or applications' needs modeled as a query workload. The idea is to pre-compute and store the results for some *automatically* selected queries – the *views* – in order to minimize a combination of query processing, view storage, and view maintenance upon update costs. I summarize the results from [GKLM10b, GKLM10a, GKLM11a], which build on a state-of-the-art view selection technique for the relational data model. In particular, the view selection technique devised for RDF supports both saturation- and reformulation-based query answering, depending on how views are materialized.

The ongoing work I present next is a first step towards *efficient query answering against XML documents with RDF annotations*. I summarize the results from [GKK$^+$11a, GKK$^+$12], which combine the XML and RDF data models and query languages into a uniform XML-RDF hybrid setting for managing annotated documents. In particular, we want to study to which extent query answering can be optimized by using at the same time the structural XML constraints (expected tree shape of the documents) and the semantic RDF constraints (expected ontological descriptions) expressed in queries.

Chapter 5 (Decentralization). The main contribution I present is *peer-to-peer data management for RDF and DL-lite*. In such systems, every peer manages its ontology and data, and can also establish semantic correspondences called *mappings* with peers having similar interests. This gives rise to a distributed data management system, in which it becomes possible to perform global data management tasks. I summarize the results from [AGR07, AAC$^+$08, AGR09], which build on decentralized consequence finding in propositional logic [ACG$^+$05a, ACG$^+$06, AG09]. Notably, these peer-to-peer systems are fully decentralized and scale to more than a thousand peers.

The ongoing work I present next is a first step towards *RDF data management in a cloud*. A cloud is a place where one rents virtual machines, disk space, and services (e.g., database access), and then pays as she uses them. I summarize the results from [BGKM12], which investigate RDF query answering in the Amazon cloud w.r.t. efficiency and monetary costs.

Chapter 6 (Conclusion and perspectives). Finally, I present my forthcoming work, which mostly corresponds to challenges for enhancing the state-of-the-art data management techniques, or for enabling new valuable ontology-based data management tasks.

[1] Graph is the RDF term for a knowledge base in AI or a database in DB.

Chapter 2

Preliminaries

I recall here the basics of the Resource Description Framework (Section 2.1) and of the DL-lite family of Description Logics (Section 2.2). DL-lite is strongly related to data management in OWL, as it includes the DL-lite$_\mathcal{R}$ description logic, which underlies the OWL dialect dedicated to the management of large datasets: OWL2 QL. I also give a brief overview of the expressivity shared by RDF and DL-lite, and by their respective query languages (Section 2.3). This chapter compiles material published in [GMR12b, GMR12a, GR10, GR12].

2.1 The Resource Description Framework (RDF)

RDF [W3Ca] is a graph data models that has been recommended by W3C since 1998. It allows defining *graphs* (Section 2.1.1) that can be queried with the *SPARQL Protocol And RDF Query Language* [W3Cc] (Section 2.1.2). This language, SPARQL in short, has been recommended by W3C since 2008. The prevalent technique for answering SPARQL queries against graphs is *saturation-based query answering* (Section 2.1.3).

2.1.1 Graphs

A *graph* is a set of *triples* of the form `s p o .` (the final dot preceded by a white space belongs to the normative triple syntax). A triple states that its *subject* `s` has the corresponding *property* `p`, and the value of that property is the *object* `o`. Given a set U of *Uniform Resource Identifiers*[1] (URIs), a set L of literals (constants), and a set B of blank nodes (*unknown* URIs or literals), such that U, B and L are pairwise disjoint, a triple is *well-formed* whenever its subject belongs to $U \cup B$, its property belongs to U, and its object belongs to $U \cup B \cup L$. In the following, I only consider well-formed triples.

Blank nodes are essential features of RDF allowing the support of *incomplete information*. For instance, one can use a blank node $\_{:}b_1$ to state that the country of $\_{:}b_1$ is *France* while the city of the same $\_{:}b_1$ is *Brest*. Many such blank nodes can co-exist within a graph, e.g., one may also state that the country of $\_{:}b_2$ is *Romania* while the city of $\_{:}b_2$ is *Timişoara*; at the same time, the population of *Timişoara* can be said to be an unspecified constant $\_{:}b_3$.

Notations I use `s`, `p`, `o` and $\_{:}b$ in triples (possibly with subscripts) as placeholders. That is, `s` stands for values in $U \cup B$, `p` stands for values in U, `o` represents values from $U \cup B \cup L$, and $\_{:}b$ denotes values in B. Strings between quotes as in "*string*" denote literals. Finally, the set of values (URIs, blank nodes, literals) of a graph `G` is denoted `Val(G)`.

Figure 2.1 shows how to use triples to describe resources; from now on, I use the name rdf for the normative RDF namespace[2] when writing the URIs of classes and properties comprised in the RDF standard.

[1] Uniform Resource Identifiers provide naming schemes for referring to resources using keys, in the usual database sense.

[2] A namespace is a URI used to group resources. When a namespace is given a `name`, a `resource` within that namespace can be simply written `name:resource`. For instance, `lri:iasi` refers to the resource `iasi` within the namespace `lri`; it corresponds to the URI `http://www.lri.fr/iasi` whenever the namespace `http://www.lri.fr/` is named `lri`.

Constructor	Triple
Class assertion	s rdf:type o .
Property assertion	s p o .

Figure 2.1: RDF statements.

A more intuitive representation of a *graph* can be drawn from its triples. Every (distinct) subject or object value is represented by a node labeled with this value. For each triple, there is a directed edge from the subject node to the object node, which is labeled with the property value.

Example 1 (Running example) *The two representations in Figure 2.2 are equivalent as they model the same graph* G*. The namespaces for user-defined classes and properties were omitted for the sake of readability. This graph describes the* Digital Object Identifier[3] doi_1 *that belongs to an unknown class, whose title (*hasTitle*) is "Complexity of Answering Queries Using Materialized Views", whose author (*hasAuthor*) is "Serge Abiteboul" and which has an unknown contact author (*hasContactA*). This paper is in the proceedings of (*inProceedingsOf*) an unknown resource whose name (*hasName*) is "PODS'98". Lastly, the URI* cikm2012 *is a conference and the property that associates names (*hasName*) has been created by "John Doe".*

$$\mathtt{G}= \begin{array}{l}\{\mathrm{doi}_1\ \mathrm{rdf{:}type}\ \_{:}b_0\ .,\ \mathrm{doi}_1\ \mathrm{hasTitle}\ \text{“}CAQUMV\text{”}\ .,\\ \mathrm{doi}_1\ \mathrm{hasAuthor}\ \text{“}SA\text{”}\ .,\ \mathrm{doi}_1\ \mathrm{hasContactA}\ \_{:}b_1\ .,\\ \mathrm{doi}_1\ \mathrm{inProceedingsOf}\ \_{:}b_2\ .,\ \_{:}b_2\ \mathrm{hasName}\ \text{“}PODS'98\text{”}\ .,\\ \mathrm{cikm2012}\ \mathrm{rdf{:}type}\ \mathrm{conference}\ .,\\ \mathrm{hasName}\ \mathrm{createdBy}\ \text{“}John\ Doe\text{”}\ .\}\end{array}$$

$\mathtt{G} =$

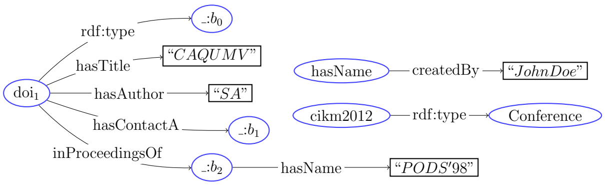


Figure 2.2: Alternative graph representations.

A valuable feature of RDF is RDF Schema (RDFS) that allows enhancing the descriptions in RDF graphs. An RDF Schema declares *semantic constraints* between the classes and the properties used in graphs. Figure 2.3 shows the allowed constraints and how to express them; from now on, I use the name rdfs for the normative RDFS namespace when writing the URIs of classes and properties comprised in the RDFS standard.

Constructor	Triple
Subclass constraint	s rdfs:subClassOf o .
Subproperty constraint	s rdfs:subPropertyOf o .
Domain typing constraint	s rdfs:domain o .
Range typing constraint	s rdfs:range o .

Figure 2.3: RDFS statements expressing semantic constraints between classes and properties.

Example 2 (Continued) *Consider next to the above graph* G*, a schema stating that poster papers (*posterCP*) together with the unknown class* $\_{:}b_0$ *are subclasses of conference papers (*confP*), which are scientific papers (*paper*). Moreover, titles (*hasTitle*), authors (*hasAuthor*), contact authors (*hasContactA*) – who are authors – are used to describe papers. Papers are also described by the conferences (*conference*) in whose proceedings (*inProceedingsOf*) they appear. Finally, names (*hasName*) describe conferences, and creators (*createdBy*) describe resources. The extended graph* G′ *of* G *corresponding to this schema is depicted in Figure 2.4.*

Entailment The W3C names *RDF entailment* the mechanism through which, based on the set of explicit triples and some *entailment rules* (to be described shortly), *implicit triples* are derived. I denote by $\vdash^i_{\mathrm{RDF}}$ *immediate entailment*, i.e., the process of deriving new triples through a single application of an entailment rule. More generally, a triple s p o . is entailed by a graph G, denoted G $\vdash_{\mathrm{RDF}}$ s p o . if and only if there is a sequence of applications of immediate entailment rules that leads from G to s p o ., where at each step of the entailment sequence, the triples previously entailed are also taken into account.

[3] http://www.doi.org

$$\mathtt{G}'=\mathtt{G}\cup\ \begin{array}{l}\{\text{posterCP rdfs:subClassOf confP .,}\ \ \text{\_:}b_0\ \text{rdfs:subClassOf confP .,}\ \ \text{confP rdfs:subClassOf paper .,}\\ \ \text{hasTitle rdfs:domain paper .,}\ \ \text{hasTitle rdfs:range rdfs:Literal .,}\ \ \text{hasAuthor rdfs:domain paper .,}\\ \ \text{hasAuthor rdfs:range rdfs:Literal .,}\ \ \text{hasContactA rdfs:subPropertyOf hasAuthor .,}\\ \ \text{inProceedingsOf rdfs:domain confP .,}\ \ \text{inProceedingsOf rdfs:range conference .,}\\ \ \text{hasName rdfs:domain conference .,}\ \ \text{hasName rdfs:range rdfs:Literal .,}\ \ \text{createdBy rdfs:range rdfs:Literal .}\}\end{array}$$

Figure 2.4: Extended graph for Example 2.

Graph saturation The immediate entailment rules allow defining the (finite) *saturation* (a.k.a. closure) of a graph $\mathtt{G}$, which is the graph, denoted $\mathtt{G}^\infty$, defined as the fixpoint of:

- $\mathtt{G}^0 = \mathtt{G}$
- $\mathtt{G}^\alpha = \mathtt{G}^{\alpha-1} \cup \{\mathtt{s\ p\ o\ .} \mid \mathtt{G}^{\alpha-1} \vdash^i_{\text{RDF}} \mathtt{s\ p\ o\ .}\}$

The saturation of a graph is unique (up to blank node renaming), and does not contain any implicit triples (they have been made explicit by saturation). An obvious connection holds between the triples entailed by a graph $\mathtt{G}$ and its saturation: $\mathtt{G} \vdash_{\text{RDF}} \mathtt{s\ p\ o\ .}$ if and only if $\mathtt{s\ p\ o\ .} \in \mathtt{G}^\infty$. It is worth noticing that RDF entailment is part of the RDF specification itself, and therefore the semantics of a graph is its saturation. From the RDF standard perspective, any graph $\mathtt{G}$ is equivalent to, and models, its saturation $\mathtt{G}^\infty$.

Immediate entailment rules I give here an overview of the different kinds of immediate entailment rules upon which RDF entailment relies.

A first kind of rule generalizes triples using blank nodes. In the running example, doi_1 rdf:type _:b_0 . entails _:b_3 rdf:type _:b_0 . Indeed, if doi_1 is an instance of _:b_0, then there exists an instance of _:b_0.

A second kind of rule derives entailed triples from the semantics of built-in classes and properties. E.g., RDF provides rdfs:Class whose semantics is the set of all built-in *and* user-defined classes, with the striking effect that rdfs:Class is an instance of itself. As a result, in the running example, doi_1 rdf:type _:b_0 . entails that _:b_0 is a class, i.e., _:b_0 rdf:type rdfs:Class .

Finally, the third kind of rule derives entailed triples from the constraints modeled in an RDF Schema. Some rules derive entailed RDFS statements, through the transitivity of class and property inclusions, and from inheritance of domain and range typing. For example, in the running example, _:b_0 rdfs:subClassOf confP . *and* confP rdfs:subClassOf paper . entail _:b_0 rdfs:subClassOf paper .; hasAuthor rdfs:domain paper . *and* hasContactA rdfs:subPropertyOf hasAuthor . entail hasContactA rdfs:domain paper . Some other rules derive entailed RDF statements, through the propagation values (URIs, blank nodes, and literals) from sub-classes and sub-properties to their super-classes and super-properties, and from properties to classes typing their domains and ranges. In the example, hasContactA rdfs:subPropertyOf hasAuthor . *and* doi_1 hasContactA _:b_1 . entail doi_1 hasAuthor _:b_1 .; hasAuthor rdfs:domain paper . *and* doi_1 hasAuthor _:b_1 . entail doi_1 rdf:type paper .

Graph consistency Graphs can be *inconsistent* only if they use *typed literals* instead of literals. Roughly speaking, RDF allows reusing pre-defined datatypes from XML Schema [W3Cf], a schema language for the XML tree data model. While literal typing is an RDF refinement of practical importance, it only introduces a very simple and specific form of inconsistency that is not of theoretical interest. Therefore, in this thesis, I only consider *consistent* graphs.

2.1.2 Queries

I consider the well-known subset of SPARQL consisting of *basic graph pattern* (BGP) queries. A BGP is a *set of triple patterns*, or triples in short. Each triple has a subject, property and object. Subjects and properties can be URIs, blank nodes or variables; objects can also be literals.

A boolean BGP query is of the form `ASK WHERE` $\{t_1, \ldots, t_\alpha\}$, while a non-boolean BGP query is of the form `SELECT` $\bar{x}$ `WHERE` $\{t_1, \ldots, t_\alpha\}$, where $\{t_1, \ldots, t_\alpha\}$ is a BGP; the variables $\bar{x}$ in the head of the query are called *distinguished variables*, and are a subset of the variables occurring in $t_1, \ldots, t_\alpha$.

Notations Without loss of generality, in the following I will use the traditional conjunctive query notation $q(\bar{x})$:- $t_1, \ldots, t_\alpha$ for both `ASK` and `SELECT` queries (for boolean queries, $\bar{x}$ is empty). I use x, y, and z (possibly with subscripts) to denote variables in queries. I denote by $\texttt{VarBl}(q)$ the set of variables *and* blank nodes occurring in the query q.

Query evaluation Given a query q and a graph $\texttt{G}$, the *evaluation of* q *against* $\texttt{G}$ is: $q(\texttt{G}) = \{\bar{x}_\mu \mid \mu : \texttt{VarBl}(q) \to \texttt{Val}(\texttt{G})$ is a total assignment s.t. $\{t_1, \ldots, t_\alpha\}_\mu \subseteq \texttt{G}\}$.

In the above, for any triple or set of triples O, I denote by O_μ the result of replacing every occurrence of a variable or blank node $e \in \texttt{VarBl}(q)$ in O by the value $\mu(e) \in \texttt{Val}(\texttt{G})$. If q is boolean, the empty answer set encodes false, while the non-empty answer set made of the empty tuple $\emptyset_\mu = \langle\rangle$ encodes true.

Observe that the normative evaluation *treats the blank nodes in a query as non-distinguished variables*. That is, one could consider equivalently queries without blank nodes or queries without non-distinguished variables.

Answer set of a query The evaluation of q against $\texttt{G}$ only uses $\texttt{G}$'s explicit triples, thus may lead to an incomplete answer set. The (correct) *answer set* of q against $\texttt{G}$ is obtained by the evaluation of q against $\texttt{G}^\infty$, denoted by $q(\texttt{G}^\infty)$.

Example 3 (Continued) *The following query asks for the authors of papers published in the proceedings of a conference somehow related to PODS'98:*

$$q(x)\text{:- } y_1 \text{ hasAuthor } x \text{ .}, y_1 \text{ inProceedingsOf } y_2 \text{ .}, y_2\ y_3 \text{ ``}PODS'98\text{'' .}$$

That query could be equivalently written into:

$$q(x)\text{:- } \_{:}b_0 \text{ hasAuthor } x \text{ .}, \_{:}b_0 \text{ inProceedingsOf } \_{:}b_1 \text{ .}, \_{:}b_1\ \_{:}b_2 \text{ ``}PODS'98\text{'' .}$$

The answer set of q *against* $\texttt{G}'$ *is:* $q(\texttt{G}'^\infty) = \{\langle\text{``}SA\text{''}\rangle, \langle\_{:}b_1\rangle\}$. *The answer* "$SA$" *results from the assignment* $\mu = \{y_1 \to \mathrm{doi}_1, x \to \text{``}SA\text{''}, y_2 \to \_{:}b_2, y_3 \to \mathrm{hasName}\}$, *while the answer* $\_{:}b_1$ *results from* $\texttt{G}' \vdash_{\mathrm{RDF}} \mathrm{doi}_1 \text{ hasAuthor } \_{:}b_1$. *and the assignment* $\mu = \{y_1 \to \mathrm{doi}_1, x \to \_{:}b_1, y_2 \to \_{:}b_2, y_3 \to \mathrm{hasName}\}$.

Note that evaluating q *against* $\texttt{G}'$ *leads to the incomplete answer set* $q(\texttt{G}') = \{\langle\text{``}SA\text{''}\rangle\} \subset q(\texttt{G}'^\infty)$.

2.1.3 Query answering

The prevalent technique for *answering queries* is *saturation-based query answering*. It's straightforward, since the answer set of a query is computed exactly as it is formally defined. The saturation of the queried graph is computed (using the entailment rules), so that the answer set of every query against the (original) graph is obtained by query evaluation against the saturation. The advantage of this approach is that it is easy to implement. Its disadvantages are that graph saturation needs time to be computed and space to store all the entailed triples; moreover, the saturation must be somehow recomputed upon every graph update.

Saturation-based query answering using relational database management systems Graphs turn out to be a special case of *incomplete* relational databases based on *V-tables* under the *open-world assumption* [IJ84, AHV95]. Such tables allow using variables in their tuples, and repeating a variable within a V-table allows expressing joins on unknown values. An important result on V-table querying is that the standard relational evaluation (which sees variables in V-tables as constants) computes the complete answer set of any conjunctive query [IJ84, AHV95]. From a practical viewpoint, this provides a possible way of answering BGP queries against graphs using standard relational database management systems (RDBMSs, in short).

From the above observations, a graph $\texttt{G}$ can be encoded into a (single) V-table $\texttt{Triple}(\texttt{s}, \texttt{p}, \texttt{o})$ storing the triples of $\texttt{G}$ as tuples, in which blank nodes become variables. Given a BGP query $q(\bar{x})$:- $\texttt{s}_1\ \texttt{p}_1\ \texttt{o}_1$., $\ldots$, $\texttt{s}_n\ \texttt{p}_n\ \texttt{o}_n$., in which blank nodes have been equivalently replaced by fresh non-distinguished variables, the RDF evaluation $q(\texttt{G})$ of q against $\texttt{G}$ is obtained by the relational evaluation of the conjunctive query $Q(\bar{x})$:- $\bigwedge_{i=1}^{n} \texttt{Triple}(\texttt{s}_i, \texttt{p}_i, \texttt{o}_i)$ against the aforementioned `Triple` table. Indeed, RDF and relational evaluations coincide with the above encoding, as relational evaluation amounts to finding all the total assignments from the variables of the query to the values (constants and variables) in the `Triple` table, so that the query becomes a subset of that `Triple` table.

It follows that evaluating $Q(\bar{x})$:- $\bigwedge_{i=1}^{n} \texttt{Triple}(\texttt{s}_i, \texttt{p}_i, \texttt{o}_i)$ against the `Triple` table containing the saturation of $\texttt{G}$, instead of $\texttt{G}$ itself, computes the answer set of q against $\texttt{G}$. Hence, BGP queries against graphs can be evaluated by a standard RDBMS.

Example 4 (Continued) *Provided that the saturation of the above graph* $\mathtt{G}'$ *is encoded in a V-table* $\mathtt{Triple}(\mathtt{s},\mathtt{p},\mathtt{o})$ *as described above, the answer set of the following BGP query:*

$$q(x)\text{:- } y_1 \text{ hasAuthor } x \text{ .}, y_1 \text{ inProceedingsOf } y_2 \text{ .}, y_2\ y_3 \text{ ``}PODS'98\text{'' .}$$

against $\mathtt{G}'$ *(i.e.,* $q(\mathtt{G}'^{\infty})$*) is the same as the result of evaluating the following relational conjunctive query against the* `Triple` *table:*

$$Q(x)\text{:- } \mathtt{Triple}(y_1, \text{hasAuthor}, x) \wedge \mathtt{Triple}(y_1, \text{inProceedingsOf}, y_2) \wedge \mathtt{Triple}(y_2, y_3, \text{``}PODS'98\text{''}).$$

2.2 The DL-lite family of Description Logics

The DL-lite family [CGL$^+$07] of Descriptions Logics [BCM$^+$03] allows defining *knowledge bases* (Section 2.2.1) that can be queried with the well-known *conjunctive queries* (Section 2.2.2), a.k.a. select-project-join queries, from the relational database theory [AHV95]. The prevalent techniques for consistency checking and query answering is *first order logic (FOL) reducibility*, which reduces these tasks to the evaluation of FOL queries (Section 2.2.3).

2.2.1 Knowledge bases

Generally speaking, a DL *knowledge base* (KB) consists of a schema called a *Tbox* and its associated dataset called an *Abox*. A Tbox $\mathtt{T}$ is defined upon a *signature* (a.k.a. *vocabulary*), denoted $\mathtt{sig}(\mathtt{T})$, which is the disjoint union of a set of unary relations called *atomic concepts* and a set of binary relations called *atomic roles*. A Tbox is a set of constraints called *terminological axioms*, typically inclusion constraints between complex concepts or roles, i.e., unary or binary DL formulae built upon atomic relations using the constructors allowed in DL under consideration. An Abox defined upon $\mathtt{sig}(\mathtt{T})$ is a set of facts called *assertional axioms*, relating DL formulae to their instances. The legal KBs vary according to the DL used to express terminological and assertional axioms, and to the restrictions imposed on those axioms.

In DL-lite, the concepts and roles that can be built from atomic concepts and atomic roles are of the following form:

$$B \to A \mid \exists R, \qquad C \to B \mid \neg B, \qquad R \to P \mid P^-, \qquad E \to R \mid \neg R$$

where A denotes an *atomic concept*, P an *atomic role*, and P^- the *inverse* of P; B denotes a *basic concept* (i.e., an atomic concept A or an *unqualified existential quantification on a basic role* $\exists R$) and R a *basic role* (i.e., an atomic role P or its inverse P^-); C denotes a *general concept* (i.e., a basic concept or its negation) and E a *general role* (i.e., a basic role or its negation).

The (set) semantics of concepts and roles is given in terms of interpretations. An *interpretation* $I = (\Delta^I, .^I)$ consists of a nonempty *interpretation domain* Δ^I and an *interpretation function* $.^I$ that assigns a subset of Δ^I to each atomic concept, and a binary relation over Δ^I to each atomic role. The semantics of non-atomic concepts and non-atomic roles is defined as follows:

- $(P^-)^I = \{(o_2, o_1) \mid (o_1, o_2) \in P^I\}$,
- $(\exists R)^I = \{o_1 \mid \exists o_2\ (o_1, o_2) \in R^I\}$, and
- $(\neg B)^I = \Delta^I \backslash B^I$ and $(\neg R)^I = \Delta^I \times \Delta^I \backslash R^I$.

The axioms allowed in a Tbox of DL-lite are concept inclusion constraints of the form $B \sqsubseteq C$, role inclusion constraints of the form $R \sqsubseteq E$, and functionality constraints on roles of the form $(\mathit{funct}\ R)$. Observe that negated concepts or roles are only allowed on the right hand side of inclusion constraints, whereas only positive concepts or roles occur on the left hand side of such constraints. Moreover, only basic roles occur in functionality constraints.

Inclusions of the form $B_1 \sqsubseteq B_2$ or $R_1 \sqsubseteq R_2$ are called *positive inclusions (PIs)*, while inclusions of the form $B_1 \sqsubseteq \neg B_2$ or of the form $R_1 \sqsubseteq \neg R_2$ are called *negative inclusions (NIs)*. PIs allow expressing inclusion dependencies, while NIs and functionalities allow expressing integrity constraints (ICs).

An interpretation $I = (\Delta^I, .^I)$ is a *model of an inclusion* $B \sqsubseteq C$ (resp. $R \sqsubseteq E$) if $B^I \subseteq C^I$ (resp. $R^I \subseteq E^I$). It is a *model of a functionality constraint* $(\mathit{funct}\ R)$ if the binary relation R^I is a function, i.e., $(o, o_1) \in R^I$ and $(o, o_2) \in R^I$ implies $o_1 = o_2$. I is a *model of a Tbox* if it is a model of all of its constraints. A Tbox is *satisfiable* if it has a model. A Tbox $\mathtt{T}$ *logically entails* (a.k.a. *implies*) a constraint α, written $\mathtt{T} \models \alpha$, if every model of $\mathtt{T}$ is a model of α. Finally, a Tbox $\mathtt{T}$ *logically entails* (a.k.a. *implies*) a Tbox $\mathtt{T}'$, written $\mathtt{T} \models \mathtt{T}'$, if every model of $\mathtt{T}$ is a model of $\mathtt{T}'$; and two Tboxes $\mathtt{T}$ and $\mathtt{T}'$ are *logically equivalent*, written $\mathtt{T} \equiv \mathtt{T}'$, iff $\mathtt{T} \models \mathtt{T}'$ and $\mathtt{T}' \models \mathtt{T}$.

1. $\texttt{Publication} \sqsubseteq \exists\texttt{hasTitle}$, $(\mathit{funct}\ \texttt{hasTitle})$
2. $\texttt{Publication} \sqsubseteq \exists\texttt{hasDate}$, $(\mathit{funct}\ \texttt{hasDate})$
3. $\texttt{Publication} \sqsubseteq \exists\texttt{hasVenue}$, $(\mathit{funct}\ \texttt{hasVenue})$
4. $\texttt{Publication} \sqsubseteq \exists\texttt{hasAuthor}$
5. $\exists\texttt{hasTitle} \sqsubseteq \texttt{Publication}$
6. $\texttt{ConfPaper} \sqsubseteq \texttt{Publication}$, $\texttt{JournPaper} \sqsubseteq \texttt{Publication}$, $\texttt{ConfPaper} \sqsubseteq \neg\texttt{JournPaper}$
7. $\texttt{ShortPaper} \sqsubseteq \texttt{ConfPaper}$, $\texttt{FullPaper} \sqsubseteq \texttt{ConfPaper}$, $\texttt{FullPaper} \sqsubseteq \neg\texttt{ShortPaper}$, $\texttt{Survey} \sqsubseteq \texttt{JournPaper}$

Figure 2.5: A DL-lite Tbox $\texttt{T}$ representing scientific publications.

Example 5 (Running example) *Consider the Tbox* $\texttt{T}$ *in Figure 2.5, representing domain knowledge about scientific publications. Its signature* $\texttt{sig}(\texttt{T})$ *consists of the atomic concepts* $\texttt{Publication}$, $\texttt{ConfPaper}$, $\texttt{ShortPaper}$, $\texttt{FullPaper}$, $\texttt{JournPaper}$, $\texttt{Survey}$, *and of the atomic roles* $\texttt{hasTitle}$, $\texttt{hasDate}$, $\texttt{hasVenue}$, *and* $\texttt{hasAuthor}$.

The constraints in $\texttt{T}$ *state that any publication has a single title (1), a single date of publication (2), a single venue (3), and at least one author (4). In addition, only publications have a title (5), papers in conference proceedings or in journals (which are disjoint) are publications (6), short papers or full papers (which are disjoint) are papers in conference proceedings, and surveys are journal papers (7).*

The Tbox implies the constraint $\texttt{JournPaper} \sqsubseteq \exists\texttt{hasAuthor}$, *which means that a journal paper has at least one author, as it contains* $\texttt{JournPaper} \sqsubseteq \texttt{Publication}$ *and* $\texttt{Publication} \sqsubseteq \exists\texttt{hasAuthor}$.

It also implies the constraint $\texttt{FullPaper} \sqsubseteq \neg\texttt{Survey}$, *which means that surveys and full papers are disjoint, as it contains* $\texttt{FullPaper} \sqsubseteq \texttt{ConfPaper}$, $\texttt{ConfPaper} \sqsubseteq \neg\texttt{JournPaper}$, *and* $\texttt{Survey} \sqsubseteq \texttt{JournPaper}$.

An Abox consists of a finite set of membership assertions of the form $A(a)$ and $P(a, b)$, i.e., on atomic concepts and on atomic roles, stating respectively that a is an instance of A and that the pair of constants (a, b) is an instance of P. The interpretation function of an interpretation $I = (\Delta^I, .^I)$ is extended to constants by assigning to each constant a a distinct object $a^I \in \Delta^I$, i.e., the so called *unique name assumption* holds. An interpretation I is a *model of the membership assertion* $A(a)$ (resp. $P(a, b)$) if $a^I \in A^I$ (resp., $(a^I, b^I) \in P^I$). It is a *model of an Abox* if it satisfies all of its assertions.

Example 6 (Continued) *Consider the Abox in Figure 2.6, representing factual knowledge about scientific publications. It is expressed as relational tables and states in particular that:*

- doi_1 *is the* Digital Object Identifier[4] *(DOI) of the full paper entitled "Complexity of Answering Queries Using Materialized Views" and published in PODS'98 by Serge Abiteboul ("SA") and Oliver M. Duschka ("OD"),*
- doi_2 *is the DOI of the survey entitled "Answering queries using views: A survey" and published in VLDB Journal in 2001 by Alon Y. Halevy ("AH"), and*
- doi_3 *is the DOI of the journal paper entitled "MiniCon: A scalable algorithm for answering queries using views" and published in VLDB Journal in 2001 by Rachel Pottinger ("RP") and Alon Y. Halevy ("AH").*

A KB $\mathcal{K}$ is a pair made of a Tbox $\texttt{T}$ and an Abox $\texttt{A}$, denoted $\mathcal{K} = \langle \texttt{T}, \texttt{A} \rangle$. An interpretation I is a *model of a KB* $\mathcal{K} = \langle \texttt{T}, \texttt{A} \rangle$ if it is a model of both $\texttt{T}$ and $\texttt{A}$. A KB $\mathcal{K}$ is *satisfiable*, a.k.a. *consistent*, if it has at least one model. Observe that Tboxes and Aboxes are always consistent. That is, a KB is inconsistent whenever there is a contradiction between its Abox and Tbox. A KB $\mathcal{K}$ *logically entails*, a.k.a. implies, a constraint or assertion β, written $\mathcal{K} \models \beta$, if every model of $\mathcal{K}$ is a model of β.

Example 7 (Continued) *Consider the consistent KB* $\mathcal{K} = \langle \texttt{T}, \texttt{A} \rangle$ *associating the above Tbox and Abox for scientific publications. It implies* $\texttt{JournPaper}(doi_2)$ *due to* $\texttt{Survey} \sqsubseteq \texttt{JournPaper}$ *in* $\texttt{T}$ *and* $\texttt{Survey}(doi_2)$ *in* $\texttt{A}$. *Adding* $\texttt{FullPaper}(doi_2)$ *to this KB would make it inconsistent, as* $\mathcal{K}$ *entails* $\texttt{FullPaper} \sqsubseteq \neg\texttt{Survey}$ *(cf. previous example) and* $\texttt{Survey}(doi_2)$ *is in* $\texttt{A}$

Observe that any KB can be written equivalently as a FOL KB and a relational database following the *open-world assumption (OWA)* [AHV95]. The correspondences for Tbox constraints are summarized in Figure 2.7 for PIs, in Figure 2.8 for NIs, and in Figure 2.9 for functionalities. As for Abox assertions, they are simply FOL facts (i.e., ground atoms) and instances for atomic concepts and roles.

[4] http://www.doi.org

Publication	
	$\dots$

hasTitle		
	doi_1	”CAQUMV”
	doi_2	”AQUVAS”
	doi_3	”MC : ASAAQUV”
	$\dots$	$\dots$

hasDate		
	doi_1	”1998”
	doi_2	”2001”
	doi_3	”2001”
	$\dots$	$\dots$

hasVenue		
	doi_1	”PODS”
	doi_2	”VLDBJ”
	doi_3	”VLDBJ”
	$\dots$	$\dots$

hasAuthor		
	doi_1	”SA”
	doi_1	”OD”
	doi_2	”AH”
	doi_3	”AH”
	doi_3	”RP”
	$\dots$	$\dots$

ConfPaper	
	$\dots$

JournPaper	
	doi_3
	$\dots$

ShortPaper	
	$\dots$

FullPaper	
	doi_1
	$\dots$

Survey	
	doi_2
	$\dots$

Figure 2.6: A DL-lite Abox A for scientific publications.

DL notation	**FOL notation**	**Relational notation (OWA)**
$A \sqsubseteq A'$	$\forall x[A(x) \Rightarrow A'(x)]$	$A \subseteq A'$
$A \sqsubseteq \exists P$	$\forall x[A(x) \Rightarrow \exists y P(x,y)]$	$A \subseteq \Pi_1(P)$
$A \sqsubseteq \exists P^-$	$\forall x[A(x) \Rightarrow \exists y P(y,x)]$	$A \subseteq \Pi_2(P)$
$\exists P \sqsubseteq A$	$\forall x[\exists y P(x,y) \Rightarrow A(x)]$	$\Pi_1(P) \subseteq A$
$\exists P^- \sqsubseteq A$	$\forall x[\exists y P(y,x) \Rightarrow A(x)]$	$\Pi_2(P) \subseteq A$
$\exists Q \sqsubseteq \exists P$	$\forall x[\exists y Q(x,y) \Rightarrow \exists z P(x,z)]$	$\Pi_1(Q) \subseteq \Pi_1(P)$
$\exists Q \sqsubseteq \exists P^-$	$\forall x[\exists y Q(x,y) \Rightarrow \exists z P(z,x)]$	$\Pi_1(Q) \subseteq \Pi_2(P)$
$\exists Q^- \sqsubseteq \exists P$	$\forall x[\exists y Q(y,x) \Rightarrow \exists z P(x,z)]$	$\Pi_2(Q) \subseteq \Pi_1(P)$
$\exists Q^- \sqsubseteq \exists P^-$	$\forall x[\exists y Q(y,x) \Rightarrow \exists z P(z,x)]$	$\Pi_2(Q) \subseteq \Pi_2(P)$
$P \sqsubseteq Q^-$ or $P^- \sqsubseteq Q$	$\forall x,y[P(x,y) \Rightarrow Q(y,x)]$	$P \subseteq \Pi_{2,1}(Q)$ or $\Pi_{2,1}(P) \subseteq Q$
$P \sqsubseteq Q$ or $P^- \sqsubseteq Q^-$	$\forall x,y[P(x,y) \Rightarrow Q(x,y)]$	$P \subseteq Q$ or $\Pi_{2,1}(P) \subseteq \Pi_{2,1}(Q)$

Figure 2.7: DL-lite PI axioms in FOL and relational notations. For the relational notation, which corresponds to unary and binary *inclusion dependencies*, we assume that the first and second attributes of any atomic role are named 1 and 2 respectively.

DL notation	**FOL notation**	**Relational notation (OWA)**
$A \sqsubseteq \neg A'$	$\forall x[A(x) \Rightarrow \neg A'(x)]$	$A \cap A' \subseteq \bot$
$A \sqsubseteq \neg\exists P$	$\forall x[A(x) \Rightarrow \neg\exists y P(x,y)]$	$A \cap \Pi_1(P) \subseteq \bot$
$A \sqsubseteq \neg\exists P^-$	$\forall x[A(x) \Rightarrow \neg\exists y P(y,x)]$	$A \cap \Pi_2(P) \subseteq \bot$
$\exists P \sqsubseteq \neg A$	$\forall x[\exists y P(x,y) \Rightarrow \neg A(x)]$	$A \cap \Pi_1(P) \subseteq \bot$
$\exists P^- \sqsubseteq \neg A$	$\forall x[\exists y P(y,x) \Rightarrow \neg A(x)]$	$A \cap \Pi_2(P) \subseteq \bot$
$\exists Q \sqsubseteq \neg\exists P$	$\forall x[\exists y Q(x,y) \Rightarrow \neg\exists z P(x,z)]$	$\Pi_1(Q) \cap \Pi_1(P) \subseteq \bot$
$\exists Q \sqsubseteq \neg\exists P^-$	$\forall x[\exists y Q(x,y) \Rightarrow \neg\exists z P(z,x)]$	$\Pi_1(Q) \cap \Pi_2(P) \subseteq \bot$
$\exists Q^- \sqsubseteq \neg\exists P$	$\forall x[\exists y Q(y,x) \Rightarrow \neg\exists z P(x,z)]$	$\Pi_2(Q) \cap \Pi_1(P) \subseteq \bot$
$\exists Q^- \sqsubseteq \neg\exists P^-$	$\forall x[Q(y,x) \Rightarrow \neg\exists z P(z,x)]$	$\Pi_2(Q) \cap \Pi_2(P) \subseteq \bot$
$P \sqsubseteq \neg Q^-$ or $P^- \sqsubseteq \neg Q$	$\forall x,y[P(x,y) \Rightarrow \neg Q(y,x)]$	$P \cap \Pi_{2,1}(Q) \subseteq \bot$ or $\Pi_{2,1}(P) \cap Q \subseteq \bot$
$P \sqsubseteq \neg Q$ or $P^- \sqsubseteq \neg Q^-$	$\forall x,y[P(x,y) \Rightarrow \neg Q(x,y)]$	$P \cap Q \subseteq \bot$ or $\Pi_{2,1}(P) \cap \Pi_{2,1}(Q) \subseteq \bot$

Figure 2.8: DL-lite NI axioms in FOL and relational notations. For the relational notation, which corresponds to *exclusion/disjointness dependencies*, we assume that the first and second attributes of any atomic role are named 1 and 2 respectively. We also assume that $\bot$ the empty relation.

DL notation	**FOL notation**	**Relational notation (OWA)**
$(funct\ P)$	$\forall x,y,z[P(x,y) \wedge P(x,z) \Rightarrow y = z]$	$P:\ 1 \rightarrow 2$
$(funct\ P^-)$	$\forall x,y,z[P(y,x) \wedge P(z,x) \Rightarrow y = z]$	$P:\ 2 \rightarrow 1$

Figure 2.9: DL-lite functionality axioms in FOL and relational notations. For the relational notation, which corresponds to *functional dependencies*, we assume that the first and second attributes of any atomic role are named 1 and 2 respectively.

2.2.2 Queries

A FOL query q is of the form $q(\bar{x})$:- $\phi(\bar{x})$ where $\phi(\bar{x})$ is a FOL formula, the *free* variables of which are *only* the variables $\bar{x}$, and the predicates of which are either *atomic* concepts or roles. The *arity* of a query is the number of its free variables, e.g., 0 for a *boolean query*. When $\phi(\bar{x})$ is of the form $\exists\bar{y}\ conj(\bar{x},\bar{y})$ with $conj(\bar{x},\bar{y})$ a conjunction of atoms, q is called a *conjunctive query*. Conjunctive queries, a.k.a. select-project-join queries, are the core relational database queries.

Given an interpretation $I = (\Delta^I, .^I)$, the semantics q^I of a boolean query q is defined as true if $[\phi(\emptyset)]^I = \mathrm{true}$, and false otherwise, while the semantics q^I of a query q of arity $n \geq 1$ is the relation of arity n defined on Δ^I as follows: $q^I = \{\bar{e} \in (\Delta^I)^n \mid [\phi(\bar{e})]^I = \mathrm{true}\}$. An interpretation that evaluates a boolean query to true,

NI	**Corresponding $unsat$ query**
$A \sqsubseteq \neg A'$ or $A' \sqsubseteq \neg A$	$\exists x[A(x) \wedge A'(x)]$
$A \sqsubseteq \neg \exists P$ or $\exists P \sqsubseteq \neg A$	$\exists x, y[A(x) \wedge P(x, y)]$
$A \sqsubseteq \neg \exists P^-$ or $\exists P^- \sqsubseteq \neg A$	$\exists x, y[A(x) \wedge P(y, x)]$
$\exists Q \sqsubseteq \neg \exists P$ or $\exists P \sqsubseteq \neg \exists Q$	$\exists x, y, z[Q(x, y) \wedge P(x, z)]$
$\exists Q \sqsubseteq \neg \exists P^-$ or $\exists P^- \sqsubseteq \neg \exists Q$	$\exists x, y, z[Q(x, y) \wedge P(z, x)]$
$\exists Q^- \sqsubseteq \neg \exists P$ or $\exists P \sqsubseteq \neg \exists Q^-$	$\exists x, y, z[Q(y, x) \wedge P(x, z)]$
$\exists Q^- \sqsubseteq \neg \exists P^-$	$\exists x, y, z[Q(y, x) \wedge P(z, x)]$
$P \sqsubseteq \neg Q^-$ or $Q^- \sqsubseteq \neg P$ or $P^- \sqsubseteq \neg Q$ or $Q \sqsubseteq P^-$	$\exists x, y[P(x, y) \wedge Q(y, x)]$
$P \sqsubseteq \neg Q$ or $Q \sqsubseteq \neg P$ or $P^- \sqsubseteq \neg Q^-$ or $Q^- \sqsubseteq \neg P^-$	$\exists x, y[P(x, y) \wedge Q(x, y)]$

Figure 2.10: From NI axioms to $unsat$ queries.

respectively a non-boolean query to a non empty set, is a *model* of that query.

Let q be a query against a KB $\mathcal{K} = \langle \mathtt{T}, \mathtt{A} \rangle$. If q is non-boolean, the answer set of q against $\mathcal{K}$ is defined as: $ans(q, \mathcal{K}) = \{\bar{t} \in \mathcal{C}^n \mid \mathcal{K} \models q(\bar{t})\}$ where $\mathcal{C}$ is the set of constants appearing in the KB, $q(\bar{t})$ is the closed formula obtained by replacing in the query definition the free variables in $\bar{x}$ by the constants in $\bar{t}$, and $\mathcal{K} \models q(\bar{t})$ means as usual that every model of $\mathcal{K}$ is a model of $q(\bar{t})$. If q is boolean, the answer set of q against $\mathcal{K}$ is by convention either $\{true\}$ or $\{false\}$: $ans(q, \mathcal{K}) = \{true\}$ if and only if $\mathcal{K} \models q()$, i.e., every model of $\mathcal{K}$ is a model of $q()$. This corresponds to the so-called *certain answers semantics* requiring that an answer to a query, given a set of constraints (expressed here as a Tbox), to be an answer in all the models satisfying the constraints.

Example 8 (Continued) *Consider the following query against the above KB $\mathcal{K}$ asking for the doi's of journal paper and their authors: $q(x, y)$:- $\mathtt{JournalPaper}(x) \wedge \mathtt{hasAuthor}(x, y)$. Its answer set is: $ans(q, \mathcal{K}) = \{(doi_2, "AH"), (doi_3, "AH"), (doi_3, "RP")\}$.*

2.2.3 Consistency checking and query answering

The DL-Lite family [CGL$^+$07] has been designed so that data management is *FOL-reducible*. This property allows reducing a data management task over a KB $\langle \mathtt{T}, \mathtt{A} \rangle$ to the evaluation against $\mathtt{A}$ *only* of a FOL query computed using $\mathtt{T}$ *only*.

The main idea of FOL-reducibility is to be able to perform a data management task in two separate steps: a first reasoning step that produces the FOL query and a second step which evaluates this query in a pure relational fashion. Indeed, FOL queries can be processed by SQL engines, thus taking advantage of well-established query optimization strategies supported by standard relational database management systems.

In fact, FOL-reducibility of data management holds in DL-lite only if we *forbid* functionality constraints on roles involved in right-hand sides of role inclusion constraints[5]. E.g., in Figure 2.5, having $\mathtt{hasContactAuthor} \sqsubseteq \mathtt{hasAuthor}$ and $(\mathit{funct}\ \mathtt{hasContactAuthor})$ in $\mathtt{T}$ is legal, while having $\mathtt{hasContactAuthor} \sqsubseteq \mathtt{hasAuthor}$ and $(\mathit{funct}\ \mathtt{hasAuthor})$ would be illegal. In the following, we only consider DL-lite Tboxes and KBs in which this restriction holds. Note that, as shown in [CGL$^+$07], if we do not impose the above restriction on DL-lite, instance checking (a particular case of query answering) is P-complete in data complexity, rulling out FOL-reducibility of query answering since data complexity of answering a FOL query is in $AC_0 \subset P$ [Var82, AHV95].

Consistency checking It has been shown in [CGL$^+$07] that given a Tbox $\mathtt{T}$, it is always possible to construct a FOL query q_{unsat} such that $ans(q_{unsat}, \mathtt{A}) = \{\text{true}\}$[6] iff the KB $\langle \mathtt{T}, \mathtt{A} \rangle$ is inconsistent, for any Abox $\mathtt{A}$ associated with $\mathtt{T}$. [CGL$^+$07] provides the $\mathsf{Consistent}(\mathcal{K})$ algorithm to check the consistency of a KB $\mathcal{K}$ based on this result.

Building the q_{unsat} query relies on the computation of the *IC-closure* of $\mathtt{T}$, i.e., the set of the integrity constraints (NI or functionality constraints) that are implied by $\mathtt{T}$: each constraint in the IC-closure is transformed into a conjunctive boolean query looking for counter-examples to it ; q_{unsat} is the union of these $unsat$ queries. The transformation of the integrity constraints into $unsat$ queries corresponds in fact to their negation and is summarized in Figure 2.10 for NIs and in Figure 2.11 for functionalities.

Example 9 (Continued) *The $unsat$ query corresponding to the negation of the NI* $\mathtt{JournPaper} \sqsubseteq \neg\mathtt{ConfPaper}$ *is $q()$:- $\exists x$ $\mathtt{JournPaper}(x) \wedge \mathtt{ConfPaper}(x)$.*

[5] This corresponds to the dialect DL-lite$_{\mathcal{A}}$ of DL-lite.

[6] By a slight abuse of notation, I denote hereinafter the answer set $ans(q, \langle \emptyset, \mathtt{A} \rangle)$ of a query q against a KB $\langle \emptyset, \mathtt{A} \rangle$ by $ans(q, \mathtt{A})$, which corresponds exactly to the standard relational evaluation of q against the relational database $\mathtt{A}$.

Functionality	**Corresponding $unsat$ query**
$(funct\ P)$	$\exists x, y, z[P(x, y) \wedge P(x, z) \wedge y \neq z)]$
$(funct\ P^-)$	$\exists x, y, z[P(y, x) \wedge P(z, x) \wedge y \neq z)]$

Figure 2.11: From functionality axioms to $unsat$ queries.

RDF(S) statement	DL-lite statement	Note
$\mathtt{s}$ rdfs:subClassOf $\mathtt{o}$.	$\mathtt{s} \sqsubseteq \mathtt{o}$	$\mathtt{s}, \mathtt{o}$ are classes/atomic concepts
$\mathtt{s}$ rdfs:subPropertyOf $\mathtt{o}$.	$\mathtt{s} \sqsubseteq \mathtt{o}$	$\mathtt{s}, \mathtt{o}$ are properties/atomic roles
$\mathtt{s}$ rdfs:domain $\mathtt{o}$.	$\exists \mathtt{s} \sqsubseteq \mathtt{o}$	$\mathtt{s}$ is a property / an atomic role $\mathtt{o}$ is a class / an atomic concept
$\mathtt{s}$ rdfs:range $\mathtt{o}$.	$\exists \mathtt{s}^- \sqsubseteq \mathtt{o}$	$\mathtt{s}$ is a property / an atomic role $\mathtt{o}$ is a class / an atomic concept
$\mathtt{s}$ rdf:type $\mathtt{o}$.	$\mathtt{o}(\mathtt{s})$	$\mathtt{o}$ is a class / an atomic concept
$\mathtt{s}$ $\mathtt{p}$ $\mathtt{o}$.	$\mathtt{p}(\mathtt{s}, \mathtt{o})$	$\mathtt{p}$ is a property / an atomic role

Figure 2.12: Correspondences between RDF graphs and DL-lite KBs. Blank nodes are not allowed and assertions cannot refer to classes or properties.

Reformulation-based query answering It has been shown in [CGL$^+$07] that given a Tbox $\mathtt{T}$ and for any query q built upon atomic concepts and roles of $\mathtt{T}$, it is always possible to construct a FOL query q_{ref} called its *perfect reformulation*, such that $ans(q, \langle \mathtt{T}, \mathtt{A} \rangle) = ans(q_{ref}, \mathtt{A})$ for any Abox $\mathtt{A}$ associated with $\mathtt{T}$. [CGL$^+$07] provides the $\mathsf{PerfectRef}(q, \mathtt{T})$ algorithm which computes the perfect reformulation q_{ref} of q using – PIs of – $\mathtt{T}$ only (i.e., independently of $\mathtt{A}$), which is a union of conjunctive queries built upon atomic concepts and roles of $\mathtt{T}$.

Example 10 (Continued) *Consider the previous query asking for the doi's of journal paper and their authors:* $q(x, y)$:- $\mathtt{JournalPaper}(x) \wedge \mathtt{hasAuthor}(x, y)$. *It's reformulation computed by* $\mathsf{PerfectRef}$ *is the union query:* $q(x, y)$:- $\mathtt{JournalPaper}(x) \wedge \mathtt{hasAuthor}(x, y) \cup q(x, y)$:- $\mathtt{Survey}(x) \wedge \mathtt{hasAuthor}(x, y)$.

2.3 RDF meets DL-lite

RDF and DL-lite on the one hand, and SPARQL and the relational conjunctive queries for DL-lite on the other hand, share some expressivity. I recall here the correspondences between these data models (Section 2.3.1) and between their query languages (Section 2.3.2).

2.3.1 Correspondence between RDF and DL-lite: the DL fragment of RDF

RDF can be used to model some DL KBs, actually some DL-lite KBs and, conversely, DL-lite can be used to model some RDF graphs.

Figure 2.12 exhibits the exact correspondence between RDF graphs and DL-lite KBs, provided some restriction have been made on the RDF model:

1. Blank nodes are disallowed, ruling out the possibility to express complex incomplete information, in particular about unknown classes or properties. This RDF feature does not translate into DLs which, as FOL languages, does not permit the (existential) quantification on concepts and roles. For instance, the RDF graph $\{$posterCP rdfs:subClassOf confP .$, doi$ rdf:type posterCP .$\}$ translates into the DL-lite KB $\langle \{\text{posterCP} \sqsubseteq \text{confP}\}, \{\text{posterCP}(doi)\} \rangle$, while $\{\_{:}b$ rdfs:subClassOf confP .$, doi$ rdf:type $\_{:}b$.$\}$ does not translate, as it would correspond to the non-well-formed KB $\exists x \langle \{x \sqsubseteq \text{confP}\}, \{x(doi)\} \rangle$.

2. Values in assertions cannot be classes or properties, ruling out the possibility to talk about them. This RDF feature also does not translate into DLs which, as FOL languages, make a distinction between relations and constants. For instance, the graph $\{doi_1$ hasAuthor $"SA"$.$,$ $\text{hasAuthor sameAs writtenBy}$.$\}$ does not translate into a DL-lite KB, as such a KB cannot model that hasAuthor is – at the same time – both a relation and a constant.

3. RDF entailment is limited to RDFS entailment only, i.e., the entailment rules dedicated to RDF Schema given in Figures 2.13–2.16.

This fragment of RDF, that allows modeling DL-lite KBs, is widely known as *the DL fragment of RDF*.

Triple	Entailed triple ($\vdash^{i}_{\text{RDF}}$)
s rdfs:subClassOf o .	s rdfs:subClassOf s .
s rdfs:subClassOf o .	o rdfs:subClassOf o .
s rdfs:subPropertyOf o .	s rdfs:subPropertyOf s .
s rdfs:subPropertyOf o .	o rdfs:subPropertyOf o .
s rdfs:domain o .	s rdfs:subPropertyOf s .
s rdfs:domain o .	o rdfs:subClassOf o .
s rdfs:domain rdfs:Literal .	s rdfs:subPropertyOf s .
s rdfs:range o .	s rdfs:subPropertyOf s .
s rdfs:range o .	o rdfs:subClassOf o .
s rdfs:range rdfs:Literal .	s rdfs:subPropertyOf s .

Figure 2.13: RDFS entailment from a single RDFS statement.

Triple	Entailed triple ($\vdash^{i}_{\text{RDF}}$)
s rdf:type o .	o rdfs:subClassOf o .
s p o .	p rdfs:subPropertyOf p .

Figure 2.14: RDFS entailment from a single RDF statement.

Triples	Entailed triple ($\vdash^{i}_{\text{RDF}}$)
s rdfs:subClassOf o ., o rdfs:subClassOf o_1 .	s rdfs:subClassOf o_1 .
s rdfs:subPropertyOf o ., o rdfs:subPropertyOf o_1 .	s rdfs:subPropertyOf o_1 .
s rdfs:domain o ., o rdfs:subClassOf o_1 .	s rdfs:domain o_1 .
s rdfs:range o ., o rdfs:subClassOf o_1 .	s rdfs:range o_1 .
s rdfs:domain o ., s_1 rdfs:subPropertyOf s .	s_1 rdfs:domain o .
s rdfs:range o ., s_1 rdfs:subPropertyOf s .	s_1 rdfs:range o .

Figure 2.15: RDFS entailment from two RDFS statements.

Triples	Entailed triple ($\vdash^{i}_{\text{RDF}}$)
s_1 rdfs:subClassOf s_2 ., s rdf:type s_1 .	s rdf:type s_2 .
p_1 rdfs:subPropertyOf p_2 ., s p_1 o .	s p_2 o .
p rdfs:domain s ., s_1 p o_1 .	s_1 rdf:type s .
p rdfs:range s ., s_1 p o_1 .	o_1 rdf:type s .

Figure 2.16: RDFS entailment from combining RDFS and RDF statements.

SPARQL (BGP) triple pattern	Relational conjunctive query atom	Note
s rdf:type o .	o(s)	o is a class / an atomic concept
s p o .	p(s, o)	p is a property / an atomic role

Figure 2.17: Correspondences between conjunctive queries and SPARQL (BGP) queries.

2.3.2 Correspondence between SPARQL and relational conjunctive queries for DL-lite

Relational conjunctive queries for DL-lite turn out to be a subset of SPARQL queries, actually of BGP queries. Figure 2.17 exhibits the exact correspondence between these query languages.

The language of relational conjunctive queries for DL-lite is *strictly less* expressive than the BGP queries, which allows, in addition, to use variables in place of classes and properties, in order to express unspecified relations. For instance, the query $q(x)$:- y_1 hasAuthor x .,y_1 inProceedingsOf y_2 .,y_2 y_3 $''PODS'98''$. has no counterpart in the language of relational conjunctive queries, as it would correspond to the non-well-formed query: $q(x)$:- $\text{hasAuthor}(y_1, x), \text{inProceedingsOf}(y_1, y_2), y_3(y_2, ''PODS'98'')$.

Chapter 3

Design

The main contribution I present is the design of *robust module-based data management techniques for DL-lite* (Section 3.1). Module-based data management amounts to handling data using an ontology – a module – that derives from that of a preexisting well-established ontology-based data management system. The novel notion of *robust module* basically allows enhancing data integrity and complementing the answers to queries in ontology-based data management systems.

The ongoing work I present next is the design of *query answering techniques for RDF that are robust to graph updates* (Section 3.2). Saturation-based query answering and the alternative technique called reformulation-based query answering are revisited: a saturation maintenance technique is designed for the former to limit the re-saturation effort upon graph updates, while the latter – de facto robust to updates – is extended to a larger fragment of RDF than those investigated in the literature.

3.1 Robust module-based data management in DL-lite

Context Since 2010, I have been working with Marie-Christine Rousset (PR, Univ. Grenoble) on robust module-based data management in DL-lite.

Our preliminary results were published in the proceedings of Journées Bases de Données Avancées (BDA) in 2010 [GR10] ; all our results obtained so far were published in IEEE Transactions on Knowledge and Data Engineering (TKDE) in 2012 [GR12].

Motivations In many application domains (e.g., medicine or biology), comprehensive ontologies resulting from collaborative initiatives are made available. For instance, SNOMED is an ontology containing more than 400.000 concept names covering various areas such as anatomy, diseases, medication, and even geographic locations. Such well-established ontologies are often associated with reliable data that have been carefully collected, cleansed, and verified, thus providing *reference* ontology-based data management systems (DMSs) in different application domains.

A good practice is therefore to build on the efforts made to design reference DMSs whenever we have to develop our own DMS with specific needs. A way to do this is to extract from the reference DMS the piece of ontology relevant to our application needs, possibly to personalize it with extra-constraints w.r.t. our application under construction, and then to manage our own dataset using the resulting ontology.

For instance, the MyCF DMS (MyCorporisFabrica, www.mycorporisfabrica.org, [PBJ$^+$09]) has been built *by hand* from the FMA DMS (Foundational Model of Anatomy, sig.biostr.washington.edu/projects/fm). The extraction step has focused on particular parts of the human body (e.g., hand, foot, and knee), while the personalization step has enriched the descriptions of these parts with both 3D geometrical and bio-mechanical information. Notably, careful attention was paid so that MyCF still conforms with FMA at the end of the *manual* process.

Contributions In [GR10, GR12], we revisit the reuse of a reference DMS in order to build a new DMS with specific needs. We go one step further by not only considering the *design* of a module-based DMS (i.e., how to extract a module from an ontology): we also study how a module-based DMS can benefit from the reference DMS throughout its life-cycle. More specifically, our main contributions are:

1. We provide novel properties of *robustness* for modules in order to enhance data integrity and to complement the answers to queries. From a module *robust to consistency checking*, for any data update in a module-based DMS, we show how to query the reference DMS for checking whether the local update does not bring any inconsistency with the data and the constraints of the reference DMS. From a module *robust to query answering*, for any query asked to a module-based DMS, we show how to query the reference DMS for obtaining additional answers by also exploiting the data stored in the reference DMS. In addition, we provide a *polynomial time algorithm* for extracting robust modules from ontologies.

2. We define sufficient conditions for *safe module personalization*, that are natural from a practical viewpoint, under which we show that global consistency checking and global query answering are preserved. In addition, we provide a *polynomial time algorithm* for checking whether a module personalization is safe.

3. We devise optimizations for *on-demand* or *upon update* global consistency checking: in both case, we have characterized the exact subset of relevant integrity constraints to be checked for deciding global consistency.

4. We propose optimizations for *practical* module-based data management: we have defined *minimal* modules and how to compute them in polynomial time. Such modules are desirable, since non-minimality induces useless extra-computation in well-established DMSs (e.g., QuOnto[1]).

In the following, I give some intuitions on and I exemplify our key notions of module (Section 3.1.1), of robustness to consistency checking (Section 3.1.2) and of robustness to query answering (Section 3.1.3).

3.1.1 Module

Recent work in description logics (DLs) [BCM$^+$03] provides different solutions to achieve a reuse of a *reference* DMS. All these solutions consist in extracting a *module* from an existing Tbox such that all the constraints concerning the *signature of interest* (i.e., concepts and roles) for the application under construction are captured in the module [SPS09]. Existing definitions of modules in the literature basically resort to the notion of (deductive) conservative extension of a Tbox or of uniform interpolant of a Tbox, a.k.a. forgetting about *non*-interesting relations of a Tbox. [GLW06] formalizes those two notions for Tboxes and discusses their connection. Up to now, conservative extension has been considered for defining a module as a *subset* of a Tbox. In contrast, forgetting has been considered for defining a module as only logically *implied* by a Tbox (by definition forgetting cannot lead to a subset of a Tbox in the general case). Both kinds of modules have been investigated in various DLs, e.g., DL-lite [KPS$^+$09, WWTP08], $\mathcal{EL}$ [KWW09a, KLWW08, KWW09b], and $\mathcal{ALC}$ [CHKS07, KLWW08, WWT$^+$09].

Our definition of module extends and encompasses the existing definitions. In contrast with [GLW06, CHKS07, KLWW08, KPS$^+$09], we do not impose modules of a Tbox to be subsets of it. In contrast with [WWTP08, KWW09a, KWW09b, WWT$^+$09], we do not impose the signature of modules to be restricted to the relations (i.e., concepts and roles) of interest. In fact, as we will see later on, the aforementioned the robustness properties for a module may enforce the signature of modules to contain relations from the reference Tbox that are not relations of interest (but that are logically related to them). We therefore simply require that a module captures some constraints of the Tbox only, including all the (implied) constraints built only upon the relations of interest.

Example 11 (Running example) *Consider a DL-lite reference DMS for scientific publications in Figure 3.1, whose KB has been introduced in Section 2.2.*

Suppose that we have to develop a DMS about scientific publications. If we are interested in managing journal papers and their authors only, we can extract a module from $\mathtt{T}$ *w.r.t. the signature* Γ *made of the* relations of interest $\mathtt{JournPaper}$ *and* $\mathtt{hasAuthor}$.

Let $\mathtt{T}'$ *and* $\mathtt{T}''$ *be the following Tboxes:* $\mathtt{T}' = \{\mathtt{JournPaper} \sqsubseteq \exists\mathtt{hasAuthor}\}$ *and* $\mathtt{T}'' = \{\mathtt{Publication} \sqsubseteq \exists\mathtt{hasAuthor}, \exists\mathtt{hasTitle} \sqsubseteq \mathtt{Publication}, \mathtt{ConfPaper} \sqsubseteq \mathtt{Publication}, \mathtt{JournPaper} \sqsubseteq \mathtt{Publication}, \mathtt{ShortPaper} \sqsubseteq \mathtt{ConfPaper}, \mathtt{FullPaper} \sqsubseteq \mathtt{ConfPaper}, \mathtt{Survey} \sqsubseteq \mathtt{JournPaper}\}$.

$\mathtt{T}'$ *and* $\mathtt{T}''$ *are both modules of* $\mathtt{T}$ *w.r.t.* Γ. *Notably, they contain or entail* $\mathtt{JournPaper} \sqsubseteq \exists\mathtt{hasAuthor}$, *the only constraint built upon* Γ *only that is implied by* $\mathtt{T}$ *(it is implied by* $\mathtt{JournPaper} \sqsubseteq \mathtt{Publication}$ *and* $\mathtt{Publication} \sqsubseteq \exists\mathtt{hasAuthor}$ *in* $\mathtt{T}$*).* $\mathtt{T}'$ *is not of subset of* $\mathtt{T}$, *it is actually the result of forgetting the non-interesting relations in* $\mathtt{T}$ *;* $\mathtt{T}''$ *is a subset of* $\mathtt{T}$.

Observe that a module of a Tbox w.r.t. a given signature of interest may not be unique.

[1] http://www.dis.uniroma1.it/quonto/

T:

1. Publication $\sqsubseteq$ $\exists$hasTitle, ($funct$ hasTitle)
2. Publication $\sqsubseteq$ $\exists$hasDate, ($funct$ hasDate)
3. Publication $\sqsubseteq$ $\exists$hasVenue, ($funct$ hasVenue)
4. Publication $\sqsubseteq$ $\exists$hasAuthor
5. $\exists$hasTitle $\sqsubseteq$ Publication
6. ConfPaper $\sqsubseteq$ Publication, JournPaper $\sqsubseteq$ Publication, ConfPaper $\sqsubseteq$ $\neg$JournPaper
7. ShortPaper $\sqsubseteq$ ConfPaper, FullPaper $\sqsubseteq$ ConfPaper, FullPaper $\sqsubseteq$ $\neg$ShortPaper, Survey $\sqsubseteq$ JournPaper

A:

Publication	
	...

hasTitle		
	doi_1	”CAQUMV”
	doi_2	”AQUVAS”
	doi_3	”MC : ASAAQUV”
	...	...

hasDate		
	doi_1	”1998”
	doi_2	”2001”
	doi_3	”2001”
	...	...

hasVenue		
	doi_1	”PODS”
	doi_2	”VLDBJ”
	doi_3	”VLDBJ”
	...	...

hasAuthor		
	doi_1	”SA”
	doi_1	”OD”
	doi_2	”AH”
	doi_3	”AH”
	doi_3	”RP”
	...	...

ConfPaper	
	...

JournPaper	
	doi_3
	...

ShortPaper	
	...

FullPaper	
	doi_1
	...

Survey	
	doi_2
	...

Figure 3.1: A reference DMS $\mathcal{K} = \langle \mathtt{T}, \mathtt{A} \rangle$ for scientific publications.

T$'$: JournPaper $\sqsubseteq$ $\exists$hasAuthor

A$'$:

JournPaper	
	doi_1
	...

hasAuthor		
	doi_1	”SA”
	doi_1	”OD”
	...	...

Figure 3.2: A module-based DMS defined by the Tbox T$'$ and the Abox A$'$.

3.1.2 Robustness to consistency checking for global consistency checking

Robustness to consistency checking allows a module-based DMS deciding whether its data conforms to the whole reference system (ontology and data). Indeed, a module-based DMS can be locally consistent while, at the same time, its data may contradict these of the reference DMS. Detecting this kind of inconsistency, called a *global* inconsistency, is important since it indicates that local data contradicting the reference DMS is probably erroneous.

The basic idea is therefore to use the whole reference DMS (Tbox and Abox) as extra-constraints to be satisfied by a module-based DMS. Of course, we do not want to import the whole reference DMS into our own DMS in order to do this. Instead, we extend the notion of module so that *global consistency checking* can be performed on demand or upon update: We ensure that the module captures the (possibly implied) constraints from the reference Tbox that are required to detect inconsistency related to the relations of interest, i.e., the integrity constraints involving at least a relation of interest. Then, at global consistency checking time, these constraints are verified against the *distributed* Abox consisting of the Abox of the module-based DMS plus that of the reference DMS. This amounts to building and evaluating a particular conjunctive query involving the two DMSs, which looks for the existence of a counter-example to any of these constraints. From a practical viewpoint, this is achieved by a slight modification of the Consistent algorithm [CGL$^+$07] (cf. Section 2.2.3).

Example 12 (Continued) *Suppose that one chooses the previous module* T$'$ *to build its own module-based DMS for scientific publications. Now, suppose that the person in charge of populating the module-based DMS stores doi_1 in the local* JournPaper *table, and its authors* ”SA” *and* ”OD” *in the local* hasAuthor *table, as illustrated in Figure 3.2.*

It is easy to see that though the module-based DMS is consistent, it is inconsistent together with the reference DMS: doi_1 is a journal paper in the local DMS, while it is a (full) conference paper in the reference DMS. This violates a constraint of the reference Tbox ([6.] in T*).*

Making the module T$'$ *robust to consistency checking requires adding integrity constraints for detecting inconsistency related to the relation of interest* JournPaper *and* hasAuthor*. From now on, I denote by* R$_{\mathrm{ref}}$ *a relation* R *from a reference Tbox that is not of interest for the module under consideration. It follows from [GR10, GR12] that an extension of* T$'$ *that is robust to consistency checking is:* {JournPaper $\sqsubseteq$ $\exists$hasAuthor, JournPaper $\sqsubseteq$ $\neg$ConfPaper$_{\mathrm{ref}}$, JournPaper $\sqsubseteq$ $\neg$FullPaper$_{\mathrm{ref}}$, JournPaper $\sqsubseteq$ $\neg$ShortPaper$_{\mathrm{ref}}$, Survey$_{\mathrm{ref}}$ $\sqsubseteq$ JournPaper}.

Note that these constraints bring relations into the module that are not of interest w.r.t. the application needs. With such a module, at global consistency checking time, the query to evaluate against the DMSs, which looks for a possible counter-example to any of these constraints, is: $q_{unsat}()$*:-* $[\exists x\ (\texttt{JournPaper}(x) \vee \texttt{JournPaper}_{\text{ref}}(x)) \wedge \texttt{FullPaper}_{\text{ref}}(x)] \vee \cdots$ *where the distributed evaluation is reflected in the names of the relations (remind that* $\texttt{R}$ *denotes a local relation, while* $\texttt{R}_{\text{ref}}$ *denotes the corresponding relation in the reference DMS). Here, the first disjunct in* $q_{unsat}()$ *looks for a possible counter-example to* $\texttt{JournPaper} \sqsubseteq \neg\texttt{FullPaper}_{\text{ref}}$*. The above* $q_{unsat}()$ *exhibits a global inconsistency due to* doi_1 *belonging to the local* $\texttt{JournPaper}$ *table of the module-based DMS and to the* $\texttt{FullPaper}_{\text{ref}}$ *table of the reference DMS.*

3.1.3 Robustness to query answering for global query answering

Robustness to query answering allows a module-based system complementing its answers to queries with the help of the reference DMS. This is particularly useful when the module-based DMS provides no or too few answers.

The basic idea is therefore to use the data of the reference DMS when queries are asked to the module-based DMS. Again, we do not want to import the whole reference DMS into our own DMS in order to do this. Instead, we extend the notion of module so that *global query answering* can be performed: We ensure that the module captures the constraints in the reference Tbox that are required to answer any query built upon the relations of interest, i.e., all the constraints that allow specializing the atoms of such queries. Then, at global query answering time, these constraints are used to identify the relevant data for a given query within the *distributed* Abox consisting of the Abox of the module-based DMS plus that of the reference DMS. From a practical viewpoint, this is achieved by a slight modification of the PerfectRef algorithm [CGL$^+$07] (cf. Section 2.2.3).

Example 13 (Continued) *Consider the conjunctive query* $Q(x)$*:-* $\texttt{JournPaper}(x) \wedge \texttt{hasAuthor}(x, "AH")$ *asking for the journal papers written by Alon Y. Halevy.*

Making $\texttt{T}'$ *robust to query answering requires adding inclusion constraints which allows exhibiting implicit tuples for the relation of interest* $\texttt{JournPaper}$ *and* $\texttt{hasAuthor}$*. It follows from [GR10, GR12] that an extension of* $\texttt{T}'$ *that is robust to query answering is:* $\{\texttt{JournPaper} \sqsubseteq \exists\texttt{hasAuthor}, \texttt{Publication}_{\text{ref}} \sqsubseteq \exists\texttt{hasTitle}_{\text{ref}}, \texttt{Publication}_{\text{ref}} \sqsubseteq \exists\texttt{hasAuthor}, \exists\texttt{hasTitle}_{\text{ref}} \sqsubseteq \texttt{Publication}_{\text{ref}}, \texttt{ConfPaper}_{\text{ref}} \sqsubseteq \texttt{Publication}_{\text{ref}}, \texttt{JournPaper} \sqsubseteq \texttt{Publication}_{\text{ref}}, \texttt{ShortPaper}_{\text{ref}} \sqsubseteq \texttt{ConfPaper}_{\text{ref}}, \texttt{FullPaper}_{\text{ref}} \sqsubseteq \texttt{ConfPaper}_{\text{ref}}, \texttt{Survey}_{\text{ref}} \sqsubseteq \texttt{JournPaper}\}$*. Again, such constraints bring relations into the module that are not of interest w.r.t. the application needs. With such a module, at global query answering time, the query to evaluate against the two DMSs is the perfect reformulation* $Q(x)$*:-* $[(\texttt{JournPaper}(x) \vee \texttt{JournPaper}_{\text{ref}}(x)) \wedge (\texttt{hasAuthor}(x, "AH") \vee \texttt{hasAuthor}_{\text{ref}}(x, "AH"))] \vee [\texttt{Survey}_{\text{ref}}(x) \wedge (\texttt{hasAuthor}(x, "AH") \vee \texttt{hasAuthor}_{\text{ref}}(x, "AH"))] \vee \cdots$ *which models all the ways to obtain answers from the distributed Aboxes. Here, the second disjunct in* $Q(x)$ *results from the inclusion constraint* $\texttt{Survey}_{\text{ref}} \sqsubseteq \texttt{JournPaper}$*. In particular,* $Q(x)$ *finds* doi_2 *and* doi_3 *as global answers, due to the presence in the reference DMS of:* doi_2 *in the* $\texttt{Survey}_{\text{ref}}$ *table,* $(doi_2, "AH")$ *in the* $\texttt{hasAuthor}_{\text{ref}}$ *table;* doi_3 *in the* $\texttt{JournPaper}_{\text{ref}}$ *table, and* $(doi_3, "AH")$ *in the* $\texttt{hasAuthor}_{\text{ref}}$ *table.*

3.2 Towards SPARQL query answering robust to RDF graph updates

Context Since 2011, I have been working with Ioana Manolescu (DR, Inria Saclay–Île-de-France) and Alexandra Roatiş (PhD student, Univ. Paris-Sud, co-advised by Ioana and me) on query answering robust to graph updates.

Our preliminary results were published in the proceedings of the Congrès francophone Reconnaissances des Formes et Intelligence Artificielle (RFIA) [GMR12b] and of the International World Wide Web Conference (WWW) [GMR12a] in 2012; all our results obtained so far have been submitted to the ACM Conference on Information and Knowledge Management (CIKM) in 2012.

Motivations Saturation-based query answering is the prevalent technique for answering SPARQL queries against a graph (cf. Section 2.1.3). It consists in pre-computing the saturation of the graph, i.e., materializing all its entailed triples, so that answers are then obtained by evaluating the queries against the saturation. While this technique typically leads to fast query run-time, saturation requires *time* to be pre-computed, *space* to be stored, and must be somehow *recomputed* upon graph updates. Yet, saturation-based query answering robust to graph updates has not received much attention in the literature [BK03, BKO$^+$11].

An alternative technique for answering queries against a graph is *reformulation-based query answering*. It consists in reformulating every query w.r.t. the queried graph into a *reformulated query*, which, evaluated against the (non-saturated) graph, yields the exact answer set. The point is that query reformulation is made at query run-time, i.e., w.r.t. the current state of the graph, making it de facto robust to graph updates. Reformulating queries takes typically little time at query run-time, however reformulated queries are often more *complex* (thus more *costly*) to evaluate than the original ones.

While saturation-based query answering takes into account the whole RDF and SPARQL recommendations, reformulation-based query answering is still confined to fragments of them. So far, it has been studied for the DL fragment of RDF (cf. Section 2.3.1) and the relational conjunctive queries [AGR07, CGL$^+$07, GOP11], and slight extension thereof [AGP09, GKLM11a, KMK08, UvHSB11], which consider the extension of the DL fragment where values can be both relations (classes/properties) and constants (blank nodes are however still disallowed), and BGP queries [KMK08, UvHSB11, GKLM11a] or SPARQL queries [AGP09].

Contributions In [GMR12b, GMR12a], we revisit the state-of-the-art saturation- and reformulation-based query answering techniques in the light of graph updates. We carry our investigations in our *DataBase (DB) fragment of RDF*, which extends the aforementioned ones notably with the support of incomplete information (i.e., blank nodes), and for the BGP queries.

Our main contributions are the design and comparison of saturation- and reformulation-based query answering techniques *robust to graph updates*:

1. For saturation-based query answering, we prove the correctness of a novel saturation maintenance technique allowing to limit the re-saturation effort upon graph updates: it yields the same graph as re-saturating from scratch.

2. For reformulation-based query answering, we first extends the state-of-the-art query reformulation algorithms to the DB fragment of RDF by adding the appropriate new reformulation rules, then we show how to evaluate the reformulated queries. The subtle point is that the correctness of the technique requires to evaluate the reformulated query in a non-standard fashion, as using the normative SPARQL evaluation would lead to unsound answer sets.

3. We compare our two techniques on well-established benchmarks (Barton[2], DBLP[3], and DBpedia[4]).

Importantly, both techniques have been carefully devised to be deployed on top of *any* RDF-tuned or off-the-shelf relational database management system (RDBMS), e.g., [AMMH07, NW10, WKB08], PostgreSQL[5], MySQL[6], Oracle[7], DB2[8],etc.

[2] http://simile.mit.edu/wiki/Dataset:_Barton
[3] http://thedatahub.org/dataset/fu-berlin-dblp
[4] http://wiki.dbpedia.org/Downloads37
[5] http://www.postgresql.org
[6] http://www.mysql.com
[7] http://www.oracle.com
[8] http://www-01.ibm.com/software/data/db2

In the following, I first introduce our DB fragment of RDF (Section 3.2.1), and then I give some intuitions on and I exemplify our saturation-based query answering technique with saturation maintenance (Section 3.2.2) and our reformulation-based query answering technique (Section 3.2.3).

3.2.1 The database fragment of RDF

We define the *database (DB) fragment of RDF* by *restricting entailment* (cf. Section 2.1) *to RDFS entailment*, i.e., to the entailment rules dedicated to RDF Schema shown in Figures 2.13–2.16 (page 16); *not restricting graphs in any way*, in other words, any triple that the RDF recommendation allows is also allowed in the DB fragment. We call a graph belonging to our DB fragment a *database*. The name of this fragment follows from the fact that both saturation- and reformulation-based query answering can be devised on top of *any* off-the-shelf or RDF-tuned RDBMS.

From now on, we focus on saturation- and reformulation-based query answering for *instance-level queries*, i.e., queries to be answered against the RDF statements of a (saturated) database only (i.e., against facts, not against RDFS statements modeling schema constraints). Observe that such queries are the standard database queries and the most common knowledge representation queries. To answer such queries, it suffices to consider the entailment rules in Figure 2.16 (page 16) [GMR12b, GMR12a].

3.2.2 Saturation-based query answering

Our first technique is based on the *multiset* saturation of a database, in which triples appear as many times as they can be derived.

This multiset saturation is the crux of our *saturation maintenance upon graph updates*. For *insertion* – adding triples to the database – and *deletion* – removing triples from the database –, our multiset saturation is maintained as follows: inserting a triple already in the database, or deleting a triple that is not in the database, does not affect the current multiset saturation; otherwise, inserting (deleting) a given triple also adds to (removes from) the current multiset saturation any RDF statement whose derivation uses this given triple.

Queries are then evaluated using this particular saturation to obtained their answers. To do so, the multiset saturation can be compactly represented as the usual saturation, i.e., a *set* of triples, in which every triple is tagged with (i) a boolean indicating whether it is either in the database or only entailed by the database and (ii) an integer indicating how many times it appears in the multiset saturation. Importantly, from a practical viewpoint, this compact representation allows delegating saturation-based query answering to *any* RDBMS: it can be stored in a `Triple` table (cf. Section 2.1.3) with two extra-columns, so as to answer queries using relational evaluation.

Example 14 (Saturation maintenance) *Consider again the RDF running example in Section 2.1. Let the graph* $\mathtt{G}'$ *be our database* `db`*. Its multiset saturation is shown, using its compact representation, in Figure 3.3, before and after deleting the RDFS statement* $\_{:}b_1$ rdfs:subClassOf confP *.; only the RDF statements are shown.*

An occurrence of doi_1 rdf:type confP . *has been deleted because* $\_{:}b_1$ rdfs:subClassOf confP ., doi_1 rdf:type $\_{:}b_0$. $\vdash_{\text{RDF}}$ doi_1 rdf:type confP .

An occurrence of doi_1 rdf:type paper . *has been deleted because of the above deletion and the fact that* confP rdfs:subClassOf paper ., doi_1 rdf:type confP . $\vdash_{\text{RDF}}$ doi_1 rdf:type paper .

It is worth noticing here that although removing the RDFS statement $\_{:}b_1$ rdfs:subClassOf confP . *changes the multiset saturation, it does not change the saturation. Indeed, the triples that have been removed still have derivations producing them, e.g.,*

- inProceedingsOf rdfs:domain confP ., doi_1 inProceedingsOf $\_{:}b_2$. $\vdash_{\text{RDF}}$ doi_1 rdf:type confP .
- hasTitle rdfs:domain paper ., doi_1 hasTitle "$CAQUMV$" . $\vdash_{\text{RDF}}$ doi_1 rdf:type paper .

3.2.3 Reformulation-based query answering

Our second technique is based on reformulating queries w.r.t. a database, so that their answer sets are obtained by evaluating their reformulations against the (non-saturated) database.

In our case, a query reformulation is a union of BGP queries built from the exhaustive applications of 13 reformulation rules. Each rule produces a new BGP query from a preexisting BGP query (the original one or an already produced one) either by binding a variable to class or property values found in the database, or by specializing a triple atom using an RDFS statement in the database.

Triple	Explicit	Times
doi$_1$ rdf:type _:b_0 .	true	1
doi$_1$ hasTitle "$CAQUMV$" .	true	1
doi$_1$ hasAuthor "SA" .	true	1
doi$_1$ hasContactA _:b_1 .	true	1
doi$_1$ inProceedingsOf _:b_2 .	true	1
_:b_2 hasName "$PODS'$98" .	true	1
cikm2012 rdf:type conference .	true	1
hasName createdBy "*John Doe*" .	true	1
_:b$_2$ rdf:type conference .	false	2
doi$_1$ rdf:type confP .	false	2
doi$_1$ rdf:type paper .	false	5
doi$_1$ hasAuthor _:b_1 .	false	1

Triple	Explicit	Times
doi$_1$ rdf:type _:b_0 .	true	1
doi$_1$ hasTitle "$CAQUMV$" .	true	1
doi$_1$ hasAuthor "SA" .	true	1
doi$_1$ hasContactA _:b_1 .	true	1
doi$_1$ inProceedingsOf _:b_2 .	true	1
_:b_2 hasName "$PODS'$98" .	true	1
cikm2012 rdf:type conference .	true	1
hasName createdBy "*John Doe*" .	true	1
_:b$_2$ rdf:type conference .	false	2
doi$_1$ rdf:type confP .	false	1
doi$_1$ rdf:type paper .	false	4
doi$_1$ hasAuthor _:b_1 .	false	1

Figure 3.3: Multiset saturation before (left) and after (right) deleting the triple _:b_1 rdfs:subClassOf confP .

	Query reformulation	Reformulation rule
1	$q(x, y)$:- x rdf:type y .	original query
2	$\cup$ $q(x,$ confP$)$:- x rdf:type confP .	from 1 by binding y to the class confP
3	$\cup$ $q(x,$ posterCP$)$:- x rdf:type posterCP .	from 1 by binding y to the class posterCP
4	$\cup$ $q(x,$ _:$b_0)$:- x rdf:type _:b_0 .	from 1 by binding y to the class _:b_0
5	$\cup$ $q(x,$ paper$)$:- x rdf:type paper .	from 1 by binding y to the class paper
6	$\cup$ $q(x,$ conference$)$:- x rdf:type conference .	from 1 by binding y to the class conference
7	$\cup$ $q(x,$ confP$)$:- x rdf:type posterCP .	from 2 since posterCP rdfs:subClassOf confP .
8	$\cup$ $q(x,$ confP$)$:- x rdf:type _:b_0 .	from 2 since _:b_0 rdfs:subClassOf confP .
9	$\cup$ $q(x,$ confP$)$:- x inProceedingsOf z .	from 2 since inProceedingsOf rdfs:domain confP .
10	$\cup$ $q(x,$ paper$)$:- x rdf:type confP .	from 5 since confP rdfs:subClassOf paper .
11	$\cup$ $q(x,$ paper$)$:- x hasTitle z .	from 5 since hasTitle rdfs:domain paper .
12	$\cup$ $q(x,$ paper$)$:- x hasAuthor z .	from 5 since hasAuthor rdfs:domain paper .
13	$\cup$ $q(x,$ conference$)$:- z inProceedingsOf x .	from 6 since inProceedingsOf rdfs:range conference .
14	$\cup$ $q(x,$ conference$)$:- x hasName z .	from 6 since hasName rdfs:domain conference .
15	$\cup$ $q(x,$ paper$)$:- x rdf:type posterCP .	from 10 since posterCP rdfs:subClassOf confP .
16	$\cup$ $q(x,$ paper$)$:- x rdf:type _:b_0 .	from 10 since _:b_0 rdfs:subClassOf confP .
17	$\cup$ $q(x,$ paper$)$:- x inProceedingsOf z .	from 10 since inProceedingsOf rdfs:domain confP .
18	$\cup$ $q(x,$ paper$)$:- x hasContactA z .	from 12 since hasContactA rdfs:subPropertyOf hasAuthor .

Figure 3.4: Reformulation of $q(x, y)$:- x rdf:type y . w.r.t. the database db.

However, in the DB fragment of RDF, query reformulation may bring into reformulated queries some blank nodes from the database, assuming that the reformulations refer precisely to these particular blank nodes in the database. The point is that the SPARQL/BGP query evaluation treats blank nodes as non-distinguished variables (cf. Section 2.1.2). As a result, evaluating our reformulations with the normative evaluation leads to unsound answer sets. The crux of our technique is therefore a *non-standard evaluation* of a query against a database, which treats blank nodes as constants. Importantly, from a practical viewpoint, the non-standard query evaluation of our reformulation-based query answering technique can be delegated to *any* RDBMS by storing the database in a `Triple` table (cf. Section 2.1.3), since non-standard and relational evaluations coincide when blank nodes are considered constants.

Example 15 (Query reformulation and non-standard evaluation) *Consider again the RDF running example in Section 2.1. Let the graph* G′ *be our database* db*.*

The reformulation of the query $q(x, y)$:- x rdf:type y . *w.r.t.* db*, asking for all resources and the classes to which they belong, is shown in Figure 3.4.*

Its normative evaluation against db *yields erroneous answers. For instance,* $\langle$cikm2012, confP$\rangle$ *is an erroneous answer resulting from the evaluation against* db *of the* 8^{th} *BGP query* $q(x,$ confP$)$*:-* x rdf:type _:b_0 *., with the assignment* $\mu = \{x \rightarrow$ cikm2012, _:$b_0 \rightarrow$ conference$\}$.

The issue here is that, when $q(x,$ confP$)$*:-* x rdf:type _:b_0 . *is produced from* _:b_0 rdfs:subClassOf confP . $\in$ db *and the* 2^{nd} *BGP query* $q(x,$ confP$)$*:-* x rdf:type confP *., the goal is to find conference paper values for x from the subclass* _:b_0 *of* confP*. However, under the normative query evaluation,* _:b_0 *is treated as a non-distinguished variable, thus the produced query returns every class instance stored in* db *as a conference paper, while under the non-standard query evaluation it only returns the instances for the class* _:b_0 *stored in* db*.*

In fact, the answer set of q against db *is:* $\{\langle$doi$_1$, _:$b_0\rangle$, $\langle$cikm2012, conference$\rangle$, $\langle$doi$_1$, confP$\rangle$, $\langle$doi$_1$, paper$\rangle$, $\langle$_:b_2, conference$\rangle\}$.

Chapter 4

Optimization

The main contribution I present is *view selection for efficient SPARQL query answering* (Section 4.1). It amounts to tuning an RDF data management system to users or applications' needs modeled as a query workload. The idea is to pre-compute and store the results for some queries *automatically* selected – the *views* – in order to minimize a combination of query processing, view storage, and view maintenance upon update costs.

The ongoing work I present next is a first step towards *efficient query answering against XML documents with RDF annotations* (Section 4.2). We propose to combine the XML and RDF data models and languages into a uniform XML-RDF hybrid setting for managing annotated documents. In particular, we want to study to which extent query answering can be optimized by using at the same time the structural XML constraints (expected tree shape of the documents) and the semantic RDF constraints (expected ontological descriptions) expressed in queries.

4.1 View selection in RDF

Context I have been working on view selection in RDF with Ioana Manolescu (DR Inria Saclay–Île-de-France) and Konstantinos Karanasos (PhD student, Inria, co-advised by Ioana and me) since 2009, and also with Julien Leblay (PhD student, Univ. Paris-Sud, co-advised by Ioana and me) since 2010.

Our preliminary results were published in the proceedings of the Journées Bases de données Avancées (BDA) [GKLM10b] in 2010; all our results obtained so far have been published in the journal Proceedings of the VLDB endowment (PVLDB) [GKLM11a] in 2011. The RDFViewS prototype[1] implementing our results was demonstrated at the International Conference on Information and Knowledge Management (CIKM) [GKLM10a] in 2010 and at Journées Bases de données Avancées (BDA) [GKLM11b] in 2011.

Motivations A basic requirement of any data management system is to provide efficient query processing. For instance, well-established query answering optimizations resort to indexes and materialized views [RG03].

Indexes are precomputed data structures that allow traversing data efficiently. For example, the *B-Tree* indexing commonly used in relational database management systems allows traversing data in logarithmic time. The use of indexes for efficient SPARQL query processing on RDF graphs has received significant attention in the literature [AMMH07, NW08, SGK^{+}08, WKB08, NW09].

Materialized views are queries whose answer sets have been precomputed and stored, so as to be efficiently accessed. [Hal01]. A first option for answering queries using materialized view is *query rewriting*. In this case, given a query and some views, the goal is to rewrite this query into an equivalent (or maximally contained) one that refers to these views only. The answer set of the original query is then obtained efficiently by evaluating the rewritten query against the materialized views. Query rewriting for SPARQL query answering has not received much attention yet. However, some of my results [AGR09, AGR10], obtained in the setting of Description Logics, directly transfer to the DL fragment of RDF and the relational conjunctive queries of SPARQL (cf. Section 2.3.1). Another option for answering queries using materialized view is *view selection*. The idea is to tune query processing w.r.t. users or applications' needs modeled by a query workload. In this case, the goal is to pre-compute from the workload the views to materialize together with the query rewritings, in order to minimize a combination of

[1] `http://tripleo.saclay.inria.fr/rdfvs`

rewriting processing, view storage, and view maintenance upon update costs. At query run-time, the answer sets of workload queries are efficiently obtained by evaluating the corresponding rewritings against the materialized views. View selection for SPARQL query *evaluation*, i.e., *without* taking into account RDF entailment, has been investigated in [CL10, CRL10] to speed up the processing of *some* workload queries.

Contributions In [GKLM10b, GKLM10a, GKLM11b, GKLM11a], we revisit the view selection approach of [TS97, TLS01] for the relational model and the conjunctive queries/views, in the light of the RDF data model and the SPARQL query language. Notably, we devise a view selection technique for SPARQL query *answering*, i.e., taking into account RDF entailment, to speed up the processing of *all* workload queries. More specifically, our main contributions are:

1. We investigate the view selection problem for the so-called *plain* RDF (i.e., no RDF entailment) and the BGP queries of SPARQL. In particular, adopting the elegant framework of [TS97, TLS01], this problem is formalized as a search problem in a space of states, modeling some views to materialize and the rewritings of the *whole* query workload w.r.t. these views.

2. We then extend our view selection technique to RDFS entailment, by considering the two options of graph saturation and query reformulation. It turns out that our technique requires no special adaptation if applied to a saturated graph; for query reformulation, we focus on the DL fragment of RDF and propose an algorithm for either *pre-reformulation* of the query workload or *post-reformulation* of the views selected for materialization.

3. We also consider heuristic search strategies, since the complexity of complete search is extremely high. In particular, existing strategies for relational view selection [TS97, TLS01] grow out of memory and fail to produce a solution when the size of the query workload increases. Since triple atoms of BGP queries are short (just three attributes), BGP queries typically have many atoms, making this scale problem in RDF particularly acute. We propose a set of new strategies and heuristics which greatly improve the scalability of the search.

4. We experimentally demonstrate the effectiveness and the efficiency of the above techniques through thorough extensive experiments.

In the following, I give some intuition on and I exemplify our view selection technique for plain RDF (Section 4.1.1) and how RDFS entailment can be incorporated into it (Section 4.1.2).

4.1.1 View selection for plain RDF and BGP queries

As mentioned above, we model our view selection problem as a search problem in a space of states.

A state S is a pair $\langle V, R\rangle$, where V models a set of candidate views for materialization, and R is a set of rewritings in terms of V only for the given input BGP query workload Q. A state has a cost estimation $c^{\varepsilon}(S)$ that corresponds to a combination of rewriting processing, view storage, and view maintenance upon update costs. The point here is that the cost estimation is computed according to the (non-saturated) graph from which views have to be materialized. Our view selection problem is therefore to find within the space of all possible states w.r.t. a given workload Q, a state S whose cost is less than or equal to that of any other state S'.

In fact, we proceed as follows. We define the initial state for a query workload Q as the trivial state in which every workload query is a candidate view and every candidate rewriting is a view. Then, we define a set of 4 transitions that allows building from this initial state other ones, and so on, so as to explore/generate *all* the possible sets of candidate views/rewritings. The idea of these transitions is to transfer constraints expressed in views into rewritings, in order to exhibit some commonalities between these views: when two views become equivalent, they can be fused.

Example 16 (Search space exploration with our 4 transitions) *Consider the query workload Q made of the single query $q_1(x,z)$:- x hasPainted "SN" .,x isParentOf y .,y hasPainted z ., asking for painters that have painted "Starry Night" and having a child that is also a painter, as well as the paintings of their children.*

The initial state for this workload is: $S_0 = \langle\{v_1 = q_1\}, \{r_1 = v_1\}\rangle$

Applying a View Break on S_0 may lead to a state $S_1 = \langle\{v_2, v_3\}, \{r_1(x,z)\text{:- } v_2(x,y) \wedge v_3(x,y,z)\}\rangle$ with $v_2(x,y)$:- x hasPainted "SN" .,x isParentOf y . and $v_3(x,y,z)$:- x isParentOf y .,y hasPainted z . The idea *here is that the view v_1 that has been broken, by moving from S_0 to S_1, is simulated in the rewriting of q_1 in S_1 by joining v_2 and v_3.*

Applying now a Selection Cut on S_1 may lead to a state $S_2 = \langle\{v_3, v_4\}, \{r_1(x, z)\text{:- } v_4(x, y, ''SN'') \wedge v_3(x, y, z)\}\rangle$ with $v_4(x, y, w)$:- x hasPainted w ., x isParentOf y . The idea here is that the constant $''SN''$ that has been removed from v_2, by moving from S_1 to S_2, is pushed to the rewriting of q_1 in S_2 by enforcing the selection of $''SN''$ for the w variable of v_4.

Applying now a Join Cut on S_2 may lead to a state $S_3 = \langle\{v_4, v_5, v_6\}, \{r_1(x, z)\text{:- } v_4(x, y, ''SN'') \wedge v_5(x, y) \wedge v_6(y, z)\}\rangle$ with $v_5(x, y)$:- x isParentOf y . and $v_6(y, z)$:- y hasPainted z . The idea here is that the join on y that has been removed from v_3, by moving from S_2 to S_3, is pushed to the rewriting of q_1 in S_3.

Applying now a Join Cut on S_3 may lead to a state $S_4 = \langle\{v_5, v_6, v_7, v_8\}, \{r_1(x, z)\text{:- } v_7(x, ''SN'') \wedge v_8(x, y) \wedge v_5(x, y) \wedge v_6(y, z)\}\rangle$ with $v_7(x, w)$:- x hasPainted w . and $v_8(x, y)$:- x isParentOf y . The idea here is that the join on x that has been removed from v_4, by moving from S_3 to S_4, is pushed to the rewriting of q_1 in S_4.

Applying now a View Fusion on S_4 may lead to a state $S_5 = \langle\{v_6, v_7, v_9\}, \{r_1(x, z)\text{:- } v_7(x, ''SN'') \wedge v_9(x, y) \wedge v_6(y, z)\}\rangle$ with $v_9(x, y)$:- x isParentOf y .

Applying now a View Fusion on S_5 may lead to a state $S_6 = \langle\{v_9, v_{10}\}, \{r_1(x, z)\text{:- } v_{10}(x, ''SN'') \wedge v_9(x, y) \wedge v_{10}(y, z)\}\rangle$ with $v_{10}(x, z)$:- x hasPainted z .

4.1.2 View selection for RDF and BGP queries

I now discuss possible ways to take RDF entailment into account in our view selection technique.

Graph saturation. We reduce the problem of view selection with RDF entailment to that of view selection for plain RDF. The idea is to saturate, *prior* to the search, the graph w.r.t. which views have to be materialized, so that the cost estimations of states account for the explicit and entailed triples. While this solution is simple, it requires to saturate the whole graph w.r.t. which views have to be materialized, thus may overgrow some space constraints.

Workload-reformulation. We also reduce the problem of view selection with RDF entailment to that of view selection for plain RDF. However, we consider the DL fragment of RDF (cf. Section 2.3.1), i.e., RDFS entailment only. The idea is to reformulate the query workload *prior* to the search, leaving unchanged the graph w.r.t. which views have to be materialized. Here, query reformulation is achieved by a preliminary query reformulation technique of that presented in Section 3.2. – Of course, our more recent algorithm in Section 3.2 could now lift our view selection technique to the DB fragment of RDF and BGP queries. – The subtle point here is that the rewriting language has to be extended to that of *union* of BGP queries; given a set of queries $Q = \{q_1, \ldots, q_n\}$, and assuming that $q_i^1, \ldots, q_i^{n_i}$ are the reformulations of q_i, the initial state is now $S_0 = \langle V_0, R_0\rangle$ with $V_0 = \bigcup_{i=1}^n \{q_i^1, \ldots, q_i^{n_i}\}$ and $R_0 = \bigcup_{i=1}^n \{r_i = q_i^1 \cup \cdots \cup q_i^{n_i}\}$. While this solution is also quite simple, the size of the reformulated workload may overgrow the size for which view selection can scale, as the size of the workload is a very sensitive parameter in the size of the search space. So, from a practical viewpoint, the bigger the workload, the smaller the search space effectively investigated in a reasonable amount of time.

View-reformulation To prevent the cons of the two above solutions (i.e., a large saturated graph or a large query workload), the idea is to perform view selection for a non-reformulated workload and a non-saturated graph. Once a best state has been found, we reformulate the views selected for building the query rewritings, so as to obtain the right view materialization against the non-saturated graph. However, the technique is not so simple, since the correctness of view selection imposes that the cost estimations of states must not reflect a combination of rewriting processing, view storage, and view maintenance upon update costs w.r.t. *the views* of states, but w.r.t. *their reformulations*.

Example 17 (View selection techniques) *Let us consider the workload query $q(x_1, x_2)$:- x_1 rdf:type picture ., x_1 isLocatIn x_2 . to be optimized for a graph whose schema is:* $\mathtt{S} = \{$painting rdfs:subClassOf picture ., isExpIn rdfs:subPropertyOf isLocatIn .$\}$.

Our cost estimation framework counts the exact number of triples in the graph that match the two query atoms $q^1(x_1)$:- x_1 rdf:type picture . and $q^2(x_1, x_2)$:- x_1 isLocatIn x_2 . as well as the triples matching three relaxed atom queries*, obtained by removing the constants from q^1 and q^2: $q^3(x_1, x_2)$:- x_1 rdf:type x_2 ., $q^4(x_1, x_2)$:- x_1 x_2 picture ., and $q^5(x_1, x_2, x_3)$:- x_1 x_2 x_3 . That way, we obtain the* exact *cardinalities of atoms that may occur in views during the search. Observe that relaxed atoms are produced during the search by some state*

q^1	$q^1(x_1)$:- x_1 rdf:type picture . $\cup$ $q^1(x_1)$:- x_1 rdf:type painting .
q^4	$q^4(x_1, x_2)$:- x_1 x_2 picture . $\cup$ $q^4(x_1, \text{isLocatIn})$:- x_1 isLocatIn picture . $\cup$ $q^4(x_1, \text{isExpIn})$:- x_1 isExpIn picture . $\cup$ $q^4(x_1, \text{rdf:type})$:- x_1 rdf:type picture . $\cup$ $q^4(x_1, \text{isLocatIn})$:- x_1 isExpIn picture . $\cup$ $q^4(x_1, \text{rdf:type})$:- x_1 rdf:type painting .

Table 4.1: Sample atom query reformulations for post-reasoning.

transitions (selection cuts and join cuts). These cardinalities are the basic ingredient to estimate *the costs of states, i.e., combinations of rewriting processing, view storage, and view maintenance upon update costs.*

With view selection based on graph saturation, we first compute the above cardinalities from the saturated graph, then we perform the search for a best state. Assume that for the above query q above, there is a unique best state $S = \langle V, R \rangle$ *resulting from the initial one by a join cut followed by a selection cut, with:*

- $V = \{v_4(x_1, x_3)$:- x_1 rdf:type x_3 ., $v_3(x_1, x_2)$:- x_1 isLocatIn x_2 .$\}$
- $R = \{r_1(x_1, x_2)$:- $v_4(x_1, \text{picture}) \wedge v_3(x_1, x_2)$

With view selection based on workload pre-reformulation, we first compute the above cardinalities from the non-saturated graph, then we perform the search with the initial state $S_0 = \langle V_0, R_0 \rangle$ *where:*

- $V_0 = \{$
  $v_1^1(x_1, x_2)$:- x_1 rdf:type picture ., x_1 isLocatIn x_2 ., $v_1^2(x_1, x_2)$:- x_1 rdf:type painting ., x_1 isLocatIn x_2 ., $v_1^3(x_1, x_2)$:- x_1 rdf:type picture ., x_1 isExpIn x_2 ., $v_1^4(x_1, x_2)$:- x_1 rdf:type painting ., x_1 isExpIn x_2 .$\}$
- $R_0 = \{r_1(x_1, x_2)$:- $\bigcup_{i=1}^{4} v_1^i(x_1, x_2)\}$

Observe that for a same workload, the solutions of view selection based on graph saturation or workload pre-reformulation may differ, as reformulation introduces values that may not be in the original workload.

With view selection based on workload post-reformulation, we first compute the above cardinalities from the reformulation of the above atom queries, so that their evaluation against the non-saturated graph provides the correct numbers. Table 4.1 illustrates the obtained reformulations for q^1 *and* q^4*. Then, we perform the search for a best state which returns the same solution as view selection based on graph saturation. However, after the search has finished, this technique must reformulate the selected views to get the correct materialization. Here, instead of materializing the selected views* v_4 *and* v_3*, we materialize their reformulations:*

- *For* v_4*:* $v_4^1(x_1, x_3)$:- x_1 rdf:type x_3 . $\cup$ $v_4^2(x_1, \text{painting})$:- x_1 rdf:type painting .
  $\cup$ $v_4^3(x_1, \text{picture})$:- x_1 rdf:type picture . $\cup$ $v_4^4(x_1, \text{picture})$:- x_1 rdf:type painting .
- *For* v_3*:* $v_3^1(x_1, x_2)$:- x_1 isLocatIn x_2 . $\cup$ $v_3^2(x_1, x_2)$:- x_1 isExpIn x_2 .

4.2 Towards efficient querying of XML documents with RDF annotations

Context Since 2011, I have been working with Ioana Manolescu (DR Inria Saclay–Île-de-France), Konstantinos Karanasos (PhD student, Inria, co-advised by Ioana and me), Julien Leblay (PhD student, Univ. Paris-Sud, co-advised by Ioana and me), Yannis Katsis (Postdoc, Inria), and Stamatis Zampetakis (Master student, co-advised by Ioana and me) on managing XML documents with RDF annotations.

Our preliminary results on defining a hybrid XML-RDF data model and query language for annotated documents were published in the proceedings of the Journées Bases de données Avancées (BDA) [GKK$^+$11b] and of the Very Large Data Search (VLDS) Workshop [GKK$^+$11a] in 2011. A journal version of these results will be published in the french journal Ingénierie des Systèmes d'Information (ISI) [GKK$^+$12] in 2012. All the results obtained so far on this topic will be submitted to a special issue of the VLDB Journal in 2012.

Motivations XML has been widely adopted to represent Web data. Indeed, a plethora of XML documents can be found on the Web, modeling tree databases, XHTML pages, RSS feeds, SVG images or animations, ODF documents (e.g., Word- or Excel-like), etc. Exploiting this mass of freely available documents has become a challenge in the industry for economic watch, strategic intelligence or journalism; in politics for geopolitical or diplomatic strategies; in security for police or military surveillance. The key for using such documents is to annotate them using ontologies, so that they can be categorized, classified, and easily retrieved w.r.t. relevant criteria.

The Semantic Web standards RDF and OWL provides a general normative way for annotating Web resources by assigning them URIs, so as to give them ontological descriptions. However, annotating at the level of XML documents is *not fine-grained enough* in many applications, which require annotating within documents. However, when the documents to annotate are *XHTML pages*, one can use of RDFa [W3Cb], a W3C standard that allows annotating any part of XHTML pages with RDF descriptions ; when the documents to annotate are *ODF documents*, one can use of the ODF 1.2 standard by OASIS[2] that allows annotating office documents with some restricted RDF descriptions.

Contributions In [GKK$^+$11b, GKK$^+$11a, GKK$^+$12], we revisit the ideas developed within the WebContent project[3] [AAC$^+$08], a 2006-2009 ANR "Réseau National de Technologies Logicielles" project in which I have participated, whose goal was to integrate (in a somewhat ad hoc fashion) off-the-shelf technologies from the partners in order to manage XML documents with RDF annotations. Our main contributions are the design of a clear and clean formal setting for representing and querying annotated documents, as well as some query answering optimizations:

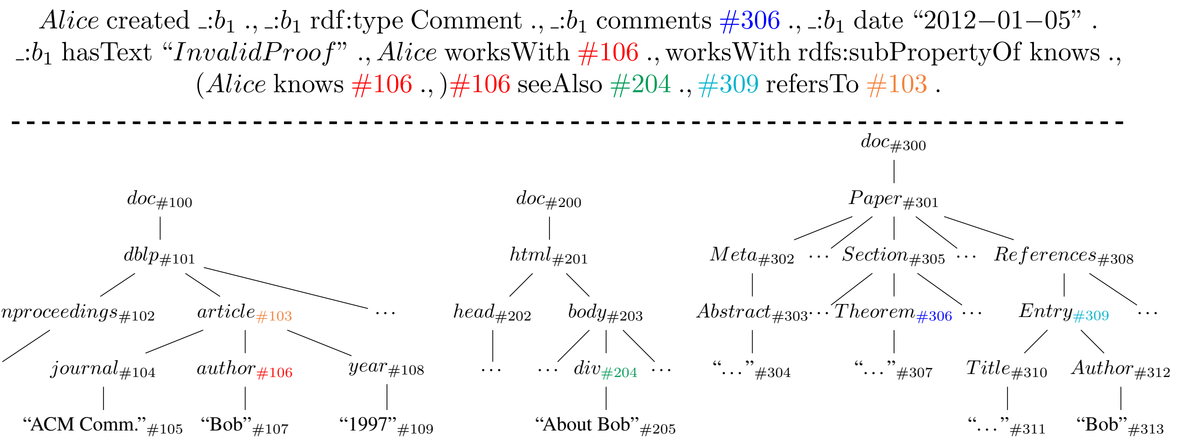
1. We define an XML-RDF hybrid framework, called XR, for managing XML documents with fine-grained RDF annotations. In particular, we allow putting *any RDF description* on *any piece (a.k.a. node) of an XML document*. Hence, we can have RDF annotations within tree databases, RSS feeds, XHTML pages, ODF documents, etc. Observe that this setting encompasses that of the RDFa standard by W3C or of the ODF 1.2 standard by OASIS.

2. To query uniformly our annotated documents, we define a query language, called XRQ, integrating the Tree-Pattern Queries (TPQs) [AYCLS01] of the standard XML query language XQuery [W3Ce], and the BGP queries of SPARQL. Hence, we allow querying annotated documents w.r.t. both their structure (with tree-shape constraints) and their semantics (with ontological constraints).

3. We also investigate how such an uniform XML-RDF setting can be used for optimizing query answering against annotated documents, by pushing some information from XML to RDF, or RDF to XML, during query processing.

In the following, I give some intuitions on and I exemplify first our XR data model and XRQ query language for annotated documents (Section 4.2.1), then some optimizations under investigations for XRQ query answering against XR documents (Section 4.2.2).

[2] `http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=office`
[3] `http://www.webcontent.fr`



$Alice$ created $\_{:}b_1$., $\_{:}b_1$ rdf:type Comment ., $\_{:}b_1$ comments #306 ., $\_{:}b_1$ date "2012−01−05" .
$\_{:}b_1$ hasText "$InvalidProof$" ., $Alice$ worksWith #106 ., worksWith rdfs:subPropertyOf knows .,
($Alice$ knows #106 .,)#106 seeAlso #204 ., #309 refersTo #103 .

Figure 4.1: Sample XR instance.

4.2.1 The XR data model and XRQ query language

The XR data model combines the XML and RDF data models in order to represent annotated documents. An XR instance comprises an XML and an RDF sub-instances. A specific URI is assigned to each node of the XML sub-instance, so that the RDF sub-instance can describe any part of an XML document in the XML sub-instance. The semantics of an XR instance, is that of the XML one plus that of the RDF one (i.e., its saturation).

The XRQ query language allows writing queries against annotated documents modeled in an XR instance. An XRQ query is made of a head and a body. The body, like an XR instance, is a made of two parts: a set of TPQs expressing structural tree constraints (e.g., child or descendant relationships), and a BGP query (i.e., a set of triple patterns) expressing the ontological constraints. Tree and triple patterns may contains variables, joins are expressed by reusing variables in multiple places of a query. In particular, joins can be expressed between tree and triple patterns. The head of an XRQ query is a set of variables appearing in the body and possibly constants. For answering such queries, tree patterns are answered against the XML sub-instance and triple patterns are answered against the RDF sub-instance, producing a set of variable bindings. The answers of a XRQ query corresponds to the tuples obtained by applying the bindings satisfying the body of the query to the query's head.

Example 18 (XR instance and XRQ query) *Consider the XR instance depicted in Figure 4.1, which consists of XML documents and their RDF annotations. The upper part shows the triples of the RDF sub-instance, while the XML sub-instance in the lower part shows node-labeled unranked ordered trees (i.e., XML documents). Node labels represent element (or attribute) tag names and edges stand for parent-child relationships. The XML sub-instance contains 3 documents, the first one is a database about DBLP bibliographical data, the second one is an XHTML homepage, and the third one is a research paper written in an open document format. Observe that every document has been assigned a URI, denoted by* $\#number$ *in the root* $doc_{\#number}$ *of the documents, as well as every node of every document, denoted by* $\#number$ *in the node tags* $\mathrm{tag}_{\#number}$ *of the documents. The RDF sub-instance specifies that Alice has created a* comment *on a Theorem of the research paper. This comment comes with a* date *and a* text*. The paper Alice has annotated* refers to *the journal paper in DBLP whose URI is #103. Alice* knows *the author of this article. This can be inferred, through RDF entailment (e.g., using saturation), from the fact that she* works with *him, and that* working with *someone is a subproperty of* knowing *someone. Finally, additional information (*seeAlso*) on this author can be found on a particular location, identified by the URI #204, of the XHTML page.*

Consider now the XRQ query depicted on Figure 4.2. The upper part of the body shows the BGP sub-query, while the lower part show the XQuery sub-query made of TPQs. Single edges are parent/child relationships; double edges are descendant relationships. A node has a uri, a val (i.e., the concatenation of the values of its descendants), and a cont (i.e., the subtree it is the root of). This query looks for an author of a 1997 journal paper in the DBLP database $doc_{\#100}$*, with a piece of XHTML page describing her, and a paper of whom is referenced in the research paper* $doc_{\#300}$*, such that this author is known by someone that has created a comment on a theorem of this research paper. It returns the creator of the comment, the XHTML description of the cited author, and the text of the commented theorem.*

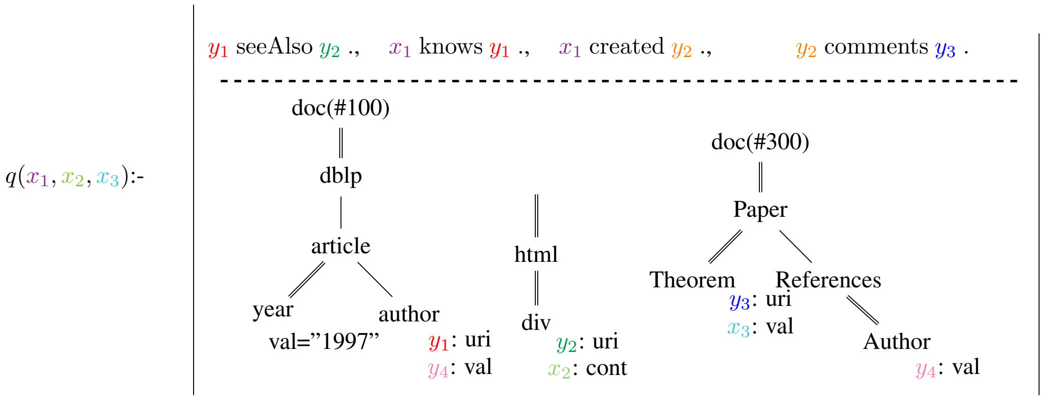


Figure 4.2: Sample XRQ query.

4.2.2 Optimization for XRQ query answering against XR documents

To answer XRQ queries against an XR instance, we are investigating three strategies.

1. The first option is to answer in parallel the XQuery sub-query of the XRQ query against the XML sub-instance and the BGP sub-query of the XRQ query against the RDF sub-instance, before joining the obtained variable bindings.

2. The second option is to answer the XQuery sub-query of the XRQ query against the XML sub-instance, and then to iterate on the obtained variable bindings to push them into the BGP sub-query of the XRQ query. This results into a union of more specific BGP queries to be evaluated against the RDF sub-instance.

3. The third option is to answer the BGP sub-query of the XRQ query against the RDF sub-instance, and then to iterate on the obtained variable bindings to push them into the XQuery sub-query of the XRQ query. This results into a union of more specific XQuery queries to be evaluated against the XML sub-instance.

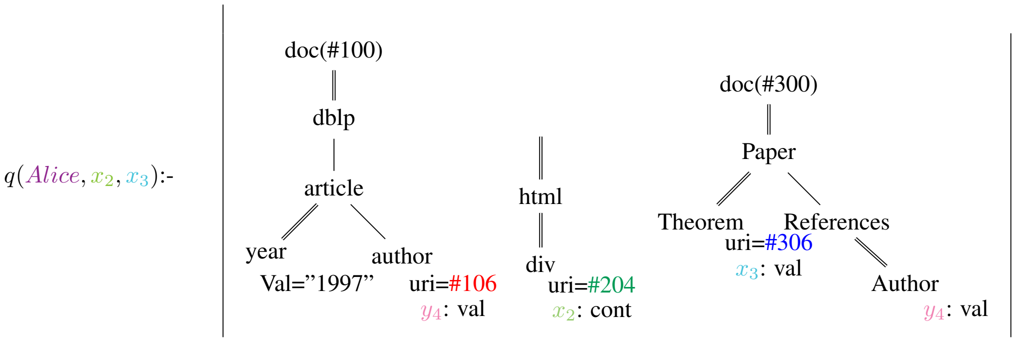
The choice of using one of these strategies is guided by a cost estimation function which, based on statistics collected from the XR instance, is able to estimate the number of variables bindings returned by the two subqueries of an XR query. If none of them is very selective, i.e., returns few variable bindings, the first strategy is a safe choice as the price to pay for joining the subqueries' results may be compensated by the parallel processing of the two subqueries. Otherwise, the other options may be used to evaluate first the most selective query, so as to push its few variables bindings to the other sub-query, making a small union of more selective queries whose processing is expected to traverse much less of the sub-instance.

Example 19 (Optimization strategies for XRQ query processing) *Consider the above examples of an XR instance and an XRQ sub-query.*

With the first query processing strategy, answering in parallel the XQuery and BGP subqueries yields respectively the variable bindings $\{y_1 = \#106, y_4 = ''Bob'', y_2 = \#204, x_2 = ''\cdots'', y_3 = \#306, x_3 = ''\cdots''\}$ *and* $\{y_1 = \#106, y_2 = \#204, x_1 = Alice, y_2 = \_{:}b_1, y_3 = \#306\}$*. Joining these results on the shared variables yields the variable bindings of the whole XRQ query* $\{x_1 = Alice, x_2 = ''\cdots'', x_3 = ''\cdots'', y_1 = \#106, y_2 = \#204, y_3 = \#306, y_4 = ''Bob''\}$*, out of which the query answers are projected.*

With the second query processing strategy, the XQuery sub-query is answered first and yields a single set of variable bindings: $\{y_1 = \#106, y_4 = ''Bob'', y_2 = \#204, x_2 = ''\cdots'', y_3 = \#306, x_3 = ''\cdots''\}$*. This set is then used to bind the corresponding variables of the BGP sub-query, leading to the new more selective BGP query* $q(x_1, ''\cdots'', ''\cdots'')$*:-* #106 seeAlso #204 ., x_1 knows #106 ., x_1 created #204 ., #204 comments #306 . *whose answers are these of the XRQ query.*

With the third query processing strategy, the BGP sub-query is answered first and yields a single set of variable bindings: $\{y_1 = \#106, y_2 = \#204, x_1 = Alice, y_2 = \_{:}b_1, y_3 = \#306\}$*. This set is then used to bind the corresponding variables of the XQuery sub-query, leading to the following more selective XQuery query, whose answers are these of the XRQ query.*


q(Alice, x2, x3):-
doc(#100)
dblp
article
year
Val="1997"
author
uri=#106
y4: val
html
div
uri=#204
x2: cont
doc(#300)
Paper
Theorem
References
uri=#306
x3: val
Author
y4: val

Chapter 5

Decentralization

The main contribution I present is *peer-to-peer data management for RDF and DL-lite* (Section 5.1). In such systems, every peer manages its own ontology and data, and can establish semantic correspondences called *mappings* with peers having similar interests. This gives rises to a fully distributed data management system, in which it becomes possible to perform global data management tasks. The idea is to decentralize state-of-the art data management algorithms, by resorting to decentralized consequence finding in a distributed propositional theory.

The ongoing work I present next is a very first step towards *RDF data management in a cloud* (Section 5.2). A cloud is a place where one can rent virtual machines, disk space, and services (e.g., database access), and then pay as she uses them. The focus is on SPARQL query answering against RDF graphs in the Amazon cloud, in the light of efficiency and monetary costs.

5.1 P2P Systems for propositional logic, RDF and DL-lite

This work comprises two seemingly unrelated research studies on *peer-to-peer inference systems for propositional logic* (Section 5.1.1) and on *peer-to-peer data management systems for RDF and DL-lite* (Section 5.1.2). In fact, they are tightly connected as the latter builds on the former.

5.1.1 P2P inference systems for propositional logic

Context I have worked on peer-to-peer inference systems in propositional logic from 2003 to 2009 with Serge Abiteboul (DR Inria Saclay–Île-de-France), Philippe Chatalic (MCF, Univ. Paris-Sud), Marie-Christine Rousset (PR, Univ. Grenoble), Laurent Simon (MCF, Univ. Paris-Sud), Philippe Adjiman (former PhD student, Univ. Paris-Sud, co-advised by Marie-Christine and me), and Nada Abdallah (former PhD student, Inria, co-advised by Serge and me).

Our preliminary milestone results were published in the proceedings of the Journées Nationales sur la résolution Pratique de Problèmes NP-Complets (JNPC) [ACG$^+$04b] and of the European Conference on Artificial Intelligence (ECAI) [ACG$^+$04a] in 2004, of the International Joint Conference on Artificial Intelligence (IJCAI) [ACG$^+$05a] in 2005, of the Journées Bases de Données Avancées (BDA) [AG08b] and of Journées Francophones de Programation par contraintes (JFPC) [AG08a] in 2008. Completed results were published in Journal on Artificial Intelligence Research (JAIR) [ACG$^+$06] in 2006, and AI Communications (AICOM) [AG09] in 2009.

Motivations P2P systems emerged in the beginning of the 21^{th} century from the very practical applications of file sharing over the Internet. They provide a flexible and scalable architecture in which every peer is autonomous, i.e., it manages its own files, and can query other peers it knows to get files whenever a requested file is not stored locally. That way, given a file request made to a peer, this peer can gradually solicit others to traverse the system and hopefully find the requested file.

Adapting this vision to AI allows envisioning *fully* decentralized inference systems where peers have their own knowledge and collaborate with others to reason about the global knowledge, seen as the virtual union of the individual knowledge of the peers. This contrasts with the traditional centralized inference systems [RN10] and their seemingly decentralized optimizations, e.g., using partition-based reasoning [AM00, AM01], that where under investigations when P2P systems arose.

Contributions In [ACG+04b, ACG+04a, ACG+05a, ACG+06, AG08b, AG08a, AG09], we introduce the propositional P2P inference systems (P2PISs) in which we investigate the standard task of *consequence finding* and for which we introduce non-standard task of *conservative extension checking of a peer*. Consequence finding is a core reasoning task in AI, as many other tasks build on it, e.g., common sense reasoning, diagnosis, or knowledge compilation [Mar00]. Conservative extension checking, which is strongly related to the notion of forgetting, becomes central in P2PISs as it allows peers to complement their knowledge about their own application domain or area of expertise.

More specifically, our contributions are:

1. We define a propositional P2PIS as a distributed clausal theory[1] of propositional logic (PL), in which each peer manages is own local clausal theory in terms of its own variables (a.k.a. alphabet), plus some mappings that are clauses involving literals from at least a remote peer. These remote literals define the remote peers with which the peer can communicate.

2. We provide the first fully decentralized consequence finding algorithm, called DECA, for these P2PISs. Given an input clause in terms of the variables of a single peer, DECA computes clausal consequences of the P2PIS plus this input clause, including all the strongest ones following from the input clause. DECA is based on the Resolution principle [Rob65, CR73] and basically uses remote literals to propagate deduction (a.k.a. derivations) gradually within the P2PIS. The space, time, and communicational complexity of DECA have also been analyzed.

3. We provide the SOMEWHERE P2P platform that implements DECA. SOMEWHERE has been used to study experimentally the scalability of DECA, which has been demonstrated up to a thousand peers. The SOMEWHERE prototype is currently entirely reengineered (thanks to an Inria grant "Action de Développement Technologique"), so as to make it available to the research community in late 2012.

4. We exhibit the importance of the notion of (non-)conservative extension in the setting of P2PISs: a peer basically *extends* its theory (using mappings) with the rest of the P2PIS (the peer theories and mappings), so that it can also use their knowledge when performing reasoning tasks; a well-known result is that the extension of a theory is not necessarily conservative. Hence, when a peer extends its theory by establishing mappings, a P2PIS may provide knowledge that the peer itself does not know, while this extra knowledge is in terms of its *own* alphabet, thus about its *own* application domain or area of expertise. We therefore provide a conservative extension checking algorithm for the peers of our P2PISs, called CECA, so that any of them can answer the question: What do the P2PIS say in my name (using my alphabet only), that I do not already now?

In the following, I first introduce our propositional P2PISs (Section 5.1.1.1), and then I give some intuitions on and I exemplify our decentralized consequence finding technique (Section 5.1.1.2) and our conservative extension checking technique (Section 5.1.1.3).

5.1.1.1 P2PISs

We define a P2PIS $\mathcal{S} = \{\mathcal{P}_i\}_{i=1..n}$ as a set of peers, where the index i models the identifier of the peer $\mathcal{P}_i$ (e.g., its IP address).

A peer $\mathcal{P}_i$ manages some knowledge modeled by a *clausal theory* of PL, $\mathcal{T}(\mathcal{P}_i)$, and a *set of mappings* with some other peers, $\mathcal{M}(\mathcal{P}_i)$. The theory of $\mathcal{P}_i$ is a set of reduced clauses (they are either the empty clause $\Box$ equivalent to false, true, or clauses without redundant literals) in terms of the proper *alphabet* of $\mathcal{P}_i$. Such an alphabet is made of propositional *variables*. The peer alphabets being disjoint, I use here the peer identifier i to note A_i the variable A of the peer $\mathcal{P}_i$. The mappings in which $\mathcal{P}_i$ is involved are stored locally in $\mathcal{M}(\mathcal{P}_i)$, i.e., a mapping is stored in every peer involved in that mapping. A mapping is a reduced clause, the literals of which use variables from $\mathcal{P}_i$ and also from some other peers. That is, a mapping with n literals may involve up to n peers.

The (distributed) *theory* modeled by a P2PIS $\mathcal{S}$ is the union of the theories and mappings of its peers. Its semantics that of a standard (centralized) theory of PL.

[1] A clause is a disjunction of literals, i.e., variables or their negation. A clausal theory is a set of clauses modeling their conjunction, i.e., a theory is a PL formula in the so-called conjunctive normal form.

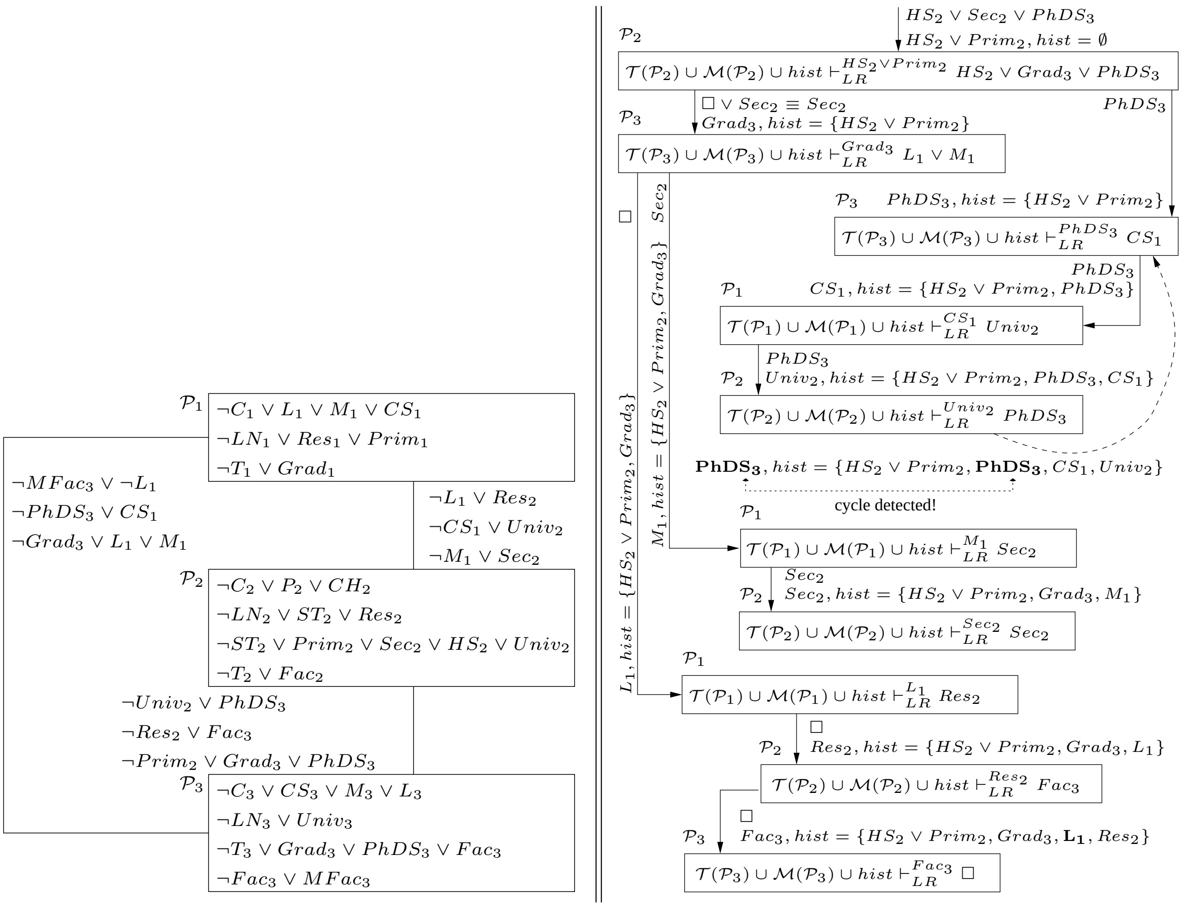

$\mathcal{P}_1$
$\neg C_1 \vee L_1 \vee M_1 \vee CS_1$
$\neg LN_1 \vee Res_1 \vee Prim_1$
$\neg T_1 \vee Grad_1$
$\neg MFac_3 \vee \neg L_1$
$\neg PhDS_3 \vee CS_1$
$\neg Grad_3 \vee L_1 \vee M_1$
$\neg L_1 \vee Res_2$
$\neg CS_1 \vee Univ_2$
$\neg M_1 \vee Sec_2$
$\mathcal{P}_2$
$\neg C_2 \vee P_2 \vee CH_2$
$\neg LN_2 \vee ST_2 \vee Res_2$
$\neg ST_2 \vee Prim_2 \vee Sec_2 \vee HS_2 \vee Univ_2$
$\neg T_2 \vee Fac_2$
$\neg Univ_2 \vee PhDS_3$
$\neg Res_2 \vee Fac_3$
$\neg Prim_2 \vee Grad_3 \vee PhDS_3$
$\mathcal{P}_3$
$\neg C_3 \vee CS_3 \vee M_3 \vee L_3$
$\neg LN_3 \vee Univ_3$
$\neg T_3 \vee Grad_3 \vee PhDS_3 \vee Fac_3$
$\neg Fac_3 \vee MFac_3$
$HS_2 \vee Sec_2 \vee PhDS_3$
$HS_2 \vee Prim_2, hist = \emptyset$
$\mathcal{P}_2$
$\mathcal{T}(\mathcal{P}_2) \cup \mathcal{M}(\mathcal{P}_2) \cup hist \vdash_{LR}^{HS_2 \vee Prim_2} HS_2 \vee Grad_3 \vee PhDS_3$
$\Box \vee Sec_2 \equiv Sec_2$
$Grad_3, hist = \{HS_2 \vee Prim_2\}$
$PhDS_3$
$\mathcal{P}_3$
$\mathcal{T}(\mathcal{P}_3) \cup \mathcal{M}(\mathcal{P}_3) \cup hist \vdash_{LR}^{Grad_3} L_1 \vee M_1$
$\Box$
Sec_2
$\mathcal{P}_3$ $PhDS_3, hist = \{HS_2 \vee Prim_2\}$
$\mathcal{T}(\mathcal{P}_3) \cup \mathcal{M}(\mathcal{P}_3) \cup hist \vdash_{LR}^{PhDS_3} CS_1$
$PhDS_3$
$\mathcal{P}_1$ $CS_1, hist = \{HS_2 \vee Prim_2, PhDS_3\}$
$\mathcal{T}(\mathcal{P}_1) \cup \mathcal{M}(\mathcal{P}_1) \cup hist \vdash_{LR}^{CS_1} Univ_2$
$PhDS_3$
$\mathcal{P}_2$ $Univ_2, hist = \{HS_2 \vee Prim_2, PhDS_3, CS_1\}$
$\mathcal{T}(\mathcal{P}_2) \cup \mathcal{M}(\mathcal{P}_2) \cup hist \vdash_{LR}^{Univ_2} PhDS_3$
$\mathbf{PhDS_3}, hist = \{HS_2 \vee Prim_2, \mathbf{PhDS_3}, CS_1, Univ_2\}$
cycle detected!
$L_1, hist = \{HS_2 \vee Prim_2, Grad_3\}$
$M_1, hist = \{HS_2 \vee Prim_2, Grad_3\}$
$\mathcal{P}_1$
$\mathcal{T}(\mathcal{P}_1) \cup \mathcal{M}(\mathcal{P}_1) \cup hist \vdash_{LR}^{M_1} Sec_2$
Sec_2
$\mathcal{P}_2$ $Sec_2, hist = \{HS_2 \vee Prim_2, Grad_3, M_1\}$
$\mathcal{T}(\mathcal{P}_2) \cup \mathcal{M}(\mathcal{P}_2) \cup hist \vdash_{LR}^{Sec_2} Sec_2$
$\mathcal{P}_1$
$\mathcal{T}(\mathcal{P}_1) \cup \mathcal{M}(\mathcal{P}_1) \cup hist \vdash_{LR}^{L_1} Res_2$
$\Box$
$\mathcal{P}_2$ $Res_2, hist = \{HS_2 \vee Prim_2, Grad_3, L_1\}$
$\mathcal{T}(\mathcal{P}_2) \cup \mathcal{M}(\mathcal{P}_2) \cup hist \vdash_{LR}^{Res_2} Fac_3$
$\Box$
$\mathcal{P}_3$ $Fac_3, hist = \{HS_2 \vee Prim_2, Grad_3, \mathbf{L_1}, Res_2\}$
$\mathcal{T}(\mathcal{P}_3) \cup \mathcal{M}(\mathcal{P}_3) \cup hist \vdash_{LR}^{Fac_3} \Box$

Figure 5.1: Sample P2PIS (left) and decentralized derivation for the input clause $HS_2 \vee Prim_2$ (right)

Example 20 (Propositional P2PIS) *Figure 5.1 (left) depicts a P2PIS made of the peers $\mathcal{P}_1$, $\mathcal{P}_2$, and $\mathcal{P}_3$. The theories of the peers are the nodes labeled with their names and the mappings between these peers are labeling the edges that link their theories. Each peer advertises a private course center by describing its courses, learners, teachers, and cooperations with the other centers.*

- *$\mathcal{P}_1$ states that the first center has (i) courses (C_1) that are language courses (L_1), math courses (M_1), or computer science courses (CS_1), (ii) learners (LN_1) that are researchers (Res_1) or students in primary school $(Prim_1)$, and (iii) teachers (T_1) that are graduates $(Grad_1)$.*
- *$\mathcal{P}_2$ states that the second center has (i) courses (C_2) that are physics courses (P_2) or chemistry courses (CH_2), (ii) learners (LN_2) that are researchers (Res_2) or students (ST_2) in primary school $(Prim_2)$, in secondary school (Sec_2), in high school (HS_2), or even in university $(Univ_2)$, and (iii) teachers (T_2) that are faculties (Fac_2).*
- *Finally, $\mathcal{P}_3$ states that the third center has (i) courses (C_3) that are language courses (L_3), math courses (M_3), or computer science courses (CS_3), (ii) learners (LN_3) that are students in university $(Univ_3)$, and (iii) teachers (T_3) that are graduates $(Grad_3)$, PhD students $(PhDS_3)$, or faculties (Fac_3) who are math faculties $(MFac_3)$.*

The three centers cooperate according to the following policies that are modeled by mappings.

- *The first center provides courses to students from the second center: it provides language courses (L_1) to researchers (Res_2), computer science courses (CS_1) to university students $(Univ_2)$, and math courses (M_1) to student in secondary school (Sec_2).*
- *The second center solicits teachers from the third center: its university students $(Univ_2)$ are taught by PhD students $(PhDS_3)$, its researchers (Res_2) are taught by faculties (Fac_3), and its students in primary school $(Prim_2)$ are taught by graduates $(Grad_3)$ or PhD students $(PhDS_3)$.*

- *Finally, teachers from the third center do, or do not, teach some courses for the first center: math faculties* $(MFac_3)$ *do not teach language courses* (L_1)*, PhD students* $(PhDS_3)$ *teach computer science courses* (CS_1)*, and graduates* $(Grad_3)$ *teach language courses* (L_1) *or math courses* (M_1)*.*

5.1.1.2 Consequence finding

Our consequence finding problem is the following: Given an input clause in terms of the variables of a single peer, we want to computes clausal consequences (a.k.a. *implicates*) of the P2PIS plus this input clause, including all the strongest ones following necessarily from this input clause (a.k.a. the *proper prime implicates* of the input clause).

The idea of our decentralized technique implemented in DECA is to decentralize *deductions* (a.k.a. derivations) based on (full) resolution or linear resolution, which provide a solution to our consequence finding problem in the centralized setting [MR72, Ino91, Ino92, Mar00].

More specifically, when a peer is solicited with an input clause, this peer derives locally, using standard centralized deductions, all the implicates following from this clause and its own knowledge (i.e., its theory and mappings). Then, for each such implicate having some remote literals, the peer builds other implicates by exhaustively replacing the remote literals by the implicates following from them. Of course, these ones are obtained by soliciting remote peers, which may in turn solicit others, and so on.

We have shown that returning all the locally derived implicates together with those built from them by soliciting remote peers provides a correct solution to our decentralized consequence finding problem. Notably, the termination and correctness of the technique rely on the use of an history memorizing the input clauses of the successive peer sollicitations that now lead to soliciting another peer. These clauses hold in the P2PIS when the peer is solicited, though they are not in the P2PIS knowledge: this extra knowledge is that of the current derivation. The solicited peer then uses this history to detect possible cyclic deductions (when the input clause is already in the history) and to reason locally in order to complement its own knowledge with that of the current derivation.

Example 21 (DECA in action) *Figure 5.1 (right) shows a particular* DECA *derivation in the P2PIS of Figure 5.1 (left). In this derivation, the history is denoted* $hist$*.*

The derivation is initiated by soliciting $\mathcal{P}_2$ *with the input clause* $HS_2 \vee Prim_2$ *and an empty history (since it is the initial solicitation). This is represented on top of Figure 5.1 (right), by the edge labeled with this input clause and empty history that reaches the top node labeled with* $\mathcal{P}_2$*. The other label of this edge is a particular answer to this solicitation, the implicate* $HS_2 \vee Sec_2 \vee PhDS_3$*, which is computed as follows.*

Upon the above solicitation, $\mathcal{P}_2$ *derives locally the implicate* $HS_2 \vee Grad_3 \vee PhDS_3$ *from its theory and mappings, the input clause, and the history. This is denoted by* $\mathcal{T}(\mathcal{P}_2) \cup \mathcal{M}(\mathcal{P}_2) \cup hist \vdash_{LR}^{HS_2 \vee Prim_2} HS_2 \vee Grad_3 \vee PhDS_3$*.*

Since two remote literals from $\mathcal{P}_3$ *are involved in this implicate,* $\mathcal{P}_3$ *receives two solicitations for obtaining the implicates following from these literals. This is depicted by the two edges going from the top node labeled with* $\mathcal{P}_2$ *to two distinct nodes labeled with* $\mathcal{P}_3$*. Again, each edge is labeled with (i) the input of the solicitation (the input clause and the current history) and (ii) an answer to the solicitation, i.e., one of the requested implicates.*

In this particular case, $\mathcal{P}_2$ *receives from* $\mathcal{P}_3$*,* Sec_2 *as an implicate following from the remote literal* $Grad_3$ *and* PhD_3 *as an implicate of itself. Observe that the computation of these two implicates have in turn solicited other peers. From these two implicates and* $HS_2 \vee Grad_3 \vee PhDS_3$*,* $\mathcal{P}_2$ *builds and returns the implicate* $HS_2 \vee Sec_2 \vee PhDS_3$*.*

Finally, it is worth noticing that the whole derivation illustrates the role of the history w.r.t. completeness and termination of the technique: the empty clause cannot be derived without the use of the history (actually L_1*) in the node at the bottom of the figure; the history is used to detect/avoid a cyclic derivation (see the dashed arrow).*

5.1.1.3 (Non-)Conservative extension checking

Our (non-)conservative extension checking problem amounts to exhibits the *witnesses to non-conservative extension of a given peer*. These witnesses are characterized by the prime implicates of the P2PIS in terms of the peer's alphabet only, that necessarily follows from a *mapping* of this peer, without being consequences of the peer alone.

The idea of the technique implemented in CECA is to rely on a variant of DECA, which compute implicates of the P2PIS in terms of the peer's alphabet only, including all the prime ones that follow from a mapping of this peer. Witnesses to non-conservative extension are then exhibited among these implicates, by removing the implicates that are not consequences of the theory of the peer. This is done by a usual centralized refutation mechanism.

Example 22 (CECA **in action**) *Consider again the P2PIS in Figure 5.1 (left).*

As for $\mathcal{P}_1$, CECA *gets $\neg L_1$ from* DeCA*, as an implicate following from the mapping $\neg L_1 \vee RES_2$. $\neg L_1$ is actually derived from $\{\neg L_1 \vee Res_2, \neg Res_2 \vee Fac_3, \neg Fac_3 \vee MFac_3, \neg MFac_3 \vee \neg L_1\}$. Moreover, $\neg L_1$ is not a consequence of $\mathcal{T}(\mathcal{P}_1)$, thus $\neg L_1$ is a witness to non-conservative extension of $\mathcal{P}_1$. As a result, the P2PIS says that there is no language courses provided by $\mathcal{P}_1$. Note that this extra knowledge is incomparable (w.r.t. the consequence relation $\models$) with that of $\mathcal{T}(\mathcal{P}_1)$.*

As for $\mathcal{P}_1$, CECA *gets $\neg ST_2 \vee Sec_2 \vee HS_2 \vee Univ_2$ from* DeCA*, as an implicate following from the mapping $\neg Prim_2 \vee Grad_3 \vee PhDS_3$. $\neg ST_2 \vee Sec_2 \vee HS_2 \vee Univ_2$ is actually derived from $\{\neg ST_2 \vee Prim_2 \vee Sec_2 \vee HS_2 \vee Univ_2, \neg Prim_2 \vee Grad_3 \vee PhDS_3, \neg Grad_3 \vee L_1 \vee M_1, \neg L_1 \vee Res_2, \neg Res_2 \vee Fac_3, \neg Fac_3 \vee MFac_3, \neg MFac_3 \vee \neg L_1, \neg M_1 \vee Sec_2, \neg PhDS_3 \vee CS_1, \neg CS_1 \vee Univ_2\}$. Moreover, $\neg ST_2 \vee Sec_2 \vee HS_2 \vee Univ_2$ is not a consequence of $\mathcal{T}(\mathcal{P}_2)$, thus $\neg ST_2 \vee Sec_2 \vee HS_2 \vee Univ_2$ is a witness to non-conservative extension of $\mathcal{P}_2$. As a result, the P2PIS says that the students of $\mathcal{P}_2$ are in secondary schools, high schools, or universities. Note that this extra knowledge is refines (w.r.t. the consequence relation $\models$) that of $\mathcal{T}(\mathcal{P}_2)$.*

The P2PIS is a conservative extension of $\mathcal{P}_3$, as there is no witness to non-conservative extension of $\mathcal{P}_3$.

5.1.2 P2P data management systems for RDF and DL-lite

Context I have worked on P2P systems from 2004 to 2010 with Serge Abiteboul (DR Inria Saclay–Île-de-France), Philippe Chatalic (MCF, Univ. Paris-Sud), Ioana Manolescu (DR Inria Saclay–Île-de-France), Marie-Christine Rousset (PR, Univ. Grenoble), Laurent Simon (MCF, Univ. Paris-Sud), Philippe Adjiman (former PhD student, Univ. Paris-Sud, co-advised by Marie-Christine and me), and Nada Abdallah (former PhD student, Inria, co-advised by Serge and me).

Preliminary milestone results on decentralized data management were published in the proceedings of the workshop Information Integration on the Web (IIWeb) [GR03a] in 2003 and the workshop on Principles and Practice of Semantic Web Reasoning (PPSWR) [ACG$^+$05b]. Completed milestone results were published in IEEE Intelligent Systems (IEEE IS) [GR03b] in 2003, Journal on Artificial Intelligence Research (JAIR) [ACG$^+$06] in 2006, Journal On Data Semantics (JODS) [AGR07] in 2007, in the journal Proceedings of the VLDB endowment (PVLDB) [AAC$^+$08] in 2008, and in the proceedings of International Joint Conference on Artificial Intelligence (IJCAI) [AGR09] in 2009 and of the Congrès francophone Reconnaissance des Formes et Intelligence Artificielle (RFIA) [AGR10].

Motivations The emergence of P2P file sharing systems over the Internet, in the beginning of the 21^{th} century, has also attracted the attention of the DB community. The first peer-to-peer data management systems (PDMSs) were proposed for the relational data model [HIST03, TIM$^+$03]. In such systems, each peer manages its own database and can also establish mappings with some peers whose application domains overlap. Mappings are inclusions between a conjunctive query against one peer and a conjunctive query against another peer. In this rather simple setting however, answering a query from the whole system is undecidable, except under severe topological constraints w.r.t. real applications (acyclic mappings) [HIST03, TIM$^+$03], or by using non-standard semantics (epistemic semantics) [FKLZ04, CGLR04].

Contributions In [GR03a, GR03b, ACG$^+$05b, ACG$^+$06, AGR07, AAC$^+$08, AGR09, AGR10], we design PDMSs for the Semantic Web data models. Based on the rather negative results obtained in the setting of relational PDMSs (see above), we focus on simple ontology languages. We choose them so as to compile PDMSs into our propositional P2PISs (cf. Section 5.1.1) and then reduce (part of) data management tasks to consequence finding. In particular, our SomeRDFS PDMS [AGR07] has been used in the WebContent project[2] [AAC$^+$08], a 2006-2009 ANR "Réseau National de Technologies Logicielles" project in which I have participated, whose goal was to integrate off-the-shelf technologies from the partners in order to manage XML documents with RDF annotations in a decentralized fashion.

More specifically, our contributions are:

1. We define the SomeOWL, SomeRDFS, and SomeDL-lite PDMSs, whose data models are the $\mathcal{CLU}$ description logic[3], RDF, and DL-lite respectively. Each peer manages its own ontology and data, and can establish mappings with peers having overlapping interests. In particular, mappings are ontological constraints involving relations (i.e., concepts and roles, or classes and properties) from several peers.

[2] `http://www.webcontent.fr`

[3] The $\mathcal{CLU}$ description logic is a fragment of the OWL-DL dialect of OWL1.

$\mathcal{P}_1$'s graph	$\mathcal{P}_2$'s graph
$\mathcal{P}_1$:Sculptor rdfs:subClassOf $\mathcal{P}_1$:Artist .	$\mathcal{P}_2$:Painting rdfs:subClassOf $\mathcal{P}_2$:Work .
$\mathcal{P}_1$:Painter rdfs:subClassOf $\mathcal{P}_1$:Artist .	$\mathcal{P}_2$:Sculpture rdfs:subClassOf $\mathcal{P}_2$:Work .
$\mathcal{P}_1$:creates rdfs:domain $\mathcal{P}_1$:Artist .	$\mathcal{P}_2$:Music rdfs:subClassOf $\mathcal{P}_2$:Work .
$\mathcal{P}_1$:creates rdfs:range $\mathcal{P}_1$:Artifact .	$\mathcal{P}_2$:refersTo rdfs:domain $\mathcal{P}_2$:Work .
$\mathcal{P}_1$:paints rdfs:subPropertyOf $\mathcal{P}_1$:creates .	$\mathcal{P}_2$:refersTo rdfs:range $\mathcal{P}_2$:Period .
$\mathcal{P}_1$:sculpts rdfs:subPropertyOf $\mathcal{P}_1$:creates .	
$\mathcal{P}_1$:sculpts rdfs:domain $\mathcal{P}_1$:Sculptor .	
$\mathcal{P}_1$:paints rdfs:domain $\mathcal{P}_1$:Painter .	
$\mathcal{P}_1$:belongsTo rdfs:domain $\mathcal{P}_1$:Artifact .	
$\mathcal{P}_1$:belongsTo rdfs:range $\mathcal{P}_1$:Movement .	
Picasso $\mathcal{P}_1$:paints Les−demoiselles−d−Avignon .	Le−dejeuner−des−canotiers rdf:type $\mathcal{P}_2$:Painting .
Picasso $\mathcal{P}_1$:sculpts La−femme−au−chapeau .	Les−demoiselles−d−Avignon $\mathcal{P}_2$:refersTo Cubism .
Les−demoiselles−d−Avignon $\mathcal{P}_1$:belongsTo Picasso−pink .	The−statue−of−David rdf:type $\mathcal{P}_2$:Sculpture .
La−femme−au−chapeau $\mathcal{P}_1$:belongsTo Modern−art .	Nutcracker rdf:type $\mathcal{P}_2$:Music .
$\mathcal{P}_1$:paints rdfs:range $\mathcal{P}_2$:Painting .	$\mathcal{P}_1$:paints rdfs:range $\mathcal{P}_2$:Painting .
$\mathcal{P}_1$:sculpts rdfs:range $\mathcal{P}_2$:Sculpture .	$\mathcal{P}_1$:sculpts rdfs:range $\mathcal{P}_2$:Sculpture .
$\mathcal{P}_1$:Artifact rdfs:subClassOf $\mathcal{P}_2$:Work .	$\mathcal{P}_1$:Artifact rdfs:subClassOf $\mathcal{P}_2$:Work .
$\mathcal{P}_1$:belongsTo rdfs:subPropertyOf $\mathcal{P}_2$:refersTo .	$\mathcal{P}_1$:belongsTo rdfs:subPropertyOf $\mathcal{P}_2$:refersTo .

Figure 5.2: Sample SomeRDFS PDMS. Within a peer's graph, RDFS statements are given first, then RDF statements, and finally the mappings.

2. We define how these PDMSs can be built on top of our propositional P2PISs, by encoding the ontology and mappings of each peer into the corresponding propositional theory and mappings. A decentralized data management technique, e.g., for consistency checking or query answering, then amounts to write a centralized data management technique with the appropriate calls to DeCA when remote information is needed. In particular, we provide query answering techniques for our SomeOWL, SomeRDFS, and SomeDL-lite PDMSs, and a consistency checking technique for our SomeDL-lite PDMS. In addition, for the SomeDL-lite PDMS, we also provide query answering and consistency checking techniques based on query rewriting, when peers access data through conjunctive views.

In the following, I first introduce the SomeRDFS PDMSs (Section 5.1.2.1), and then I give some intuitions and exemplify how query answering is performed in these PDMSs (Section 5.1.2.2).

5.1.2.1 SomeRDFS

We define a SomeRDFS PDMS $\mathcal{S} = \{\mathcal{P}_i\}_{i=1..n}$ as a set of peers, where the index i models the identifier of the peer $\mathcal{P}_i$ (e.g., its IP address or namespace).

A peers $\mathcal{P}_i$ manages some knowledge modeled by graph $\mathtt{G}(\mathcal{P}_i)$ belonging to the DL fragment of RDF (cf. Section 2.3). In particular, this graph uses the own classes and properties of $\mathcal{P}_i$. I denote the relation r of $\mathcal{P}_i$ (i.e., a class or property) by $\mathcal{P}_i{:}r$. The mappings in which a peer in which $\mathcal{P}_i$ is involved are stored locally in $\mathcal{M}(\mathcal{P}_i)$, i.e., a mapping is stored in every peer involved in that mapping. A mapping is an RDFS statements involving the vocabulary of two peers.

The (distributed) *graph* modeled a SomeRDFS $\mathcal{S}$ is the union of the graphs and mappings of its peers. Its semantics is that of a graph belonging to the DL fragment of RDF.

Example 23 (A SomeRDFS PDMS about Art) *Consider the* SomeRDFS *PDMS in Figure 5.2, which is made of two peers $\mathcal{P}_1$ and $\mathcal{P}_2$.*

The ontological knowledge of $\mathcal{P}_1$ is about artists (some of them being sculptors and/or painters), artifacts artists have created, and the artistic movements the artifacts belong to. Some artist creations are distinguished according to whether their creators are sculptors or painters. $\mathcal{P}_1$ actually stores that Picasso has painted "Les demoiselles d'Avignon" which belongs to the Picasso's pink movement, and has sculpted "La femme au chapeau" which belongs to the Modern art movement.

The ontological knowledge of $\mathcal{P}_2$ is about works (some of them being paintings, sculptures or musics) and the artistic period they refer to. $\mathcal{P}_2$ stores that "Le déjeuner des canotiers" is a painting and that "Les demoiselles d'Avignon" refers to the Cubism artistic period. It also stores that "The statue of David" is a sculpture and that "Nutcracker" is a music.

In order to collaborate, $\mathcal{P}_1$ and $\mathcal{P}_2$ have established several mappings: the artifacts that are painted according to $\mathcal{P}_1$ are painting according to $\mathcal{P}_2$; the artifacts that are sculpted according to $\mathcal{P}_1$ are sculpture according to $\mathcal{P}_2$; an artifact for $\mathcal{P}_1$ is a work for $\mathcal{P}_2$; and an artifact that belongs to a movement according to $\mathcal{P}_1$ is a work that refers to a period according to $\mathcal{P}_2$.

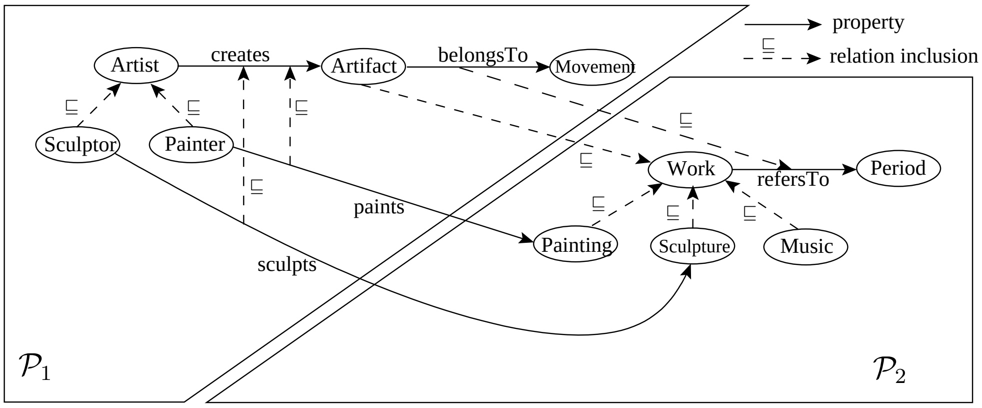


Figure 5.3: Graphical representation of the SomeRDFS PDMS of Figure 5.2

Figure 5.3 provides a graphical representation of the decentralized ontology of this PDMS, consisting of the ontology of the peers (their RDFS statements) and of their mappings.

5.1.2.2 Query answering in SomeRDFS

The query language of SomeRDFS is the fragment of SPARQL corresponding to the relational conjunctive queries (cf. Section 2.3).

The query answering technique used in SomeRDFS is based on query reformulation. Given a query against the graph of a peer, this query is first reformulated according to the whole PDMS, then its reformulation is evaluated in a distributed fashion against the relevant peers. The reformulation step amounts to reformulating the query's triples in parallel, i.e., their reformulations are independent, so that the reformulation of the whole query is the union of all possible queries obtained by replacing the original query's triples by one of their reformulations. The evaluation step then consists in evaluating this union, by contacting the appropriate peers. Observe that in the considered query language (relational conjunctive queries), triples are of the form s rdf:type $\mathcal{P}_i{:}c$. or of the form s $\mathcal{P}_i{:}p$ o ., that is we always know the relevants remote peers to contact at query evaluation time.

The reformulation of the original query's triples is the crux of the technique as they must be reformulated according to the graph modeled by the whole PDMS. This step is achieved by building SomeRDFS peers on top of SomeWhere peers. In fact, the underlying SomeWhere peer manages a propositional encoding of the RDFS statements (ontological ones and mapping ones). The propositional encoding function enc used in SomeRDFS is the following: enc(s rdfs:subClassOf o .) $= \{\neg \mathtt{s}^d \vee \mathtt{o}^d, \neg \mathtt{s}^r \vee \mathtt{o}^r\}$, enc(s rdfs:subPropertyOf o .) $=$ $\{\neg \mathtt{s}^p \vee \mathtt{o}^p\}$, enc(s rdfs:domain o .) $= \{\neg \mathtt{s}^p \vee \mathtt{o}^d\}$, and enc(s rdfs:range o .) $= \{\neg \mathtt{s}^p \vee \mathtt{o}^r\}$. The propositional variables are annotated with p, d, and r, which respectively encode that a variable represents a property, a class that may type the domain of a property, and a class that may type the range of a property. With this encoding, the correctness of our query answering technique has been shown when the reformulations of the original query's triples are computed as follows. The reformulation of triple s rdf:type $\mathcal{P}_i{:}c$. is encoded in two parallel DeCA calls: $DeCA(\neg\mathcal{P}_i{:}c^d)$ and $DeCA(\neg\mathcal{P}_i{:}c^r)$. The results to the first call are decoded as follows: if $\neg\mathcal{P}_j{:}c'^d \in DeCA(\neg\mathcal{P}_i{:}c^d)$ then the corresponding reformulation is s rdf:type $\mathcal{P}_j{:}c'$.; if $\neg\mathcal{P}_j{:}p'^p \in DeCA(\neg\mathcal{P}_i{:}c^d)$ then the corresponding reformulation is s $\mathcal{P}_j{:}p'^p$ y . with y a fresh non-distinguished variable. Observe in the latter case that s becomes a subject of the reformulated triple because the call to DeCA looks for classes that may type the domain of a property. Similarly, the results to the second call are decoded as follows: if $\neg\mathcal{P}_j{:}c'^r \in DeCA(\neg\mathcal{P}_i{:}c^r)$ then the corresponding reformulation is s rdf:type $\mathcal{P}_j{:}c'$.; if $\neg\mathcal{P}_j{:}p'^p \in DeCA(\neg\mathcal{P}_i{:}c^r)$ then the corresponding reformulation is y $\mathcal{P}_j{:}p'^p$ s . with y a fresh non-distinguished variable. Otherwise, the reformulation of a triple s $\mathcal{P}_i{:}p$ o . is encoded in a single DeCA call: $DeCA(\neg\mathcal{P}_i{:}p^p)$. The results to this call is decoded as follows: if $\neg\mathcal{P}_j{:}p'^p \in DeCA(\neg\mathcal{P}_i{:}p^p)$ then the corresponding reformulation is s $\mathcal{P}_j{:}p'$ o .

$\mathcal{P}_1$'s theory and mappings	$\mathcal{P}_2$'s theory and mappings
$\neg\mathcal{P}_1$:Sculptor^d $\vee$ $\mathcal{P}_1$:Artist^d	$\neg\mathcal{P}_2$:Painting^d $\vee$ $\mathcal{P}_2$:Work^d
$\neg\mathcal{P}_1$:Sculptor^r $\vee$ $\mathcal{P}_1$:Artist^r	$\neg\mathcal{P}_2$:Painting^r $\vee$ $\mathcal{P}_2$:Work^r
$\neg\mathcal{P}_1$:Painter^d $\vee$ $\mathcal{P}_1$:Artist^d	$\neg\mathcal{P}_2$:Sculpture^d $\vee$ $\mathcal{P}_2$:Work^d
$\neg\mathcal{P}_1$:Painter^r $\vee$ $\mathcal{P}_1$:Artist^r	$\neg\mathcal{P}_2$:Sculpture^r $\vee$ $\mathcal{P}_2$:Work^r
$\neg\mathcal{P}_1$:creates^p $\vee$ $\mathcal{P}_1$:Artist^d	$\neg\mathcal{P}_2$:Music^d $\vee$ $\mathcal{P}_2$:Work^d
$\neg\mathcal{P}_1$:creates^p $\vee$ $\mathcal{P}_1$:Artifact^r	$\neg\mathcal{P}_2$:Music^r $\vee$ $\mathcal{P}_2$:Work^r
$\neg\mathcal{P}_1$:paints^p $\vee$ $\mathcal{P}_1$:creates^p	$\neg\mathcal{P}_2$:refersTo^p $\vee$ $\mathcal{P}_2$:Work^d
$\neg\mathcal{P}_1$:sculpts^p $\vee$ $\mathcal{P}_1$:creates^p	$\neg\mathcal{P}_2$:refersTo^p $\vee$ $\mathcal{P}_2$:Period^r
$\neg\mathcal{P}_1$:sculpts^p $\vee$ $\mathcal{P}_1$:Sculptor^d	
$\neg\mathcal{P}_1$:paints^p $\vee$ $\mathcal{P}_1$:Painter^d	
$\neg\mathcal{P}_1$:belongsTo^p $\vee$ $\mathcal{P}_1$:Artifact^d	
$\neg\mathcal{P}_1$:belongsTo^p $\vee$ $\mathcal{P}_1$:Movement^r	
$\neg\mathcal{P}_1$:paints^p $\vee$ $\mathcal{P}_2$:Painting^r	$\neg\mathcal{P}_1$:paints^p $\vee$ $\mathcal{P}_2$:Painting^r
$\neg\mathcal{P}_1$:sculpts^p $\vee$ $\mathcal{P}_2$:Sculpture^r	$\neg\mathcal{P}_1$:sculpts^p $\vee$ $\mathcal{P}_2$:Sculpture^r
$\neg\mathcal{P}_1$:Artifact^d $\vee$ $\mathcal{P}_2$:Work^d	$\neg\mathcal{P}_1$:Artifact^d $\vee$ $\mathcal{P}_2$:Work^d
$\neg\mathcal{P}_1$:Artifact^r $\vee$ $\mathcal{P}_2$:Work^r	$\neg\mathcal{P}_1$:Artifact^r $\vee$ $\mathcal{P}_2$:Work^r
$\neg\mathcal{P}_1$:belongsTo^p $\vee$ $\mathcal{P}_2$:refersTo^p	$\neg\mathcal{P}_1$:belongsTo^p $\vee$ $\mathcal{P}_2$:refersTo^p

Figure 5.4: Propositional encoding of the SOMERDFS peers from Figure 5.2.

Example 24 (Decentralized query answering) *Figure 5.4 shows the propositional encoding of the* SOMERDFS *peers from Figure 5.2.*

Consider the query $q(x)$:- x rdf:type $\mathcal{P}_2$:Work . *asking for all the works to* $\mathcal{P}_2$*. Answering this query triggers two calls to* DECA

- $DeCA(\neg\mathcal{P}_2\text{:Work}^d) = \{\neg\mathcal{P}_2\text{:Work}^d, \neg\mathcal{P}_2\text{:Painting}^d, \neg\mathcal{P}_2\text{:Sculpture}^d, \neg\mathcal{P}_2\text{:Music}^d, \neg\mathcal{P}_2\text{:refersTo}^p, \neg\mathcal{P}_1\text{:Artifact}^d, \neg\mathcal{P}_1\text{:belongsTo}^p\}$
- $DeCA(\neg\mathcal{P}_2\text{:Work}^r) = \{\neg\mathcal{P}_2\text{:Work}^r, \neg\mathcal{P}_2\text{:Painting}^r, \neg\mathcal{P}_2\text{:Sculpture}^r, \neg\mathcal{P}_2\text{:Music}^r, \neg\mathcal{P}_1\text{:Artifact}^r, \neg\mathcal{P}_1\text{:creates}^p, \neg\mathcal{P}_1\text{:paints}^p, \neg\mathcal{P}_1\text{:sculpts}^p\}$

from which we obtain the reformulation
$Q(x) = q^1(x)$:- x rdf:type $\mathcal{P}_2$:Work . $\cup\, q^2(x)$:- x rdf:type $\mathcal{P}_2$:Painting . $\cup\, q^3(x)$:- x rdf:type $\mathcal{P}_2$:Sculpture . $\cup$ $q^4(x)$:- x rdf:type $\mathcal{P}_2$:Music . $\cup$ $q^5(x)$:- x $\mathcal{P}_2$:refersTo y . $\cup$ $q^6(x)$:- x rdf:type $\mathcal{P}_1$:Artifact . $\cup$ $q^7(x)$:- x $\mathcal{P}_1$:belongsTo y . $\cup\, q^8(x)$:- y $\mathcal{P}_1$:creates x . $\cup\, q^9(x)$:- y $\mathcal{P}_1$:paints x . $\cup\, q^{10}(x)$:- y $\mathcal{P}_1$:sculpts x .

The distributed evaluation of this reformulation is:

$$Q(\mathcal{S}) = \underbrace{\emptyset}_{q^1(\mathcal{S})} \cup \underbrace{\{\textit{Le-dejeuner-des-canotiers}\}}_{q^2(\mathcal{S})} \cup \underbrace{\{\textit{The-statue-of-David}\}}_{q^3(\mathcal{S})} \cup \underbrace{\{\textit{Nutcracker}\}}_{q^4(\mathcal{S})}$$
$$\cup \underbrace{\{\textit{Les-demoiselles-d-Avignon}\}}_{q^5(\mathcal{S})} \cup \underbrace{\emptyset}_{q^6(\mathcal{S})} \cup \underbrace{\{\textit{Les-demoiselles-d-Avignon,La-femme-au-chapeau}\}}_{q^7(\mathcal{S})} \cup \underbrace{\emptyset}_{q^8(\mathcal{S})}$$
$$\cup \underbrace{\{\textit{Les-demoiselles-d-Avignon}\}}_{q^9(\mathcal{S})} \cup \underbrace{\{\textit{La-femme-au-chapeau}\}}_{q^{10}(\mathcal{S})}.$$

Consider now the query $q(x,y)$:- x rdf:type $\mathcal{P}_2$:Painting ., x $\mathcal{P}_2$:refersTo y . *asking* $\mathcal{P}_2$ *for paintings and the periods they refer to. Answering this query triggers three calls to* DECA

- $DeCA(\neg\mathcal{P}_2\text{:Painting}^d) = \{\neg\mathcal{P}_2\text{:Painting}^d\}$
- $DeCA(\neg\mathcal{P}_2\text{:Painting}^r) = \{\neg\mathcal{P}_2\text{:Painting}^r, \neg\mathcal{P}_1\text{:paints}^p\}$
- $DeCA(\neg\mathcal{P}_2\text{:refersTo}^p) = \{\neg\mathcal{P}_2\text{:refersTo}^p, \neg\mathcal{P}_1\text{:belongsTo}^p\}$

from which we obtain the reformulation
$Q(x,y) = q_1(x,y)$:- x rdf:type $\mathcal{P}_2$:Painting ., x $\mathcal{P}_2$:refersTo y . $\cup$ $q^2(x,y)$:- x rdf:type $\mathcal{P}_2$:Painting ., x $\mathcal{P}_1$:belongsTo y .$\cup q^3(x,y)$:- z $\mathcal{P}_1$:paints x ., x $\mathcal{P}_2$:refersTo y .$\cup q^4(x,y)$:- z $\mathcal{P}_1$:paints x ., x $\mathcal{P}_1$:belongsTo y .

The distributed evaluation of this reformulation is:

$$Q(\mathcal{S}) = \underbrace{\emptyset}_{q^1(\mathcal{S})} \cup \underbrace{\emptyset}_{q^2(\mathcal{S})} \cup \underbrace{\{(\textit{Les-demoiselles-d-Avignon}, \textit{Cubism})\}}_{q^3(\mathcal{S})} \cup \underbrace{\{(\textit{Les-demoiselles-d-Avignon}, \textit{Picasso-pink})\}}_{q^4(\mathcal{S})}.$$

5.2 Towards cloud-based RDF data management

Context Since 2011, I have been working on RDF cloud-based data management with Ioana Manolescu (DR Inria, Saclay–Île-de-France), Andrés Aranda Andújar (Engineer, Inria), Francesca Bugiotti (PhD student, Università Roma Tré), and Zoi Kaoudi (Postdoc, Inria).

Our preliminary results on devising an RDF data management architecture within the Amazon cloud were published in the proceedings of the workshop on Data analytics in the Cloud (DanaC) [BGKM12] in 2012.

Motivations The advent of cloud computing environments has rapidly gained the interest of the data management community, as they offer quasi-unbounded resource allocation, elastic scaling up and down of the resources according to the demand, and reliable execution (tasks started in the cloud will complete even after a possible failure). Still, clouds do not provide many facilities for developing and deploying applications: they only provide very low-level building blocks like virtual machines, disk space, and some services (simple database access or job queues, etc.).

Contributions In [BGKM12], we devise an RDF data management system within the Amazon Web Services cloud infrastructure (a.k.a. AWS). This preliminary study aims at understanding the building blocks for developing applications that a well-established cloud provides, and also at identifying the missing ones w.r.t. efficient data management systems. Our contributions are:

1. We provide the AMADA system, designed within AWS, for *evaluating* BGP queries against a collection of graphs (i.e., we do not consider entailment here).

2. We propose several indexes to quickly identify the relevant graphs when evaluating a given BGP query. We also provide preliminary experiments for query evaluation in AMADA w.r.t. efficiency and monetary costs.

In the following, I first introduce the AMADA's architecture (Section 5.2.1), then I give some intuitions on and exemplify how query evaluation has been optimized (Section 5.2.2).

5.2.1 The AMADA RDF data management system for the Amazon cloud

The design of AMADA was guided by the following objectives. First, we aimed to leverage AWS resources by scaling up to large data volumes. Second, we aimed at efficiency, in particular for the graph storage and querying operations. We quantify this efficiency by the response time provided by the cloud-hosted application. Our third objective is to minimize cloud resource usage, or, in classical distributed databases terms, the total work required for our operations. This is all the more important since, in a cloud, total work translates into monetary costs.

AMADA stores graphs and indexes in a distributed fashion within AWS. *Graphs* are stored within the distributed file system S3, which is the AWS store for (very) large data. S3 records the address of every stored graphs within its internal catalog, assigning to each graph an internal URI, based on which it can be retrieved. Furthermore, we build *graph indexes* within SimpleDB, a simple database system supporting SQL-style queries based on a key-value model. Observe that while SimpleDB generalizes relational databases by supporting heterogeneous tuple and multi-valued attributes, it is more restricted in that it only supports single-relation queries (that is, no joins). We have designed several indexing strategies differing in the choice of index keys, and in their level of detail, i.e., whether they point to specific graphs, or to very fine-grained data items within the graphs. The code which actually processes queries runs on virtual machines within the Amazon Elastic Compute Cloud (EC2). Finally, in order to synchronize the distributed components of our application, we use Amazon Simple Queue Service (SQS), an AWS queue service providing asynchronous message-based communication.

Figure 5.5 gives an in-depth view of AMADA's architecture. A graph submitted to the *front-end module* is stored as a file in S3, whose URI is sent to an indexing module running on an EC2 instance. This module retrieves the corresponding graph from S3 and builds an index that is stored in SimpleDB. A query submitted to the *front-end module* is sent to a query processor module, based on RDF-3X [NW08], running on an EC2 instance. This module performs a look-up to the indexes in SimpleDB so as to find out the relevant graphs for processing the query, and evaluates the query against them. Results are written in a file stored in S3, whose URI is sent to the *front-end module*, so that it retrieves the query results from S3 and returns them.

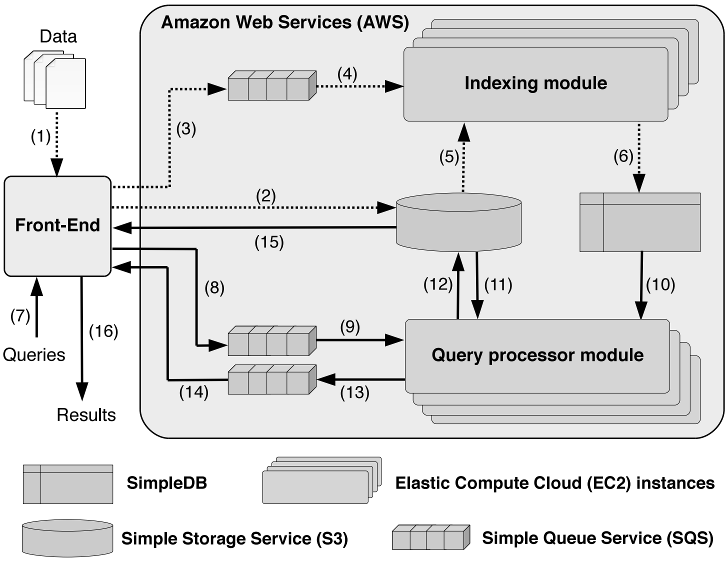


Figure 5.5: AMADA's architecture.

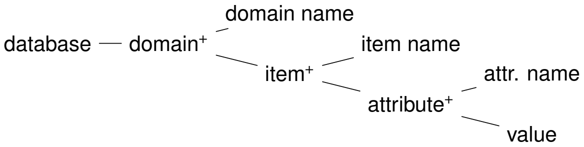


Figure 5.6: Structure of a SimpleDB database.

Scalability, parallelism and fault-tolerance AMADA exploits the elastic scaling of AWS by increasing and decreasing the number of EC2 instances running each module. The synchronization through the SQS message queues among modules supports inter-machine parallelism, whereas intra-machine parallelism is supported by multi-threading our code. AMADA also benefits from the fault-tolerance which AWS provides by periodically monitoring the queues. If an instance crashes while loading a graph or processing the query, AWS notices that the instance has not released the SQS message which had caused the work to start. In this case, another EC2 instance will take over the job.

5.2.2 Graph indexing in AMADA

An important feature of AMADA is graph indexing within SimpleDB. A description of SimpleDB's organization (outlined in Figure 5.6) is helpful to understand the available options for the index. SimpleDB data is organized in *domains*. Each domain is a collection of *items* identified by their *names*. In turn, each item has one or more *attributes*; an attribute has a *name*, and one or several *values*. An attribute value may be empty (denoted ϵ). Different items within a SimpleDB domain may have different attribute names.

The SimpleDB API provides a *get(D,k)* operation retrieving all items in the domain D having the name k, and a *put* to set values of attributes: *put(D,k,(a,v)+)* inserts the attributes *(a,v)+* into an item named k in domain D. Beyond the API, SimpleDB also provides a SQL-like higher-level language, e.g., one can use the query *select * from mydomain where Year > '1955'* to retrieve the items matching the condition. However, *a query cannot span multiple domains*, thus joins or unions have to be coded outside SimpleDB.

Conceptually, an indexing strategy $\mathcal{I}$ is a function extracting quadruplets of the form (domain name, item name, attribute name, attribute value) from an input graph $\mathtt{G}$. Indexing $\mathtt{G}$ according to $\mathcal{I}$, then, amounts to (i) computing the quadruplets in $\mathcal{I}(\mathtt{G})$ and (ii) adding these quadruplets to SimpleDB, using appropriate (batched, sometimes conditional) *put* operations. Let $\mathtt{G}$ be a graph whose URI is $U_{\mathtt{G}}$ and $s_1\ p_1\ o_1$. be a triple in $\mathtt{G}$. Let $\underline{s}$, $\underline{p}$ and $\underline{o}$ be three distinct tokens representing subjects, properties and respectively objects. The simplest indexing strategy called ATT (for attribute-based) uses a set of *subject domains* named $sd_1, sd_2, \ldots$; initially there is only

Graph for *articles*
inria:article1 inria:hasAuthor inria:bar .
inria:bar inria:hasName "Bar" .
inria:bar inria:hasNationality "American" .

Graph for *books*
inria:book1 inria:hasAuthor inria:foo .
inria:book1 inria:hasContactInfo $\_:uid1$.
inria:book2 inria:hasAuthor $\_:uid1$.
inria:foo inria:hasName "Foo" .
inria:foo inria:hasNationality "French" .
inria:foo inria:hasTel " + 3312345678" .
$\_:uid1$ inria:hasRole "Professor" .
$\_:uid1$ inria:hasTel " + 3312345679" .

Graph for *labs*
labInria:lab1 labInria:hasLocation labInria:location .
labInria:lab1 labInria:hasName "ResearchLab" .
labInria:location labInria:hasGPS "48.710715, 2.17545" .

subject domain	
item key	(attr. name, attr. value)
articles	($\underline{\text{S}}$, inria:article1), ($\underline{\text{S}}$, inria:bar)
books	($\underline{\text{S}}$, inria:book1), ($\underline{\text{S}}$, inria:foo), ($\underline{\text{S}}$, _:uid1), ($\underline{\text{S}}$, inria:book1)
labs	($\underline{\text{S}}$, labInria:lab1), ($\underline{\text{S}}$, labInria:location)

property domain	
item key	(attr. name, attr. value)
articles	($\underline{\text{P}}$, inria:hasAuthor), ($\underline{\text{P}}$, inria:hasName), ($\underline{\text{P}}$, inria:hasNationality)
books	($\underline{\text{P}}$, inria:hasAuthor), ($\underline{\text{P}}$, inria:hasContactInfo), ($\underline{\text{P}}$, inria:hasRole)($\underline{\text{P}}$, inria:hasTel), ($\underline{\text{P}}$, inria:hasNationality), ($\underline{\text{P}}$, inria:hasRole)
labs	($\underline{\text{P}}$, labInria:hasLocation), ($\underline{\text{P}}$, labInria:hasName), ($\underline{\text{P}}$, labInria:hasGPS)

object domain	
item key	(attr. name, attr. value)
articles	($\underline{\text{O}}$, inria:bar), ($\underline{\text{O}}$, "Bar"), ($\underline{\text{O}}$, "American")
books	($\underline{\text{O}}$, inria:foo), ($\underline{\text{O}}$, "Foo"), ($\underline{\text{O}}$, "+33 12345678"), ($\underline{\text{O}}$, "French"), ($\underline{\text{O}}$, "+33 1234879"), ($\underline{\text{O}}$, "Professor")
labs	($\underline{\text{O}}$, labInria:location), ($\underline{\text{O}}$, "ResearchLabs"), ($\underline{\text{O}}$, "48.710715,2.17545")

attribute-subset domain	
item key	(attr. name, attr. value)
$\underline{\text{S}}\|$inria:article1	(articles, ϵ)
$\underline{\text{S}}\|$inria:bar	(articles, ϵ)
$\underline{\text{S}}\|$inria:book1	(books, ϵ)
$\underline{\text{S}}\|$inria:book2	(books, ϵ)
$\underline{\text{S}}\|$inria:foo	(books, ϵ)
$\underline{\text{S}}\|$inria:_:uid1	(books, ϵ)
$\underline{\text{S}}\|$labinria:lab1	(labs, ϵ)
$\underline{\text{S}}\|$labInria:location	(labs, ϵ)
$\underline{\text{P}}\|$inria:hasAuthor	(articles, ϵ), (books, ϵ)
$\underline{\text{P}}\|$inria:hasName	(articles, ϵ), (books, ϵ)
$\underline{\text{P}}\|$inria:hasNationality	(articles, ϵ), (books, ϵ)
$\underline{\text{P}}\|$inria:hasTel	(books, ϵ)
$\underline{\text{P}}\|$inria:hasRole	(books, ϵ)
$\underline{\text{P}}\|$inria:hasContactInfo	(books, ϵ)
$\underline{\text{P}}\|$labInria:hasName	(labs, ϵ)
$\underline{\text{P}}\|$labinria:hasLocation	(labs, ϵ)
$\underline{\text{P}}\|$inria:hasLocation	(labs, ϵ)
$\underline{\text{P}}\|$labInria:hasGPS	(labs, ϵ)
$\underline{\text{O}}\|$inria:bar	(articles, ϵ)
$\underline{\text{O}}\|$"Bar"	(articles, ϵ)
$\underline{\text{O}}\|$"American"	(articles, ϵ)
...	...
$\underline{\text{O}}\|$"48.710715,2.17545"	(labs, ϵ)
$\underline{\text{SP}}\|$inria:article1$\|$inria:hasAuthor	(articles, ϵ)
$\underline{\text{SP}}\|$inria:bar$\|$inria:hasName	(articles, ϵ)
$\underline{\text{SP}}\|$inria:bar$\|$inria:hasNationality	(articles, ϵ)
...	...
$\underline{\text{SP}}\|$labInria:lab1$\|$labInria:hasName	(labs, ϵ)
$\underline{\text{PO}}\|$inria:hasName$\|$"Bar"	(articles, ϵ)
$\underline{\text{PO}}\|$inria:hasNationality$\|$"American"	(articles, ϵ)
$\underline{\text{PO}}\|$inria:hasAuthor$\|$inria:Bar	(articles,ϵ)
...	...
$\underline{\text{PO}}\|$labInria:hasGPS$\|$"48.710715,2.17545"	(labs, ϵ)
$\underline{\text{SO}}\|$inria:article1$\|$inria:Bar	(articles, ϵ)
$\underline{\text{SO}}\|$inria:bar$\|$"Bar"	(articles, ϵ)
$\underline{\text{SO}}\|$inria:bar$\|$"American"	(articles, ϵ)
...	...
$\underline{\text{SO}}\|$labInria:location$\|$"48.710715,2.17545"	(labs, ϵ)
$\underline{\text{SPO}}\|$inria:bar$\|$inria:hasName$\|$"Bar"	(articles, ϵ)
...	...

Figure 5.7: Sample collection of graphs (left) and their ATT (center) and ATS (right) indexing.

sd_1 and as the index overgrows it, we split over multiple domains. Similarly, ATT uses, *property domains* named $pd_1, pd_2, \ldots$ and *object domains* named $od_1, od_2, \ldots$ For $s_1\ p_1\ o_1$., ATT builds: $(sd_i, U_{\mathtt{G}}, \underline{s}, s_1)$, $(pd_j, U_{\mathtt{G}}, \underline{p}, p_1)$ and $(od_k, U_{\mathtt{G}}, \underline{o}, o_1)$, where: sd_i, pd_j and od_k are the domains where we currently insert respectively new subject, property, and object index entries. Based on an ATT index, the URIs of the graphs featuring the constant and URI values appearing in a query can be extracted from SimpleDB. Another indexing strategy is ATS (for attribute subset). It uses a single default domain which is split as the index grows. From $s_1\ p_1\ o_1$., ATS builds 7 item names: $\underline{s}s_1$, $\underline{p}p_1$, $\underline{o}o_1$, $\underline{sp}s_1p_1$, $\underline{so}s_1o_1$, $\underline{po}p_1o_1$ and $\underline{spo}s_1p_1o_1$. Each such item has a single attribute whose name is $U_{\mathtt{G}}$, and whose value is ϵ. Strategy ATS builds a larger index, in exchange for less *get* calls required to identify the relevant graphs to a given query.

Example 25 (Query evaluation in AMADA) *Consider the collection of graphs shown in Figure 5.7 together with the corresponding ATT and ATS indexing.*

Using the ATT index, the evaluation of the query $q(x)$:- x inria:hasAuthor "Foo" ., x inria:hasContactInfo y . *first performs the SimpleDB look-up queries:*

- *q1: SELECT itemName() FROM pd1 WHERE* $\underline{P}$ *= inria:hasAuthor;*
- *q2: SELECT itemName() FROM od1 WHERE* $\underline{O}$ *= "Foo";*
- *q3: SELECT itemName() FROM pd1 WHERE* $\underline{P}$ *= inria:hasContactInfo;*

AMADA *then intersects the results of q1 and q2 to ensure that the* inria:hasAuthor *property and the* ''Foo'' *value occur in the same graph. The result is then* unioned *with the result of q3 to obtain the graphs on which the BGP query will be evaluated. Indeed, SPARQL semantics allows matches for this query to span over multiple graphs, i.e., query results are defined against a global "merged" graph.*

Based on the ATS index, the evaluation of the same query first performs the SimpleDB look-up queries:

- *q1: GetAttributes(d1,* $\underline{PO}\|$*inria:hasAuthor*$\|$*"Foo")*
- *q2: GetAttributes(d1,* $\underline{P}\|$*inria:hasContactInfo)*

AMADA *then intersects the results of q1 and q2 to obtain the graphs against which the query will be evaluated.*

Chapter 6

Perspectives

In the next years, I will continue the ongoing work presented in this HDR thesis on RDF query answering robust to updates (cf. Section 3.2), on managing XML documents with RDF annotations (cf. Section 4.2), and on RDF data management in clouds (cf. Section 5.2). In addition, I will work on the following topics.

6.1 Multidimensional analysis of RDF graphs

I am currently working on the multidimensional analysis of RDF graph with Dario Colazzo (MCF, Univ. Paris-Sud), Ioana Manolescu (DR, Inria Saclay–Île-de-France), and Alexandra Roatiş (PhD student, Univ. Paris-Sud, co-advised by Dario, Ioana, and me).

Multidimensional analysis of data allows focusing on data according to some criteria of interest, possibly at several levels of granularity. Its killer application is Business Intelligence, which accounts for the performances of public or private organizations w.r.t. strategic criteria, so as to enlighten their decision makers. So far, multidimensional data analysis techniques have been confined to the relational data model, as almost all business data is relational. However, with the advent of Open Data and the worldwide project[1] of making such data available on the Web in RDF, I have obtained a three years grant for investigating multidimensional analysis of RDF graphs. This research is funded by *Région Paris–Île-de-France*[2] and the *Pôle d'Excellence Digiteo*[3].

Starting from the well-established multidimensional analysis techniques for relational data, we are currently devising their RDF counterparts, taking into account the essential RDF features of implicit information (through entailment), incomplete information (through blank nodes), etc.

6.2 Practical algorithms for ontology-based data management

I will work on practical algorithms for ontology-based data management with Jean-François Baget (CR, Inria Sophia-Antipolis Méditerranée), Meghyn Bienvenu (CR, CNRS), Marie-Laure Mugnier (PR, Univ. Montpellier), and Marie-Christine Rousset (PR, Univ. Grenoble).

Ontology-based data management is a quite young area of study, and despite important recent advances, including the identification of interesting tractable ontology languages (e.g., DL-lite), much work remains to be done in designing *scalable* ontology-based data management algorithms. Also, in real-world applications involving dynamic data, it is very likely that *inconsistencies* arise, rendering standard querying algorithms useless (as everything is entailed from a contradiction). Appropriate techniques for dealing with inconsistent data are thus crucial to the successful use of ontology-based data management in practice, yet have been little explored so far.

Scalable query answering and handling inconsistencies in lightweight Description Logics is at the core of the PAGODA project that has been accepted by the Agence Nationale pour la Recherche[4] in 2012. Also, Despoina Trivela (PhD student, NTUA, Greece) will join Meghyn and me to work on these topics for a six months ERASMUS study period[5], starting in november 2012.

[1] The linked data project: `http://linkeddata.org`

[2] `http://www.dimlsc.fr`

[3] `http://www.digiteo.fr`

[4] `http://www.agence-nationale-recherche.fr/`

[5] `http://ec.europa.eu/education/lifelong-learning-programme/doc80_en.htm`

Bibliography

[AAC$^+$08] Serge Abiteboul, Tristan Allard, Philippe Chatalic, Georges Gardarin, A. Ghitescu, François Goasdoué, Ioana Manolescu, Benjamin Nguyen, M. Ouazara, A. Somani, Nicolas Travers, Gabriel Vasile, and Spyros Zoupanos. Webcontent: efficient p2p warehousing of web data. *Proceedings of the Very Large Data Base Endowment (PVLDB)*, 1(2), 2008. (Demonstration paper).

[ACG$^+$04a] Philippe Adjiman, Philippe Chatalic, François Goasdoué, Marie-Christine Rousset, and Laurent Simon. Distributed reasoning in a peer-to-peer setting. In *European Conference on Artificial Intelligence (ECAI)*, 2004. (Poster paper).

[ACG$^+$04b] Philippe Adjiman, Philippe Chatalic, François Goasdoué, Marie-Christine Rousset, and Laurent Simon. Raisonnement distribué dans un environnement de type pair-à-pair. In *Journées Nationales sur la résolution Pratique de Problèmes NP-Complets (JNPC)*, 2004.

[ACG$^+$05a] Philippe Adjiman, Philippe Chatalic, François Goasdoué, Marie-Christine Rousset, and Laurent Simon. Scalability study of peer-to-peer consequence finding. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.

[ACG$^+$05b] Philippe Adjiman, Philippe Chatalic, François Goasdoué, Marie-Christine Rousset, and Laurent Simon. Somewhere in the semantic web. In *International Workshop on Principles and Practice of Semantic Web Reasoning (PPSWR)*, 2005.

[ACG$^+$06] Philippe Adjiman, Philippe Chatalic, François Goasdoué, Marie-Christine Rousset, and Laurent Simon. Distributed reasoning in a peer-to-peer setting: Application to the semantic web. *Journal of Artificial Intelligence Research (JAIR)*, 25:269–314, 2006.

[AG08a] Nada Abdallah and François Goasdoué. Calcul de conséquences pour le test d'extension conservative dans un système pair-à-pair. In *Journées Francophones de Programmation par Contraintes (JFPC)*, 2008.

[AG08b] Nada Abdallah and François Goasdoué. Systèmes pair-à-pair sémantiques et extension (non) conservative d'une base de connaissances. In *Journées de bases de données avancées (BDA)*, 2008.

[AG09] Nada Abdallah and François Goasdoué. Non-conservative extension of a peer in a p2p inference system. *AI Communications (AICOM)*, 22(4):211–233, 2009.

[AGP09] Marcelo Arenas, Claudio Gutierrez, and Jorge Pérez. Foundations of rdf databases. In *Reasoning Web Summer School*, 2009.

[AGR07] Philippe Adjiman, François Goasdoué, and Marie-Christine Rousset. Somerdfsin the semantic web. *Journal on Data Semantics (JODS)*, 8:158–181, 2007.

[AGR09] Nada Abdallah, François Goasdoué, and Marie-Christine Rousset. Dl-liter in the light of propositional logic for decentralized data management. In *Internet Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

[AGR10] Nada Abdallah, François Goasdoué, and Marie-Christine Rousset. Gestion décentralisée de données en dl-lite. In *Congrès francophone de Reconnaissance des Formes et d'Intelligence Artificielle (RFIA)*, 2010.

[AHV95] Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[AM00] E. Amir and S. McIlraith. Partition-based logical reasoning. In *International Conference on the Principles of Knowledge Representation and Reasoning (KR)*, pages 389–400, 2000.

[AM01] E. Amir and S. McIlraith. Theorem proving with structured theories. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 624–634, 2001.

[AMMH07] Daniel J. Abadi, Adam Marcus, Samuel R. Madden, and Kate Hollenbach. Scalable semantic web data management using vertical partitioning. In *International Conference on Very Large Databases (VLDB)*, 2007.

[AMR$^+$12] Serge Abiteboul, Ioana Manolescu, Philippe Rigaux, Marie-Christine Rousset, and Pierre Senellart. *Web Data Management*. Cambridge University Press, 2012.

[AYCLS01] Sihem Amer-Yahia, SungRan Cho, Laks V. S. Lakshmanan, and Divesh Srivastava. Minimization of tree pattern queries. In *ACM SIGMOD International Conference on the Management of Data (SIGMOD)*, 2001.

[BCM$^+$03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[BGKM12] Francesca Bugiotti, François Goasdoué, Zoi Kaoudi, and Ioana Manolescu. Rdf data management in the amazon cloud. In *International Workshop on Data analytics in the Cloud (DanaC)*, 2012.

[BK03] Jeen Broekstra and Arjohn Kampman. Inferencing and truth maintenance in RDF Schema: Exploring a naive practical approach. In *Workshop on Practical and Scalable Semantic Systems (PSSS)*, 2003.

[BKO$^+$11] Barry Bishop, Atanas Kiryakov, Damyan Ognyanoff, Ivan Peikov, Zdravko Tashev, and Ruslan Velkov. OWLIM: A family of scalable semantic repositories. *Semantic Web (JSW)*, 2(1), 2011.

[CGL$^+$07] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *dl-lite* family. *Journal of Automated Reasoning (JAR)*, 39(3):385–429, 2007.

[CGLR04] D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Logical fondation of peer-to-peer data integration. In *ACM SIGACT-SIGMOD-SIGART Conference on the Principles of Database Systems (PODS)*, 2004.

[CHKS07] B. Cuenca Grau, I. Horrocks, Y. Kazakov, and U. Sattler. Just the right amount: extracting modules from ontologies. In *International World Wide Web Conference (WWW)*, 2007.

[CL10] Roger Castillo and Ulf Leser. Selecting materialized views for RDF data. In *Workshops of the International Conference on Web Engineering (ICWE)*, 2010.

[CM08] Michel Chein and Marie-Laure Mugnier. *Graph-based Knowledge Representation: Computational Foundations of Conceptual Graphs*. Springer Publishing Company, Incorporated, 2008.

[CR73] C.-L. Chang and R.C.-T. Lee. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, 1973.

[CRL10] Roger Castillo, Christian Rothe, and Ulf Leser. RDFMatView: Indexing RDF data using Materialized SPARQL Queries. In *International Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS)*, 2010.

[FKLZ04] E. Franconi, G. Kuper, A. Lopatenko, and I. Zaihrayeu. Queries and updates in the coDB peer-to-peer database system. In *International Conference on Very Large Databases (VLDB)*, 2004.

[GKK$^+$11a] François Goasdoué, Konstantinos Karanasos, Yannis Katsis, Julien Leblay, Ioana Manolescu, and Stamatis Zampetakis. Growing Triples on Trees: an XML-RDF Hybrid Model for Annotated Documents. In *International Workshop on Searching and Integrating New Web Data Sources (VLDS)*, 2011.

[GKK$^+$11b] François Goasdoué, Konstantinos Karanasos, Yannis Katsis, Julien Leblay, Ioana Manolescu, and Stamatis Zampetakis. Growing Triples on Trees: an XML-RDF Hybrid Model for Annotated Documents. In *Journées de Bases de Données Avancées (BDA)*, October 2011.

[GKK$^+$12] François Goasdoué, Konstantinos Karanasos, Yannis Katsis, Julien Leblay, Ioana Manolescu, and Stamatis Zampetakis. Des triplets sur des arbres: un modèle hybride XML-RDF pour documents annotés. *Ingénierie des Systèmes d'Information (ISI)*, 2012.

[GKLM10a] François Goasdoué, Konstantinos Karanasos, Julien Leblay, and Ioana Manolescu. RDFViewS: a storage tuning wizard for RDF applications. In *International Conference on Knowledge Management (CIKM)*, 2010. (Demonstration paper).

[GKLM10b] François Goasdoué, Konstantinos Karanasos, Julien Leblay, and Ioana Manolescu. Materialized View-Based Processing of RDF Queries. In *Bases de Données Avancées (BDA)*, 2010.

[GKLM11a] François Goasdoué, Konstantinos Karanasos, Julien Leblay, and Ioana Manolescu. View selection in semantic web databases. *Proceedings of the Very Large Data Bases endowment (PVLDB)*, 5(2), 2011.

[GKLM11b] François Goasdoué, Konstantinos Karanasos, Julien Leblay, and Ioana Manolescu. RDFViewS: A Storage Tuning Wizard for RDF Applications. In *Journées de Bases de Données Avancées (BDA)*, October 2011.

[GLW06] S. Ghilardi, C. Lutz, and F. Wolter. Did i damage my ontology? a case for conservative extensions in description logics. In *International Conference on the Principles of Knowledge Representation and Reasoning (KR)*, 2006.

[GMR12a] François Goasdoué, Ioana Manolescu, and Alexandra Roatis. Getting more RDF support from relational databases. In *World Wide Web Conference (WWW)*, 2012. (Poster paper).

[GMR12b] François Goasdoué, Ioana Manolescu, and Alexandra Roatiş. Répondre aux requêtes par reformulation dans les bases de données RDF. In *Congrès francophone Reconnaissance des Formes et Intelligence Artificielle (RFIA)*, 2012.

[GOP11] Georg Gottlob, Giorgio Orsi, and Andreas Pieris. Ontological queries: Rewriting and optimization. In *International Conference on Data Engineering (ICDE)*, 2011. Keynote.

[GR03a] François Goasdoué and Marie-Christine Rousset. Querying distributed data through distributed ontologies: A simple but scalable approach. In *International Workshop on Information Integration on the Web (IIWeb)*, 2003.

[GR03b] François Goasdoué and Marie-Christine Rousset. Querying distributed data through distributed ontologies: A simple but scalable approach. *IEEE Intelligent Systems (IEEE IS)*, 18(5):60–65, 2003.

[GR10] François Goasdoué and Marie-Christine Rousset. Modules sémantiques robustes pour une réutilisation saine en dl-lite. In *Bases de Données Avancées (BDA)*, 2010.

[GR12] François Goasdoué and Marie-Christine Rousset. Robust module-based data management. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2012.

[Gru09] Tom Gruber. Ontology. In *Encyclopedia of Database Systems*. Springer US, 2009.

[Hal01] Alon Y. Halevy. Answering queries using views: A survey. *The Very Large Databases Journal (VLDBJ)*, 10(4), 2001.

[HIST03] A. Y. Halevy, Z. G. Ives, D. Suciu, and I. Tatarinov. Schema mediation in peer data management systems. In *International Conference on Data Engineering (ICDE)*, 2003.

[IJ84] Tomasz Imielinski and Witold Lipski Jr. Incomplete information in relational databases. *Journal of the ACM (JACM)*, 31(4), 1984.

[Ino91] K. Inoue. Consequence-finding based on ordered linear resolution. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 158–164, 1991.

[Ino92] K. Inoue. Linear resolution for consequence finding. *Artificial Intelligence (AI)*, 2-3(56):301–353, 1992.

[KLWW08] B. Konev, C. Lutz, D. Walther, and F. Wolter. Semantic modularity and module extraction in description logics. In *European Conference on Artificial Intelligence (ECAI)*, 2008.

[KMK08] Zoi Kaoudi, Iris Miliaraki, and Manolis Koubarakis. RDFS reasoning and query answering on DHTs. In *International Semantic Web Conference (ISWC)*, 2008.

[KPS+09] R. Kontchakov, L. Pulina, U. Sattler, T. Schneider, P. Selmer, F. Wolter, and M. Zakharyaschev. Minimal module extraction from DL-Lite ontologies using QBF solvers. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

[KWW09a] B. Konev, D. Walther, and F. Wolter. Forgetting and uniform interpolation in extensions of the description logic EL. In *International Workshop on Description Logics (DL)*, 2009.

[KWW09b] B. Konev, D. Walther, and F. Wolter. Forgetting and uniform interpolation in large-scale description logic terminologies. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2009.

[Len11] Maurizio Lenzerini. Ontology-based data management. In *ACM Conference on Information and Knowledge Management (CIKM)*, 2011.

[Mar00] Pierre Marquis. *Handbook on Defeasible Reasoning and Uncertainty Management Systems*, volume 5, chapter Consequence Finding Algorithms, pages 41–145. Kluwer Academic Publisher, 2000.

[MR72] E. Minicozzi and R. Reiter. A note on linear resolution strategies in consequence-finding. *Artificial Intelligence (AI)*, 3(1-3):175–180, 1972.

[NW08] Thomas Neumann and Gerhard Weikum. RDF-3X: a RISC-style engine for RDF. *Proceedings of the VLDB Endowment (PVLDB)*, 1(1), 2008.

[NW09] Thomas Neumann and Gerhard Weikum. Scalable join processing on very large RDF graphs. In *ACM SIGMOD International Conference on the Management of Data (SIGMOD)*, 2009.

[NW10] Thomas Neumann and Gerhard Weikum. x-RDF-3X: Fast querying, high update rates, and consistency for RDF databases. *Proceedings of the VLDB Endowment (PVLDB)*, 3(1), 2010.

[PBJ+09] O. Palombi, G. Bousquet, D. Jospin, S. Hassan, L. Revéret, and F. Faure. My corporis fabrica: A unified ontological, geometrical and mechanical view of human anatomy. In *Modelling the Physiological Human, 3D Physiological Human Workshop (3DPH)*, 2009.

[RG03] Raghu Ramakrishnan and Johannes Gehrke. *Database management systems (3rd edition)*. McGraw-Hill, 2003.

[RN10] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Prentice Hall, 2010.

[Rob65] J. A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM (JACM)*, 12(1):23–41, 1965.

[SGK+08] Lefteris Sidirourgos, Romulo Goncalves, Martin Kersten, Niels Nes, and Stefan Manegold. Column-store support for RDF data management: not all swans are white. *Proceedings of the VLDB Endowment (PVLDB)*, 1(2), 2008.

[SPS09] Heiner Stuckenschmidt, Christine Parent, and Stefano Spaccapietra, editors. *Modular Ontologies: Concepts, Theories and Techniques for Knowledge Modularization*, volume 5445 of *Lecture Notes in Computer Science*. Springer, 2009.

[TIM$^+$03] I. Tatarinov, Z. Ives, J. Madhavan, A. Halevy, D. Suciu, N. Dalvi, X. Dong, Y. Kadiyska, G. Miklau, and P. Mork. The piazza peer data management project. *ACM SIGMOD Records*, 32(3), 2003.

[TLS01] Dimitri Theodoratos, Spyros Ligoudistianos, and Timos K. Sellis. View selection for designing the global data warehouse. *Data and Knowledge Engineering (DKE)*, 39(3), 2001.

[TS97] Dimitri Theodoratos and Timos K. Sellis. Data warehouse configuration. In *International Conference on Very Large Databases (VLDB)*, 1997.

[UvHSB11] Jacopo Urbani, Frank van Harmelen, Stefan Schlobach, and Henri Bal. QueryPIE: Backward reasoning for OWL Horst over very large knowledge bases. In *International Semantic Web Conference (ISWC)*, 2011.

[Var82] M. Y. Vardi. The complexity of relational query languages. In *ACM Symposium on Theory of Computing (STOC)*, 1982.

[W3Ca] W3C. Resource description framework. http://www.w3.org/2001/sw/wiki/RDF.

[W3Cb] W3C. Resource description framework in XHTML attributes. http://www.w3.org/2010/02/rdfa.

[W3Cc] W3C. Sparql protocol and query language. http://www.w3.org/2001/sw/wiki/SPARQL.

[W3Cd] W3C. Web ontology language. http://www.w3.org/2001/sw/wiki/OWL.

[W3Ce] W3C. Xml query. http://www.w3.org/XML/Query.

[W3Cf] W3C. Xml schema. http://www.w3.org/XML/Schema.

[WKB08] Cathrin Weiss, Panagiotis Karras, and Abraham Bernstein. Hexastore: sextuple indexing for Semantic Web data management. *Proceedings of the VLDB Endowment (PVLDB)*, 1(1), 2008.

[WWT$^+$09] K. Wang, Z. Wang, R. W. Topor, J. Z. Pan, and G. Antoniou. Concept and role forgetting in ALC ontologies. In *International Semantic Web Conference (ISWC)*, 2009.

[WWTP08] Z. Wang, K. Wang, R. W. Topor, and J. Z. Pan. Forgetting concepts in DL-Lite. In *Extended Semantic Web Conference (ESWC)*, 2008.