Distributed knowledge sharing and production through collaborative e-Science platforms

Alban Gaignard

UNIVERSITÉ DE NICE - SOPHIA ANTIPOLIS

ÉCOLE DOCTORALE STIC
Sciences et Technologies de l'Information et de la Communication

THÈSE

pour l'obtention du grade de

Docteur en Sciences

de l'Université de Nice-Sophia Antipolis
Mention : INFORMATIQUE

Présentée et soutenue par

Alban GAIGNARD

Distributed knowledge sharing and production through collaborative e-Science platforms

Thèse dirigée par Johan MONTAGNAT

preparée au laboratoire I3S, CNRS UMR-7271, équipe MODALIS

soutenue le 15 mars 2013

Jury

Rapporteurs :	Oscar CORCHO	-	Associate Professor, Universidad Politécnica de Madrid
	Ollivier HAEMMERLÉ	-	Professeur, Université Toulouse le Mirail
Directeur :	Johan MONTAGNAT	-	DR CNRS, Laboratoire I3S
Président :	Andrea TETTAMANZI	-	Professeur, Université Nice Sophia Antipolis
Examinateurs :	Olivier CORBY	-	CR INRIA, Laboratoire I3S
	Bernard GIBAUD	-	CR INSERM, Laboratoire LTSI
Invitée :	Catherine FARON ZUCKER	-	Maître de Conférence, Université Nice Sophia Antipolis

Remerciements

Je tiens à remercier en premier lieu Johan Montagnat, pour avoir accepté de se lançer avec moi dans cette aventure enrichissante. Ce travail doit beaucoup à sa rigueur méthodologique, expérimentale, à son goût pour les idées et projets scientifiques mêlant à la fois réalisme et ambition, à son exigence, à sa capacité d'écoute et à ses encouragements constants.

Ce travail n'aurait pas pu voir le jour sans le soutien actif de la direction du laboratoire I3S, Luc Pronzato puis Michel Riveill. Ce soutien m'a permis, parallèlement à des activités transverses de support, de mener des activités de recherche sur une thématique scientifique qui me tient à coeur. Merci en particulier à Michel Riveill de m'avoir soutenu et mis le pied à l'étrier, dès notre toute première rencontre.

Je tiens également à remercier les membres du jury d'avoir accepté d'évaluer mes travaux. Merci à Ollivier Haemmerlé pour l'étendue de son rapport et le temps consacré à l'examen de ce manuscrit, pour ses remarques constructives et ses encouragements.

Quisiera también agradecer a Oscar Corcho por aceptar esta tarea de evaluación. Gracias por su tiempo, sus comentarios constructivos y sus sugestiones para seguir adelante.

Ces travaux ont été menés à l'interface entre plusieurs disciplines dans un contexte collaboratif (projets ANR NeuroLOG et VIP). Merci à Bernard, Tristan, Franck, Germain, Rafael pour l'efficacité, le sérieux, et la qualité des échanges dans ces travaux d'équipe (que les distances soient courtes, pour Franck mon co-bureau ou plus longues).

Merci également à Olivier et Catherine pour leur grande disponibilité et leur implication dans ces travaux menés à l'interface entre systèmes distribués et ingénierie des connaissances.

I would also like to thank Silvia for allowing me to participate in the organization of an international scientific conference. This event was really a nice and enriching experience.

Pour leur bonne humeur, l'ambiance chaleureuse, les repas partagés, merci aux membres des équipes Rainbow, Modalis, et Wimmics, et particulièrement "aux" Philippes (Philippe L., Philippe C., Filip K., Philippe R.), à Mireille, Diane, Clémentine, Nadia, Sébastien, Simon, Stéphane, Jean-Yves, Gaëtan, Javier, Tram, Ketan.

Enfin, merci à Stéphane R.-D., pour sa curiosité, ses relectures approfondies et ses questionnements philosophiques. Plus généralement, merci à mes proches, famille et amis, pour leur présence indéfectible.

Merci à Ana pour sa patience, sa confiance, son soutien, et ces belles années.

A Maya & Adèle

Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

Contents

1.1 Context, motivations and objectives

From the first digital computers to the *Data Deluge*.

Digital computers have been originally designed in the 1950s to automate calculation tasks [Holbrook and Brown, 1982]: *"[...] calculations were performed in part with slide rules and, mainly, with desk calculators. The magnitude of this load of very tedious routine computation and the necessity of carefully checking it indicated a need for new methods"*. As an extension of these calculation tasks, digital computers have rapidly been dedicated to data processing and analysis.

In parallel with the continuous increase of computing and storage capacity of digital computers [Moore, 1965], the need for (tele)communicating data through computers led to their massive and systematic interconnection in networks. The Internet was born in the 1990s as a global world-wide network of computers. The (World Wide) Web [Berners-Lee et al., 1994] led to the fast adoption of the Internet. The Web has generalized, on a global scale, the access to information through hypertext documents. Following the example of interconnected computers in networks, Web documents are linked together through HyperText links. These links allow users to read and transparently navigate, without specifying documents locations, into a global network of documents.

The generalization of (i) computers, with to a continuous increase of their computing, storage and communication capacities, and (ii) a Web which evolved from interconnected

documents towards interconnected data (or things), leads nowadays to massive data acquisition, production, and publication (also known as *Data Deluge*) which exceeds by far our current data processing and analysis capacities [Bell et al., 2009] [Baraniuk, 2011].

As an example, high energy physics experiments conducted by CERN on the Large Hadron Colider (LHC) produced in 2010 13 petabytes of data. This amount of data would represent a 14 km high tower of CD-ROMs [Brumfiel, 2011]. To support these experiments, data is processed through a dedicated large-scale distributed computing infrastructure, the LHC Computing Grid, comprising around 200.000 computing cores, 150 petabytes of storage, and distributed over 34 countries. As another example, the Google web search engine was processing in 2008 more than twenty petabytes of data per day [Dean and Ghemawat, 2008]. Beyond pure data, we also witness a noticeable increase of medical knowledge discovery since 800.000 medical articles were catalogued in 2008 and 1 million articles were expected for 2012 [Gillam et al., 2009].

This massive and often poorly coordinated data production raises the general issue of finding the appropriate information in an unprecedented amount of data, spread at an unprecedentedly large scale.

Translational research in the *Data Deluge*.

Scientific activities nowadays face the data explosion challenge. Paradoxically, continuously producing more data does not imply that we have a better access to information and better means to interpret, analyze data, possibly opening opportunities for scientific discoveries. Data is massively produced from diverse sources, geographically dispersed, and constrained under their proper governance model. Data explosion is not only a matter of volume but also a matter of diversity [Goble and Stevens, 2008]: *"Not only are the datasets growing in size and number, but they are only partly coordinated and often incompatible"* [Goble and Roure, 2009].

Translational science results from the need to cross several data, and several methods, both originating from diverse sources and scientific disciplines to more rapidly translate research outcomes. Translational science needs to cope with the multiplicity of data sources, their diversity, their size, while still considering data as hardly relocatable. The cost of copying large amount of data to a single workstation might be non realistic due to the multiplicity and the size of sources. In addition, distributing data to several independent research centers may require dedicated and expensive infrastructures, which are often not affordable in the context of academic scientific research: *"For research to be affordable, data analysis must increasingly be done where data sets reside [...]"* [Bell et al., 2009]. Moreover, in a life-science context, biomedical data are sensitive and non relocatable for legal or ethical reasons.

Translational medicine thus faces the challenging issues of efficiently and coherently integrating large, but also diverse, distributed data, involving both structural[1] and semantic[2] heterogeneity.

[1] *e.g.* data may be expressed in incompatible formats.

[2] data may refer to several conceptualizations, terminologies or thesauri, whose meanings have several

While the data deluge raises the issue of the mass and the distribution of data, translational research, in this context, raises the issues of poorly coordinated data production from diverse and autonomous data sources, resulting in data heterogeneity. Translational research discoveries are consequently conditioned by the way we handle the distribution of massive, and diverse, data. How to search over multiple distributed and autonomous data sources ? How to efficiently find data ? How to interpret the resulting data originating from these diverse data sources ?

Understanding and integrating diverse data through Knowledge Engineering methods and Semantic Web technologies.

To coherently integrate multi-sources data, scientists need to have a clear understanding of the content of data sources. Knowledge Engineering, a discipline that emerged from Artificial Intelligence, is dedicated to build scalable knowledge based systems [Studer et al., 1998]. In the context of the continuous increase of available data sources, Knowledge Engineering has gained a lot of interest to ease the understanding of diverse data and their coherent integration. The Semantic Web is a technological layer based on Knowledge Engineering principles and applied at the scale of the Web: *"information is given a well-defined meaning, better enabling computers and people to work in cooperation"* [Berners-Lee et al., 2001]. More recently, Linked Data principles [Bizer et al., 2009a] more directly addressed the issue of the increase of data sources at the scale of the Web: *"Linked Data is simply about using the Web to create typed links between data from different sources. These may be as diverse as databases maintained by two organizations in different geographical locations, or simply heterogeneous systems within one organization that, historically, have not easily interoperated at the data level"*. We have been moving from human-readable interlinked documents (the Web) to machine-readable interlinked resources, at global scale, with explicit and formal meaning (Knowledge Engineering).

Semantic e-Science.

Translational science relies on computing infrastructures to foster cooperations/collaborations and shorten the "time-to-discovery". E-Science [Hey and Trefethen, 2005] appeared in this context of "Data Deluge" in which scientists need to achieve compute intensive tasks in highly distributed environments. We envisage in this thesis collaborative e-Science platforms as means to (i) perform *in-silico* experiments (ii) share the involved resources, and (iii) produce new meaningful information. For instance, such platforms should allow scientists to evaluate a single data analysis procedure onto several shared databases. Another typical use case would consist in comparing several data analysis procedures provided by several partners with a common reference database.

Reusing and re-purposing data sources is a major concern in this context of growing availability of, possibly opened, (linked-)data sources [Fox and Hendler, 2009]. Semantic e-Science relies on Semantic Web technologies and standards to address the sharing

precisions.

and cooperation challenges of data-intensive science. Semantic approaches for e-Science allow scientists to refer to common conceptual referents (ontologies or controlled vocabularies) so that the interpretation of diverse multi-source data is facilitated. Beyond data integration, methods integration is also facilitated by Semantic approaches, easing the composition of data processing units through scientific workflows.

Objectives of this work.

To address the data volume, data distribution and data heterogeneity issues arising from medical translational research in the *Data Deluge*, the main objective of this thesis is to tackle the coherent production and sharing of knowledge in life-science.

The main challenge for knowledge sharing is to be able to perform advanced querying on diverse and multi-source data while still considering data sources as dynamic and autonomous.

Scientific data analysis procedures participate to the massive production of data whose interpretation is difficult, even for e-Scientists. Data might be considered as noisy, foggy, due to their volume and to the lack of clear meaning, thus hampering their interpretation and analysis. The underlying idea of knowledge production is that e-Science data analysis procedures, without being instrumented with knowledge engineering, contribute to the *Data Deluge* in its negative aspect. They massively feed data stores with hardly searchable / interpretable data. How to, on one hand, ease the access and the interpretation of massive diverse multi-sources data, while on the other hand producing new data, positively feeding the *Data Deluge* ?

The knowledge production challenge addressed in this thesis consists in tackling the production of a limited amount of meaningful data – opposed to the systematic production of generic technical (meaningless) data – so that data interpretation keeps simple, but not simpler[3].

1.2 Research questions

In this thesis manuscript, we investigate **how to coherently produce and share knowledge from distributed Life-science resources**. To address this general research question, we will defend the following 3 thesis:

[3] *"Everything should be made as simple as possible, but not simpler."* – Albert Einstein

T_1 *In spite of the continuous increase of computing and storage capacities, the federation of distributed resources (versus centralized warehousing) becomes a cornerstone to enhance the adoption and the scalability of collaborative e-Science platforms.*

T_2 *Large-scale data analysis or processing is complex for end-users. Enhancing e-Science platforms with Knowledge Engineering reduces the complexity of experiments setup, and eases the exploitation of data analyzed or produced.*

T_3 *Widely spread into Life-science communities, Knowledge Engineering methods and Semantic Web technologies are at the heart of domain modeling, and a foundation for knowledge sharing and capitalization.*

While T_1 is related to how is envisaged the architectural setup for resource distribution in the general research question, T_2 and T_3 are more related to the semantic coherency of sharing and the expected knowledge resulting from the usage of collaborative platforms.

More precisely, we will address in the remainder of this manuscript the following concrete questions :

RQ_1 *How to collect knowledge from multiple distributed data sources.*
RQ_1 will lead to strong architectural constraints to address distribution issues, and covers the areas of Distributed databases, Distributed Query Processing, and Knowledge Representation (T_1, T_3).

RQ_2 *How to enable collaborations in a competitive environment.*
RQ_2 depends on architectural choices resulting from Q_1 and covers Security in Distributed Systems (T_1).

RQ_3 *How to optimize the semantic querying over distributed resources.*
RQ_3 also depends on RQ_1 architectural choices (T_1, T_3) and is directly related to distributed query processing, knowledge representation and retrieval.

RQ_4 *How to correctly annotate produced data with semantics.*
RQ_4 covers Knowledge Representation and semantic description of data analysis procedures (T_3).

RQ_5 *How to enhance the understanding of scientific data analysis results.*
RQ_5 also depends on RQ_4 and covers Provenance (in scientific data) and Automated Reasoning.

1.3 Thesis contributions

Investigating these concrete research questions led to three main contributions in the fields of distributed knowledge bases and scientific data analysis procedures (modeled and digitally represented through workflows). Two of them focus on knowledge sharing in the context of e-Science platforms, through *(i)* "secured collaborations", which address the control of data sharing in a competitive environment, and *(ii)* "knowledge base federation", which provides efficient and still expressive enough distributed query processing over scattered Semantic Web data graphs. The third contribution of this thesis focuses on the "production of knowledge based on the exploitation of workflow runs in e-Science platforms". It provides meaningful experiment summaries leveraging domain ontologies and eases the understanding of massively produced data through *"in silico"* experiments.

1.3.1 Secured collaborations

We propose in this thesis a model for secured collaborations in the context of distributed e-Science platforms. Distributed access control strategies have been proposed and implemented in the NeuroLOG neuroimaging platform, thus allowing the setup of multi-centric studies. This approach, based on classical public key infrastructures (PKIs), introduce an original and pragmatic strategy to enable collaborations in a competitive environment (RQ_2). This work has been published in the HealthGrid'09 conference as a short paper [Gaignard and Montagnat, 2009] and has been integrated and used in production in the NeuroLOG middleware [Montagnat et al., 2008b]. Secured collaborations are described in the chapter 3 of this manuscript.

1.3.2 Knowledge base federation

The second contribution of this thesis addresses the sharing of knowledge distributed over multiple data sources possibly participating in e-Science collaborative platforms. In the context of collaborative platforms dedicated to life-sciences, the autonomy property of participating data providers is fundamental. Indeed, for ethical or legal issues, it is generally not possible to re-locate sensitive data outside the data provider sites. For this reason, data warehousing approaches are disqualified. We propose in this thesis, transparent and efficient semantic federated querying strategies. Transparent, in the sense that the query engine interprets standard SPARQL queries and thus does not require explicit distribution directives to be provided by query designers. Effective, in the sense that the cost of transparent federated querying is reduced through a set of static and dynamic optimizations. Moreover, our approach is based on abstract knowledge graphs and thus allows mediating structural heterogeneity, as soon as data sources can provide a graph view on hosted data. This approach allows to efficiently collect knowledge (RQ_3) fragmented over distributed semantic repositories (RQ_1), thus easing knowledge sharing through collaborating data providers. Our approach for transparent semantic federated querying is precisely described in chapter 4 and its implementation is described in chapter 7. Our approach has been evaluated in a real controlled

distributed infrastructure in chapter 8. This work has also been published in the national workshop [Gaignard et al., 2012b] of the IC'12 conference, dedicated to semantic interoperability in e-Health, in the MICCAI-DCICTAI workshop [Gaignard et al., 2012a] of the MICCAI'12 international conference, dedicated to Data- and Compute-Intensive Clinical and Translational Imaging Applications, and as a research paper of the Web Intelligence (WI'12) international conference [Corby et al., 2012]. Finally, this work let us envisage the evolution of the NeuroLOG platform from classical relational database federation towards semantic knowledge base federation, thus enabling distributed reasoning. It opens new research perspectives such as those currently under investigation by the CrEDIBLE [4] initiative, tackling the new challenges of massive scientific data management.

1.3.3 Semantic scientific workflows

The last salient contribution of this thesis addresses the production of knowledge through the usage of e-Science platforms. On one hand, ontological modeling approaches and knowledge engineering techniques are nowadays a corner-stone to provide meaningful and coherent data integration, and thus ease the navigation within the deluge of heterogeneous data the scientific communities are facing. On the other hand, scientific workflows, based on service oriented architectures, became the agreed paradigm to process data and finally populate scientific databases. We propose in this thesis to bridge together scientific workflows and domain ontologies to (i) assist *in silico* experiment designers in semantically composing data processing services together, through semantically annotated composable services (RQ_4), and (ii) to assist scientists in the exploitation of their massive processed data (RQ_5) through semantic experiment summaries, based on technical fine-grained provenance information, and inference rules involving concepts and properties of domain ontologies. This work has been introduced in the IC'10 national workshop [Gaignard and Montagnat, 2010] dedicated to the medical semantic web, and presented in the KEOD'11 international conference [Gaignard et al., 2011], focusing on the requirements for role taxonomies to disambiguate the semantic annotation of data processing services. This work is extensively described in chapters 6, and in the experimental evaluations presented in chapter 8. Finally, this approach has been implemented through the NeuSemStore software framework introduced in chapter 7. NeuSemStore is currently used in production in the NeuroLOG neuroimaging platform and the VIP [Forestier et al., 2011a] virtual imaging platform.

1.4 Applicative context: neurosciences and medical image simulation

This work took place in the context of two projects funded by the French National Research Agency (ANR): the NeuroLOG project (ANR-06-TLOG-024) and the VIP project (ANR-09-COSI-03). These two projects share methodological commonalities as they both

[4] http://credible.i3s.unice.fr

heavily rely on Life-science ontologies and *in-silico* experiments performed through e-Science workflows.

1.4.1 Collaborative neuroscience (NeuroLOG project)

The NeuroLOG project [Montagnat et al., 2008b], [Michel et al., 2010], aimed at federating four neuroscience research centers in France. It enables collaborative neurosciences through the support of multi-centric studies and the coherent sharing of large population datasets and neuroimaging processing tools.

Motivations and challenges. The NeuroLOG project has been driven by four test-bed clinical application: multiple sclerosis, brain strokes, brain tumors and Alzheimer's disease. In this clinical context, neuroscientists manage, process and analyze a huge variety of data. Neuroimages differ by their nature (modalities for neuroimaging data such as T1 or T2 MRI, MRIs with gadolinium contrast agent, Diffusion MRI, etc.) and their structure (several neuroimaging data formats such as DICOM, Nifti, Analyze, etc.). Moreover neuroimaging studies relies on the setup of brain image databases in which raw data are complemented with descriptive metadata. These metadata may describe the context of the clinical studies, the data acquisition protocols, the examined subjects, and the scores of neuropsychological tests. Data and metadata involved in neuroscience studies are thus highly heterogeneous both in terms of nature and structure. Coherently sharing these data and metadata is challenging and requires advanced neuroimaging data integration.

Neuroscientists require means to achieve collaborative multi-centric studies, to coherently share valuable data or data analysis procedures. However, due to the sensitivity of data and the competitive nature of scientific research, neuroscientists want to collaborate but often in a limited extent, *i.e.* with a subset of the partners and over a subset of the available neuroimaging resources (both data and tools). The autonomy of neuroscience research centers must be guaranteed in terms of *(i)* daily operation of legacy databases, and (ii) access control over the hosted resources, which is partly conflicting with the setup of large-scale multi-centric studies.

Finally, neuroimaging data analysis pipelines involve generally several complex processing steps, and may be launched over large population datasets. The cost of these analysis procedures can be handled through large-scale distributed computing infrastructures such as the EGI European Grid. However, these infrastructures are difficult to comprehend for non IT experts. Although they propose security over distributed computing resources, they are not yet convincing in terms of sites autonomy, and resources access control.

Objectives. The NeuroLOG platform addresses several challenging issues when addressing the setup of multi-centric collaborative neurosciences. First it needs to leverage existing large-scale distributed computing infrastructures while still guaranteeing the autonomy of participating sites. Then, it needs to address the distribution and the

heterogeneity of neuroimaging data sources to ease the coherent sharing of both large population datasets and data analysis procedures.

NeuroLOG thus aims at easing the setup of neuroscience multi-centric studies through an ontology driven sharing of (i) complex data analysis workflows, enacted over dedicated large-scale distributed computing infrastructures, and (ii) autonomous, heterogeneous and distributed neuroimaging data sources.

Platform coarse-grained architecture. Figure 1.1 illustrates the main components of the NeuroLOG platform. It consists in a middleware which coordinates several neuroscience research centers while still guaranteeing their autonomous operation. NeuroLOG sites are thus loosely coupled. They present data and metadata interfaces which adapt, through a mediator, to the specificities of their legacy databases.

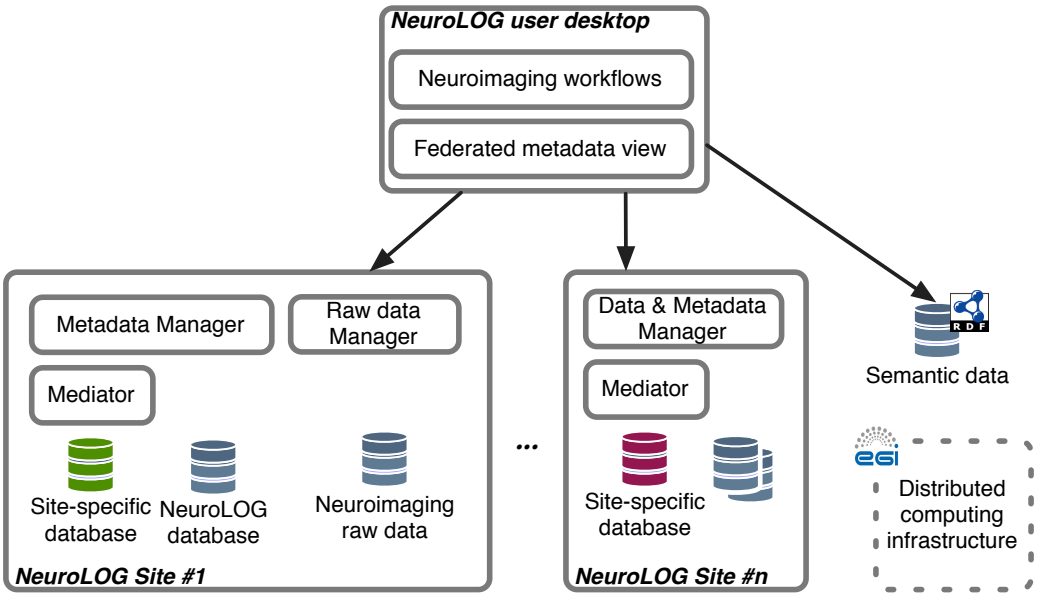

Figure 1.1: The NeuroLOG platform eases the setup of neuroimaging multi-centric studies through a semantic driven data federation.

NeuroLOG metadata manager: data sharing has been driven in NeuroLOG by a semantic approach. The NeuroLOG ontology has been developed to provide a uniform and strong conceptualization of the neuroimaging domain. This ontology has then been derived into a relational schema aimed at providing a federated view over the heterogeneous site-specific legacy databases. Each site thus exposes its (read-only) legacy database through a mediator and allows uniform querying over the distributed legacy databases. In addition on each site, a writable NeuroLOG database is deployed, natively using the federated schema. This database aims at storing the data produced by the neuroimaging workflows enacted through the platform.

NeuroLOG semantic data repository: to allow rich semantic querying, a semantic data store was periodically populated from the federated relational databases based on an SQL-to-RDF mapping tool.

NeuroLOG raw data manager: the raw data manager aims at retrieving and storing raw neuroimaging files, indexed either in the legacy database, or in the NeuroLOG database. Raw neuroimaging data are considered as sensitive and the raw data manager thus requires strong access control, taking into account local security policies.

NeuroLOG user desktop: the NeuroLOG client application mainly presents two facets. First, a user interface aims at navigating and searching for available distributed data, in the context of multi-centric studies. Then, once selected, data can be pushed as input data of neuroimaging workflows which are enacted over either local or global (EGI infrastructure) computing resources.

NeuroLOG challenges provide strong motivations for this thesis. NeuroLOG federates four neuroscience research centers and operates at a national scale (IFR49-Paris, GIN-Grenoble, IRISA-Rennes, INRIA-Sophia Antipolis). The heterogeneity and the potential volume of neuroimaging data correspond to the problematics of the data deluge considered in this work (RQ_1).

Moreover NeuroLOG integrates two levels of metadata representation, distributed relational data and centralized semantic data. This is confusing from an end-user perspective, due to different levels of query expressivity, and different distribution setups. Indeed, federated query processing has been implemented for relational databases in NeuroLOG but distributed querying technologies were not yet mature for handling distributed semantic data sources. In addition, NeuroLOG faces the issue of periodically synchronizing a centralized semantic data repository, based on distributed and autonomous relational data sources, which also presents a single point of failure. In this context, studying how to combine expressive querying and distributed semantic data stores seems particularly appealing (RQ_3).

Finally, NeuroLOG, as a collaborative e-Science platform, faces the challenge of easing the setup of multi-centric studies, while still guaranteeing the autonomy of participating data sources, at least in terms of access control. This security issue (RQ_2) is relevant in the context of this thesis since working with life-science resources differs from working with open datasets, due to legal or ethical concerns.

1.4.2 Multi-modal and multi-organ medical image simulation (VIP project)

The Virtual Imaging Platform (VIP) [Glatard et al., 2013] addresses multi-modal and multi-organ medical image simulation. A medical image simulation consists in combining, *i)* a description of a medical image acquisition device, specifying its physical characteristics and its parameters, *ii)* a description of the simulation scene, specifying the geometry and the spacial coordinates of both the device and the object for which

an acquisition should be performed, and *iii)* an object model, representing anatomical (possibly pathological) or physiological objects.

Medical image simulation is envisaged in VIP as multi-modal and multi-organ. Multi-modal, because the platform integrates several simulators and predefined simulation workflows for each modality (Computed Tomography, Magnetic Resonance, Positron Emission Tomography, and Ultrasound) ; and multi-organ, because the platform allows for using several anatomical or physiological models.

Motivations and challenges. This platform has been first motivated by medical image simulation needs. Simulating the production of medical images is crucial when designing and prototyping new medical imaging devices. Moreover it is possible to perform several acquisition protocols on a single device. Simulation is also crucial when prototyping new acquisition protocols.

Beyond device manufacturers, simulating medical images is also interesting for scientists involved in medical image processing. Synthetic medical images are a form of ground truth needed when evaluating or comparing image processing algorithms. In addition, these synthetic medical images can be used to adapt the parameters of the anatomical/pathological models, to finally obtain more realistic simulation results.

Finally, having a platform which eases the production of synthetic medical images based on several medical image modalities and several anatomical models has lot of educational interests. Indeed, such a platform can help students focusing on the physical, anatomical, pathological, or physiological parameters of a simulation rather than the technical specifies of several independent simulators.

Performing medical image simulation is challenging due to the following reasons. First, simulators are complex softwares with a steep learning curve (fine parameterization, requiring for a deep understanding of their physical principles) and hardly interoperable. Second, the organ models are also complex, possibly involving complex anatomical/pathological characteristics, movement or longitudinal follow-up. Finally, from a computing perspective, realistic simulation are heavy, in terms of calculation tasks, and thus require dedicated computing infrastructures.

Objectives. The VIP platform aims first at easing the access to medical image simulators through a simple web interface which integrates several simulators, several organ models, and leverage a dedicated distributed computing infrastructure required to handle multiple heavy simulation tasks. The second objective of the platform is to ease the sharing of both organ models, and simulators, following a semantic approach.

Platform architecture. Figure 1.2 illustrates the main components of the VIP platform which, based on organ models such as the thorax model on the left part of the figure, allows producing synthetic medical images of this model, such as the CT and PET virtual acquisitions on the right part of the figure.

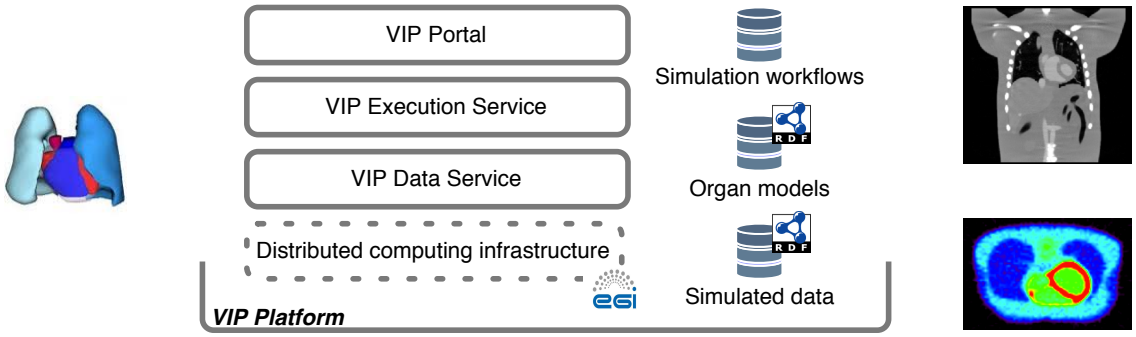

Figure 1.2: The VIP platform, easing the access to medical image simulators, organ models, and leveraging the EGI distributed computing infrastructure to handle heavy simulation.

VIP Portal: the portal component provides a user interface with a direct web access without requiring any software installation on the user workstation. Basically, it allows end-users to launch medical image simulations by selecting the appropriate simulator and specifying its parameters based on already launched simulations or new set of parameters. When launched, simulation tasks can be monitored through the portal. Due to their computing cost and their complex distribution over the EGI distributed computing infrastructure, it is important to provide information on the status of running simulations.

VIP Catalogs: the VIP platform is based on two main catalogs corresponding to the left and right parts of figure 1.2. The organ model catalog is a semantic repository which stores, for each model, the set of its source raw files, and a set of semantic annotations leveraging the VIP simulation ontology. It allows associating a clear and formal meaning to organ model entities, and performing advanced querying. Similarly, the simulated data catalog consists in a set of semantic annotations describing the nature of simulated data (their modality, their physical characteristics, etc.) based on the VIP ontology, and their provenance, in terms of input organ models and input parameters.

VIP Execution service: the execution service is based on the MOTEUR data-driven workflow engine which handles the parallelization and the relocating of computing tasks. Simulation workflows are executed on the VIP platform and the generated computing tasks are processed on dedicated computing resources (the EGI european grid infrastructure).

VIP Data service: the data service is responsible for achieving the data transfers between the VIP platform and the EGI storage elements. Simulation processes are computed on the EGI resources and require having data stored and accessible beforehand. At the end of a simulation, end-users want to retrieve their simulated data, and the data service is responsible for transferring back, on-demand, the resulting data through the portal.

The challenges of the VIP platform are particularly relevant in the context of this thesis. Indeed VIP is an e-Science platform in which simulation workflows can be considered as heavy *"in silico"* experiments. Medical image simulations are demanding in terms of computing resources, and VIP relies as a consequence on the EGI large-scale distributed computing infrastructure. Between november 2011 and november 2012, 1155 simulations were run, which corresponds to almost 400 CPU years, for 321 registered user originating from 38 countries.

VIP massively produces simulated data. It faces the issue of coherently sharing both the input organ models, the simulator themselves, the simulated data and their associated knowledge. VIP faces the issues of producing not only raw data, but also populating the simulated-data repository with meaningful data (RQ_4, RQ_5), bridging the gap between the technical simulation workflows and the VIP domain ontology.

1.5 Thesis outline

This report is organized in three parts. Part I addresses the coherent sharing of knowledge in the context of digital life-science platforms. Part II addresses the coherent production of knowledge resulting from the usage of e-Science platform. Finally, part III presents how the thesis contributions have been implemented and evaluated, mainly in terms of querying scalability and expressivity, and in terms of automated knowledge production in a concrete medical image analysis scenario.

The remainder of this document is organized as follows:

Chapter 2: We first delineate, valuable life-science resources in the context of collaborative e-Science platforms, and we introduce the main approaches and challenges of life-science data integration (RQ_1). One of the challenges consists in providing rich enough information sharing while still taking into account the specificities of life-science data sources. We then provide some background information on semantic data and analyze state-of-the art approaches targeting virtualized semantic data integration (RQ_4). We finally explore how data integration issues have been faced through concrete and significative life-science collaborative platforms.

Chapter 3: We focus in the security issues arising from the setup of life-science multi-centric studies (RQ_2). After introducing the security requirements for life-science collaborations, and analyzing the related works, we propose a security model which addresses mainly two partly conflicting objectives: *(i)* guaranteeing autonomous distributed life-science data sources with prevailing access control policies, and (ii) allowing for data sharing through coherent multi-centric studies. The implementation of the proposed security strategies is then described in the context of the NeuroLOG platform. This chapter is based on [Gaignard and Montagnat, 2009].

Chapter 4: We propose a set of strategies and algorithms dedicated to the efficient distributed query processing over multi-source semantic data (RQ_1, RQ_3), while still allowing for high expressivity with respect to the SPARQL 1.1 language features. Then we propose first experiments aimed at assessing the benefits of our approach (more deeply evaluated through chapter 8). This chapter is based on [Gaignard et al., 2012b], [Gaignard et al., 2012a], [Corby et al., 2012], and concludes the part I of this report, dedicated to knowledge sharing in collaborative life-science platforms.

Chapter 5: Towards automated knowledge production in e-Science platforms (RQ_5), we analyze first the state-of-the-art approaches aimed at bridging web services with domain ontologies, and by extension, at providing semantic scientific workflows. Then, we describe and discuss approaches based on provenance information resulting from workflow runs. We finally position our proposal for semantic e-Science experiment summaries with respect to semantic workflows and provenance-based approaches.

Chapter 6: We highlight the need for precise enough semantic service annotations, both in terms of the nature of parameters, but also in terms of their role. We introduce the need for roles modeling in service ontologies to disambiguate the meaning of consumed and produced data through processing services involved in workflows. Based on disambiguated semantic service annotations and fine-grained technical provenance information, we then propose to automatically produce, through inference rules, new meaningful semantic annotations, leveraging domain ontologies and workflow runs. This chapter is based on [Gaignard and Montagnat, 2010] and [Gaignard et al., 2011], and concludes the part II of this report, dedicated to automated knowledge production in collaborative life-science platforms.

Chapter 7: This chapter provides technical details on the implementation of our contributions. The NeuSemStore framework is first introduced as a mean to support semantic scientific workflows. Then, we technically described the distributed query processing extensions of the KGRAM framework which implements our approach for knowledge base federation.

Chapter 8: This chapter reports an experimental evaluation of the methods proposed and the software designed. It consists in three experiments exploring how the distributed query processing with KGRAM behaves at large scale, and in terms of heterogeneity and expressivity. We also explore through a real-life medical image analysis workflow, how our approach towards automated knowledge production allows better interpretation of workflows results.

Part I

Sharing distributed resources in life-Sciences

CHAPTER 2

Knowledge sharing in collaborative life-sciences: state of the art

Contents

2.1 Introduction

Collaborative e-Science platforms (also known as “scientific gateways”) are developed to ease the setup of scientific collaborations. These collaborations require exchanging scientific data and/or associated data analysis tools, in order to reduce the "time-to-discovery". In this thesis, we address data sharing issues, in the context of life-sciences, through the coherent integration of multi-source data (RQ_1). We both consider raw data, and associated domain-specific knowledge to achieve coherent integration of heterogeneous data sources.

This chapter delineates the contributions of this thesis with respect to the state-of-the-art approaches in semantic data integration. We first introduce specific resources underlying life-science collaborative platforms (section 2.1.1), general data representations and understanding layers (section 2.1.2), and the general data integration approaches (section 2.1.3). Then we provide in section 2.2 some background information on semantic data representation, querying, reasoning and persistency. The most relevant approaches for virtualized semantic data integration are described in section 2.3 and recent operational life-science platform are described in section 2.4 in terms of data and knowledge integration.

2.1.1 Life-science resources

In the context of translational biomedical research, collaborative e-Science platforms generally provide a wide range of resources, covering:

- *Data resources* correspond to the observational entities previously introduced and possibly associated information or knowledge. Life-science *data resources* are characterized by an important variability in the nature of data. Translational Life-science platforms may manipulate genomic data, micro and macro imaging, clinical case reports, thus facing structural and semantic heterogeneity. Moreover, several open databases may be contributing to the scientific activities conducted in the context of a given Life-science platform. Such platforms face distribution and integration issues since data querying should be coherently achieved over several data providers, and coherently integrated back into the Life-science platform.

- *Data processing resources* are application codes, possibly exposed remotely through *processing* services. They are responsible for data analysis or data transformations. In the context of e-Science platforms, data processing resources are generally pipelined through scientific workflows to achieve complex data processing. Scientific workflows and their constituting data processing units are more deeply introduced in chapter 5.

- *Coordinating resources* correspond to a set of metadata which coherently associate the raw data to the information needed to achieve a particular, possibly multi-centric, scientific or clinical study. These coordinating resources may involve information on pathologies or epidemiology, on available research or clinical raw datasets, information on open reference databases, information on the coordinator of the study and the participating researchers, etc.

- *Middleware resources* correspond to a set of software components hiding the underlying computing and storage infrastructure. Since the analysis and processing of life-science data, especially biomedical imaging data, requires compute- and storage-intensive capabilities, collaborative e-science platform generally rely on dedicated high-throughput clusters or open large-scale infrastructures such as

the European Grid Infrastructure (EGI). Collaborative e-Science platforms generally provide user interfaces to bridge the gap between e-Science activities and middleware operation, however, user knowledge on the underlying middleware may be required to understand and analyze possible e-science experiment failures.

- *Computing and storage resources* are a set of physical resources responsible for data storage and the computations resulting from data processing. Similarly, these resources should be hidden from the end-user perspective, but in some cases, investigating e-Science experiment failures may require technical knowledge on the underlying computing or storage infrastructure.

The contributions of this thesis mainly focus on *data* and *data processing* resources. Indeed we address in this work (i) the sharing of distributed knowledge associated to biomedical data and (ii) the production of knowledge through the usage of collaborative e-Science platforms, more precisely, through complex data processing achieved by the enactment of semantically enhanced scientific workflows.

2.1.2 Data representation and understanding

Several levels of data understanding are commonly represented through a pyramid grounded on *data* (observational entities). The *data* layer is topped with the *information* layer linking together *data* entities. The *information* layer is topped with the *knowledge* layer which organizes, or categorizes *information*. The *knowledge* layer is finally topped with *wisdom*, considered as a set of *knowledge* applied to a specific area [Ackoff, 1989].

Data results from observational entities. *Data* is generally qualified as *raw data*. It may represent sensor acquisitions, medical image acquisitions in the context of the biomedical research area, or neuro-psychological test results in the context of the cognitive-neuroscience community.

Information consists in linked data entities through established relations. In computer-based systems, *information* is generally materialized through descriptive *metadata* physically stored in files of relational databases. For instance some medical image formats include a file header (a text file companion). In the "Analyze" medical imaging format, the raw image file contains binary data and is suffixed with a `.img`, and a textual *metadata* file suffixed with `.hdr`. Metadata describes for instance, the spacial organization of each voxel, thus informing on how voxels are related to image slices, to patient orientation, finally allowing for navigation in 3D medical images and anatomical interpretation.

In the context of biomedical research studies, data are generally organized through databases to facilitate data search and scientific investigation. Relational databases offer a mean to represent *information* in the sense that data and relations are materialized through a set of tables, columns, and associations (through join operators and relation tables).

Knowledge represents the step further, in which *information* is coherently organized into categories. In the knowledge representation area, ontologies and controlled vocabularies play a key role in categorizing information. Indeed they generally result from a consensual community effort, and aim at being adopted by a large community to foster information understanding, comparison, exchange and reuse.

Knowledge representation systems have traditionally been implemented through Relational Database Management Systems (RDBMS) which allow to organize *information* (relational tuples) through typed columns inside tables. However RDBMS are not convenient to represent hierarchical organizations commonly used to understand several abstraction (generalization or specialization) layers, and they do not allow automated reasoning (i.e. classifying data) over possibly multiple categories.

Knowledge representation formalisms and languages such as Conceptual Graphs, OWL ontologies or RDF Schemas (RDFS) are dedicated to model the organization of information through categories (OWL or RDFS classes), and allow automated reasoning (entailments) through reasoners or inference engines. In these formalisms, *Knowledge* can be seen as a graph linking together semantic data entities trough semantic relations (properties). Data and relations are qualified as *semantic* since they are grounded to meaningful concepts from ontologies or controlled vocabularies. Recently, ontologies have largely been adopted in collaborative e-Science initiatives to both capture domain knowledge, and address interoperability when retrieving data from multiple and heterogeneous data repositories.

Wisdom can be defined as making use of *knowledge*. If we stay close to the information technology area, *wisdom* could be seen as the final, long-term objective of complex systems dedicated to decision support, or artificial intelligence.

With regards to these four layers, the contributions of this thesis, towards distributed knowledge sharing and production in collaborative e-science platforms, focus on both the *information* layer (the linked data entities) and the *knowledge* layer (the categorization of information).

2.1.3 Data integration: approaches and challenges

Data integration[1] consists in providing a unified, transparent, mean to work with multiple distributed data sources. Two main approaches exist to tackle data integration [Haase et al., 2010], namely, *materialized* data integration, also known as *data warehousing* or *Extract-Transform-Load* approaches, and *virtualized* data integration, also known as *federated querying* or *Distributed Query Processing* (DQP).

[1] "Information integration" may also be found in the literature to highlight the understandability of integrated data, but in this manuscript, we consider as synonyms "data integration" and "information integration".

2.1.3.1 Materialized or Virtualized data integration

Materialized data integration. This approach consists in populating a centralized repository from multiple distributed data sources, following an *Extract-Transform-Load* (ETL) approach. The *extract* phase consists in selecting, from the available data sources, the target data to be integrated. Then, the *transform* phase is critical and possibly complex. It consists in transforming the target data to obtain an homogeneous representation from the centralized data warehouse perspective. This step addresses structural and/or semantic heterogeneity between data sources and the data warehouse, and each transformation is specific to a data source. Finally, transformed data are aggregated and merged into the centralized data warehouse through the *Load* phase, possibly addressing redundancy and replication issues. Integrated data retrieval is achieved through centralized query processing against the populated data warehouse. Figure 2.1 illustrates the general principles for materialized data integration.

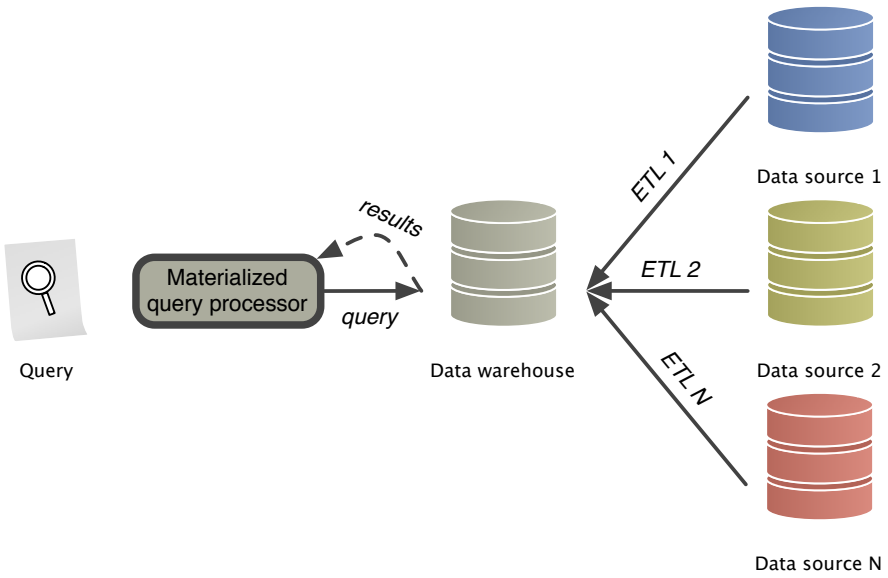

Figure 2.1: A sample materialized data integration where data from several distributed and heterogeneous data sources is extracted, transformed and loaded into a centralized data warehouse.

Virtualized data integration. This approach, also known as *data federation*, consists on the other hand, in (i) splitting the initial query into subqueries, (ii) then evaluating these subqueries against the multiple distributed data sources, and (iii) finally providing results combined together by the federated querying engine. In federated data integration approaches, data source heterogeneity is addressed through (i) query rewriting mechanisms, and (ii) data mediation to coherently fusion the results into a single query results set. Figure 2.2 illustrates the general principles for virtualized (or federated) data integration.

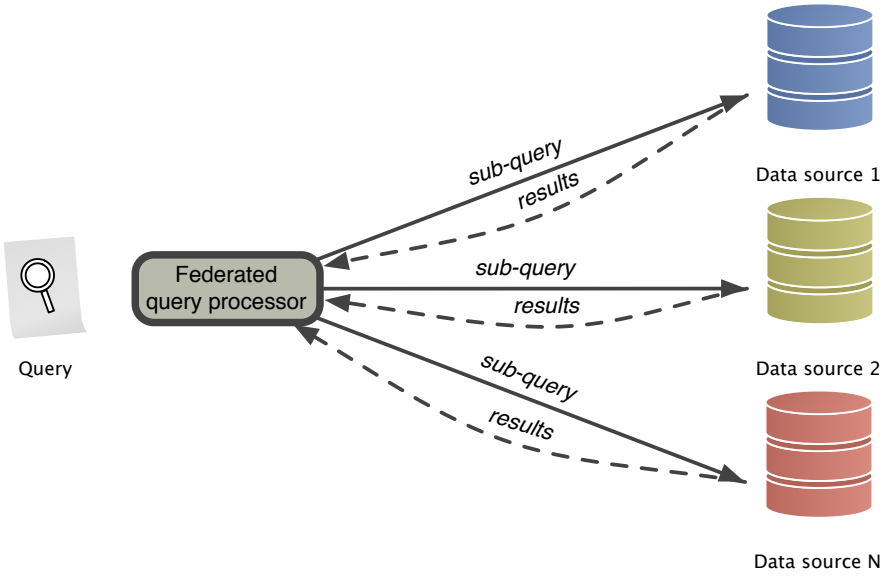

Figure 2.2: A sample federated data integration setup in which result data is dynamically retrieved from multiple distributed and heterogeneous data sources through query rewriting and distributed query evaluation.

To cope with the heterogeneity of multiple data sources, virtualized data integration systems provide a unified querying interface, based on a *global schema*. The *global schema* provides an homogeneous model over the organization of distributed data. Since data sources are generally autonomous in their day-to-day operation, they rely on *local schemas* to model the organization of their data. Inconsistencies between *local schemas* are tackled through mappings between *local schemas* and the *global schema*. Two main approaches are popular to address local schema heterogeneity in federated data integration, *Global-as-View* and *Local-as-View*.

Definition 1 *Global-As-View (GAV) approaches consist in representing the content of the global schema as a view over the data sources [Lenzerini, 2002].*

GAV approaches are considered as bottom-up approaches since the global schema is built from a set of data sources. The main drawback is that it is not adapted to dynamic federations when a new data source is added to the federation. Indeed, a new source added to the federation involves modifying the global schema. Moreover, a modification in a local schema may have an impact on the overall federation.

Definition 2 *Local-As-View (LAV) approaches consist in representing the content of each data source as a view over the global (unifying) schema [Lenzerini, 2002].*

LAV approaches are considered as top-down approaches since the global schema is fixed *a priori*, and data source participating into the federation describe the database they want to publish, based on the global schema. The main advantage is that the federation

may evolve without any modification on the global schema. Thus a new participating data source should have no impact on the other participating data sources. However, query rewriting mechanisms addressing global to local schema mapping are complex.

Haase and co-workers analysed the advantages and drawbacks of these two approaches [Haase et al., 2010]. Data warehousing might be inefficient in the context of a single user or application. Indeed, the complete distributed datasets must be transformed and loaded even if a query returns a small subset of data. Moreover, this approach is not suitable in the context of rapidly evolving remote data sources and requires for periodically going through the entire ETL process to update the content of the data warehouse. However, query processing against a single centralized data warehouse can be very efficient since no data communications are needed, and all content of data sources is available, thus allowing for optimized query processing.

But relocating the content of remote data sources into a single data warehouse may not always be feasible, due to the size and the multiplicity of data sources, or to autonomy constraints of data providers. This applies in particular to sensitive biomedical data. The main advantage of federated approaches is that data resides at data source and its complete loading is not needed anymore. Moreover, this approach is well suited for dynamic federation. Indeed data is retrieved at query evaluation time and there is no need to periodically synchronize a data warehouse. However the main drawback is the difficulty of achieving efficient Distributed Query Processing. This is due to (i) the physical distribution of data sources and thus the cost of communicating intermediate data needed to perform distributed joins and (ii) the possible heterogeneity of autonomous data sources that requires specific query rewriting mechanisms.

2.1.3.2 Life-science data integration challenges

Data integration challenges have been discussed in [Langegger, 2010] in terms of autonomy, distribution, and heterogeneity. These dimensions are not necessarily independent since when the level of autonomy of a data provider increases, the overall distribution and heterogeneity of the data integration system increases to. We propose to extend these dimensions with three other dimensions, particularly relevant in the context of collaborative life-science platforms, namely, *scalability*, *security* and *knowledge*.

Autonomy. Data providers participating in data integration initiatives, specially in the context of collaborative life-science platforms, generally operate legacy environments. To foster the adoption of collaborative e-science platforms and thus data sharing and reuse, the normal operation of the legacy environment should not be altered. Thus, data integration should be transparent, and not-invasive with respect to day-to-day legacy environment operations. Moreover, data providers may keep their autonomy in controlling the access over their hosted data. Access control is particularly relevant when dealing with life-science data sources. Resources providers must generally enforce strict security policies as soon as they host sensitive clinical or biomedical data. Moreover, valu-

able life-science data generally results from a long and costly process including population selection, data acquisition, data curation, data organization and publication through databases. Research activities being as much collaborative as competitive, researchers may be reluctant to share their material if they loose control over its management. It becomes thus challenging to provide data integration facilities while still guaranteeing data sources autonomy.

Heterogeneity. Heterogeneity is related to the autonomy property of data sources in the sense that when considering several data sources, they may all operate their own legacy environment. There is no reason for them to adopt homogeneous data representations or technical/logical organization strategies. Moreover, life-science is a broad, multi-disciplinary area involving a huge variability in the nature of observed and analyzed data (biological datasets including genomic data, micro- macro-imaging data from intra-cellular to organ scales, clinical case data, epidemiological data, etc.). Life-science data integration thus faces challenging syntactic and semantic heterogeneity to study scientific/clinical questions over this wide variety of data. Syntactic (or structural) heterogeneity refers to a same kind of data expressed through different structures or formats. For instance, patient information may be encoded through several formats. Semantic heterogeneity refers to a same kind of data described through several vocabularies, thesaurus, or semantic referents. For instance, two distinct clinical data sources may rely on different semantic referents such as the RadLex controlled vocabulary for radiology, or the general Systematized Nomenclature of Medicine – Clinical Terms (SNOMED-CT) thesaurus, thus preventing from unified querying through a unique vocabulary.

Distribution. Distribution is also closely linked to the autonomy challenge. It refers to geographically and thus physically distributed data sources, inter-connected through network communications. In the context of large-scale distribution setups (large distance between data sources, and large number of data sources), communications between data sources, especially in the case of federated data integration, highly depend on network bandwidth. Their cost need to be strongly considered. In that case, distributed information from multiple data sources need to be joined and the efficient (i.e. low-cost) distributed querying becomes challenging. Moreover, distribution raises security issues when communicating sensitive data, and fault-tolerance issues when for instance, an autonomous data source becomes unavailable.

Scalability. The scalability challenge directly derives from both the distribution of data sources, and a continuous increasing amount of data to be integrated. In addition, the web offers the opportunity to discover and interconnect world-wide data sources. In the context of collaborative life-science platforms, the data integration system must adapt to new data sources, not considered in the initial data integration plan, thus leading to an increasing amount of data to integrate. The data integration system must adapt (i) in terms of system operation (the data integration scenario could easily be augmented

with new data sources, and it should be transparent for participating data sources), and (ii) in terms of performances (end-user querying the data integration system should not be impacted by prohibitively low performances resulting from the increasing amount of data to be integrated and distribution overheads).

Security. The security challenge for data integration systems is also linked to the autonomy properties of data sources. For legal or ethical reasons, life-science data sources must keep the control over the sensitive clinical data they host. In the context of distributed data sources, data privacy (pseudonimization, encryption) must be enforced. A fine-grained distributed access-control must also be guaranteed, while still being able to log sensitive data access through journalisation. Distributed access control is particularly challenging since data sources must remain autonomous in complying with prevailing local security policies, while still allowing for collaborations in the context of multi-centric studies.

Knowledge. The knowledge challenge for data integration consists in providing a mean, through knowledge representation/management techniques, to (i) address semantic interoperability between autonomous data sources, thus fostering data reuse, and (ii) to provide high expressivity and automated reasoning for data retrieval. Ontologies have gained a lot of interest in the life-science area. They are generally used to (i) build a semantic referent delineating community knowledge for a specific field, (ii) to drive the setup of data integration system through global schemas derived from ontologies, and more recently (iii) to query heterogeneous and distributed multiple data sources through Semantic Web standards.

Towards collaborative life-science platforms, materialized data integration becomes disqualified as soon as we consider the autonomy property of data sources as a first-order requirement. Although virtualized data integration (or federated data integration) lead to costly distributed query evaluations, and possibly complex query rewriting mechanisms, they are appropriate to address the main challenges faced when providing life-science data integration.

Since ontologies, and more generally knowledge engineering methods and techniques have recently gained a huge interest in life-science areas, they open promising perspectives towards a better understanding of shared scientific data (and processing tools) and a better support to address interoperability issues faced by autonomous distributed scientific data sources. Section 2.2 briefly provides some background information on semantic data representation, on their querying, on the possible reasoning capabilities offered by semantic engines, and on their storage.

2.2 Background informations on semantic data: representation, querying, reasoning and persistency

2.2.1 Semantic data representation

2.2.1.1 The Semantic Web as a foundational technological layer

Representing the meaning of data is a need that has recently emerged from the Semantic Web area, which intends to bring meaning to the unprecedented and tremendous amount of data published over the Web.

Tim Berners-Lee presents the *Semantic Web* [Berners-Lee et al., 2001] as *"not a separate Web but an extension of the current one, in which information is given well-defined meaning, better enabling computers and people to work in cooperation"*. He also conditions its success to automated reasoning: *"computers must have access to structured collections of information and sets of inference rules that they can use to conduct automated reasoning"*. The Semantic Web is based on *Ontologies* - languages to formally describe domain specific concepts and their relations - coupled to reasoning engines, to perform domain fact deductions.

Representing the semantics of data generally consists in associating meaningful and structured metadata (*i.e.* descriptive information) to raw data. Such metadata (i) classifies raw data through well-defined concepts and (ii) link raw data together through well-defined relationships. The main objective is that metadata can be automatically processed in order, for instance, to help users in retrieving relevant information from a jungle of resources, typically data or services, spread over the Internet. But some technical means are needed to address (i) the unambiguous and unique identification of resources over the Internet, (ii) the description and linking of these resources, and (iii) the conceptualization of domains to formally express the meaning of resources (*i.e.* the setup of an ontology).

A lot of standardization efforts have been made through the W3C (or the Internet Engineering Task Force (IETF)) that led to the definition of three frameworks to address the above mentioned needs, namely URI[2] (Universal Resource Identifier), RDF (Resource Description Framework), and OWL (Web Ontology Language) which constitute the main components of the Semantic Web, and by extension provide a strong support to the Knowledge Engineering area.

2.2.1.2 Linking data

In order to share information, URIs are the *de facto* standard to unambiguously identify information. No particular assumption is made on the resources identified by a URI. They might be abstract or concrete, and are potentially not accessible over the Internet. Standard URLs are specific URIs used to locate resources over a network, and thus URLs syntax can be used to define URIs. For instance "Saint-Malo" is an ambiguous name for a city and might refer to either a french city or a canadian city. We are able to disam-

[2] more precisely, originating from the IETF RFC 3986

biguately identify "Saint-Malo" with two URIs such as "http://canada/quebec/saint-malo" and "http://france/bretagne/saint-malo".

As soon as resources are well identified through URIs, RDF can be used to attach descriptions to these resources and link them together. The main idea of RDF is to describe the semantics of the data by expressing simple statements of the form *Subject – Predicate – Object* which can also be considered as simple sentences involving a subject, a verb, and a complement. Note that when several statements are grouped together, the objects of some statements may also act as a subject of other statements which lead to a graph representation. Indeed, a set of RDF statements can be considered as a directed labelled graph where *Subjects* and *Objects* define nodes and *Predicates* define labelled directed edges.

Let us consider an example from the neuro-imaging domain where a medical image has been acquired from a scanner and is processed to extract brain tissues.

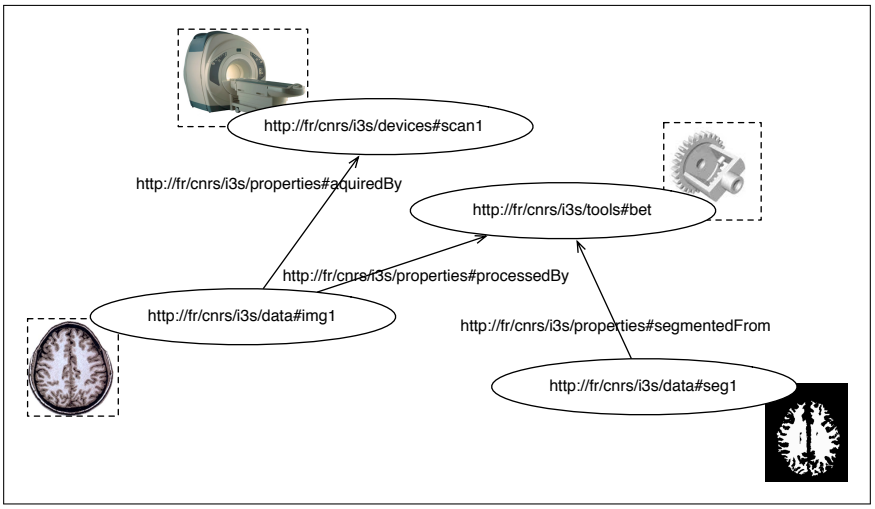

Figure 2.3: Linking raw and processed data to the acquisition equipment and processing tool

Figure 2.3 represents an RDF graph linking neuro-imaging resources identified by their URIs. It first states that a medical image has been acquired from a scanner, then, that this image has been processed by a brain extraction tool, and finally, that brain tissues have been extracted from the "Bet" brain extraction tool.

2.2.1.3 Classifying data

So far, we have seen how RDF can be used to attach descriptive information to data and to establish relationships between data. But another aim of RDF is also to allow user communities to define controlled vocabularies (or terminologies) by describing the domain specific concepts and how they are related to each others. RDFS, which stands for "RDF Schema", is an extension of RDF[3] that provides a language to define such vocabularies.

[3]as "XML Schema" extends XML to support the definition of complex types

Two main constructs are provided that aim at describing *Classes* and *Properties*.

Classes are used to group resources together. Resources that belong to a particular class are called *instances* or *individuals* of that class. Hierarchies of classes can be constructed through the definition of sub-classes.

Properties are used to define relationships between classes, and in the same way, it is also possible to define hierarchies of properties through sub-properties. Properties are characterized by (i) their *domain* – the set of classes from which the property can be attached, in other world its applicability domain – and by (ii) their *range* – the set of classes to which the property can be attached.

The following example shows a simple controlled vocabulary covering the previous example (figure 2.3).

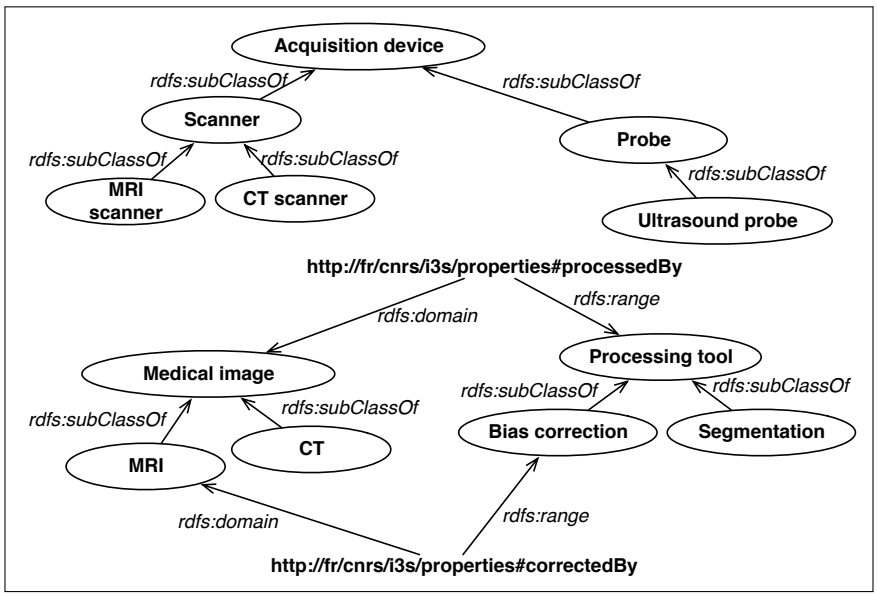

Figure 2.4: A sample controlled vocabulary expressed in RDFS

Figure 2.4 illustrates how RDFS can be used to define the concepts of a domain and their relationships. The rounded boxes represent the domain classes. Note thawt they are also identified through URIs but for readability, we only show their label. Arrows labeled *rdfs:subClassOf* constitute the hierarchies of concepts. Arrows labeled *rdfs:domain* and *rdfs:range* define domain-specific properties. For instance *processedBy* and *correctedBy* differ by their *rdfs:domain* and *rdfs:range*. More precisely *correctedBy* can only be applied to MRI images and bias correction tools, whereas *processedBy* is applicable for any medical image and any processing tool. In addition we can state in RDFS that *correctedBy* is a sub property of *processedBy*. In the remainder of the document we will refer to both RDF or RDFS standards by the RDF(S) notation.

However, for some user communities, these languages may not be sufficient to precisely model their domain, and they usually develop a more complex domain ontology. An ontology can be defined as a set of *class axioms*, *property axioms*, and *facts about individuals* (or *instances*). While *axioms* aim at formally defining the semantics of classes or properties, *facts* are assertions (or statements) describing the instances. Note that *axioms* are used to place constraints on classes and on their relationships (properties). The OWL language [W3C OWL Working Group, 2012], is a standard framework to build ontologies. OWL ontologies are written and stored in RDF (or XML) syntax so that they really appear as an extension of the existing RDF(S) stack. OWL brings new constructs, through *axioms* and allows richer exploitation through the possible entailments operationalized through reasoning engines. This document does not detail the whole extensions, with regards to the RDF(S) language, but rather illustrates some of them through the previous examples.

- *TransitiveProperty*: the "derivedFrom" property could be stated as a transitive property so that it can be automatically inferred that if A "derivedFrom" B and B "derivedFrom" C then A "derivedFrom" C.
- *InverseOf*: we could also imagine a *consume* property starting from the processing tool class and directed towards the medical image class, which is the inverse of the *processedBy* property. This would be useful, if we consider a set of instances related by only one of these two properties, to generate the inverse property with an inference engine, and thus to complete the set of relationships.
- *disjointWith*: this construction helps in validating the nature of a set of instances. For instance we could state that MRI and CT are disjoint classes so that when validating a set of instances, the system can check that none of the instances belong to both MRI and CT classes (similar to an exclusive disjunction, also known as the XOR logical operator).
- *unionOf*: OWL classes, generally stated as "complex classes", can be defined through axioms representing basic set operators such as union, intersection or complement. We can easily imagine that the *Acquisition device* class can be defined as the union of the two classes *Scanner* and *Probe*, so that an OWL reasoner infer that all instances of *Scanner* and *Probe* classes also belong to the *Acquisition device* class.
- *cardinality*: another interesting feature is the ability to define some constraints on the number of possible individuals linked together through properties. We could imagine that a medical image (*i.e.* an instance of the *Medical image* class) cannot be acquired from two different devices. This restriction can be stated by the *maxCardinality* set to 1 for the *acquiredBy* property.

OWL constructs allow richer description of domain-specific knowledge but the counterpart is that possible deductions through reasoners become more complex and time consuming. More insights will be given on the possible entailments and the impact on their computation in sections 2.2.3.1 and 2.2.3.3.

We have seen that the two standardized languages of the Semantic Web, RDF(S) and OWL, are the cornerstones in the process of building semantic applications either for storing knowledge about the domain-specific concepts or about the data itself. OWL being expressed through the existing RDF stack of the W3C, it seems thus natural to lean on RDF tools to express semantic data, either considering simple conceptualizations through RDF(S) or complex conceptualizations through OWL ontologies.

2.2.2 Semantic data querying

Following the widely adopted Semantic Web stack from the W3C, semantic data is usually represented as RDF triples. RDF data being possibly serialized through an XML syntax, it could thus be possible to use standard XML query languages to retrieve information. However, XML query languages are not suitable to query graph structures. This fact motivates the design of specific query languages adapted to the graph-based representation of RDF data. RDF query languages generally adopt an SQL-like syntax to retrieve data.

The SPARQL Protocol and RDF Query Language (SPARQL 1.0) has been standardized through a W3C recommendation [Prud'hommeaux and Seaborne, 2008]. It is the most popular RDF query language. The queries are composed of two clauses. The first one specifies the kind of the query. The second clause, the WHERE clause, consists basically in specifying a graph pattern through a set of triple patterns in which variables match target data. Queries may include conjunctions, disjunctions, or optionality. Four kinds of queries are available through the SPARQL 1.0 language:

- SELECT: returns all or a subset of the values of variables bound in the query pattern match (the WHERE clause).
- CONSTRUCT: returns RDF triples (defined in the CONSTRUCT clause) corresponding to the bound variables of the query pattern (the WHERE clause).
- ASK: returns *true* or *false* with respect to the matching of the query pattern.
- DESCRIBE: returns an RDF graph describing resources found through the graph pattern matching.

Listing 2.1: sample SPARQL query selecting all medical images acquired by a Scanner device

```
1 PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
2 PREFIX myOntology: <http://fr/cnrs/i3s/myontology#>
3
4 SELECT ?image WHERE {
5         ?image myOntology:acquiredBy ?device .
6         ?device rdf:type myOntology:Scanner .
7 }
```

Listing 2.1 illustrates a sample SPARQL query aimed at retrieving the URIs of all medical images acquired by a scanner acquisition device. Variables are prefixed by a question mark. The first triple pattern searches for *acquiredBy* properties. For each of them, the engine searches for all devices of type *Scanner*. Finally the source of the remaining *acquiredBy* property are returned as resources matching the whole WHERE clause.

2.2.3 Semantic reasoning

We have seen in the two previous sections how semantic data can be represented (OWL and RDF(S) languages) and queried, through graph pattern matching (SPARQL language). But semantic data representation provides even more valuable benefits through "reasoning", which consists in performing automated logical deductions based on data representation models and associated rules.

2.2.3.1 Description Logics

Description Logics (DL) [Nardi and Brachman, 2003, Baader and Nutt, 2003] are used to provide a logical formalism for Ontologies and Semantic Web by modeling *concepts*, *roles*, *individuals* and their *relationships*. More generally, Description Logics are used as a foundation for building knowledge bases from two main components, the terminological knowledge box, the *TBox* and the assertional knowledge box, the *ABox*. The *TBox* refers to statements regarding the terminology itself. It describes the hierarchy of concepts, and, the relationships between concepts, whereas the *ABox* refers to statements regarding the individuals, their relationships, and how they are associated to concepts of the *TBox*. In description logics, together *TBox* and *ABox* constitute a knowledge base.

What makes DL sound "exotic" for classical computer system users or programmers, is that DL are based on "open world" assumptions. Classical systems are based on "closed world" assumptions which means that if a fact is not known to be true, it is considered as *false*. On the contrary, "open world" assumptions mean that, *ABox* being considered as incomplete, if a fact is not known to be true (in the *ABox*), it does not mean that the fact is *false*, and it becomes potentially *true*. It becomes then much more complex to determine if something is *true*.

The Web ontology language OWL [Horrocks et al., 2003, Grau et al., 2008] is based on Description Logics. Indeed, OWL *axioms* can be considered as statements of the TBox while OWL *facts* can be considered as statements of the ABox. Finally, an OWL ontology, encompassing axioms and facts, can be bound to a DL Knowledge Base.

Being much more expressive than RDF(S) languages and allowing for richer inferences, OWL lead to computational issues such as decidability (related to the "open world" assumption). Three variant of OWL are thus proposed to tackle these computational issues: (i) *OWL-DL* which has a readable DL-like syntax allows for decidable inferences, (ii) *OWL-Lite* a subset of *OWL-DL* with an even more simpler syntax with tractable inferences, and finally (iii) *OWL-Full*, a syntactic and semantic extension of

RDF(S) without any usage restriction. Regarding the computational complexity of these variants, *OWL-Full* is undecidable, *OWL-DL* has exponential time complexity, and finally *OWL-Lite* is more tractable, with a deterministic and exponential time complexity. *OWL* evolved recently in *OWL2* [W3C OWL Working Group, 2012]. This version introduces a new non-RDF syntax and offers new expressivity. In addition, *OWL2* introduces 3 profiles dedicated to tractable reasoning, with different limitations on expressivity: *OWL2-EL* is well suited for large ontologies, *OWL2-QL* is focused on tight integration with relational databases and is suitable for large set of individuals, and finally *OWL2-RL* is focused on tight integration with rule engines.

2.2.3.2 Conceptual graphs

Conceptual graphs are a formalism introduced by John Sowa [Sowa, 1984] to capture the underlying meaning of data, to link them together and to identify patterns between them [Polovina, 2007]. Conceptual graphs (CGs) are appealing because their formalism is logically precise, their representation is human-readable, and finally they are computationally tractable. RDF graphs can easily be mapped to Conceptual Graphs since they share a similar principle: concept nodes are connected to relation nodes and form together a labelled graph.

Projection is the key operation providing reasoning capabilities in CG. It consists in calculating a specialization/generalization relation between two graphs. It is generally stated that a "general" graphs "projects" into a more "specialized" graph. The projection operation lead to the identification of patterns in data which is particularly suited in the context of information retrieval.

2.2.3.3 Reasoning tasks

As presented in section 2.2.1.1, one of the objectives of the Semantic Web (and by extension, the Knowledge Engineering area) is to lean on reasoning engines to exploit semantically annotated data and therefore deduce new knowledge specific to a domain covered by the underlying domain ontology. Based on different constructs and level of expressivity, the languages of the Semantic Web allow several kinds of deductions with different complexities and computation costs.

Inherited from Description Logics, reasoning tasks target the meaningful exploitation of knowledge bases by means of validation or deduction of new facts. Reasoning tasks are generally split between the *TBox* and the *ABox* as some are related to inferences or validation at the conceptual level, while other ones are more related to instances validation and classification. Some of the main reasoning tasks are briefly introduced below and illustrated through the *TBox* graphically defined in figure 2.4.

Several reasoning tasks may be achieved over the *TBox* of a knowledge base:

- *Subsumption* is a key reasoning task which consists in determining if a concept is more general than another one. For instance, the class "Acquisition device" *subsumes* the class "Scanner" as a scanner is a specialization of an acquisition device.

- *Classification* consists in determining the appropriate place of a newly defined concept into a hierarchy of concepts linked together by subsumption relations. For instance if we define a "Medical device" as the union of "Probe", "Scanner", this class should appear as a sub-class of "Acquisition device" and as a super-class of both "Probe" and "Scanner".
- *Logical implication* consists in checking if relationships between concepts are logical consequences of the assertions of the *TBox*.
- *Equivalence* consists in testing if two classes or properties, potentially defined by different set of assertions or axioms, are equivalent. Note that the OWL *sameAs* construct is used to state that entities are strictly the same.
- *Satisfiability* is a key reasoning task with the following underlying idea. A newly added concept in the *TBox* must make sense with regards to the *TBox* and does not lead to any contradictory knowledge. The logical foundation of satisfiability is that an interpretation (the set of deductible facts) of the concept must satisfy all the axioms (all the assertions) of the *TBox*.

Reasoning tasks regarding the *ABox* of a knowledge base:

- *Instance checking* is the main reasoning task over the *ABox* (and also the *TBox*, by extension). It validates that an individual belongs to a specified concept. *Instance checking* is the foundation for all other reasoning tasks.
- *Knowledge base consistency* consists in verifying that each concept of the *TBox* admits at least one individual (different from).
- *Realization* consists in, for a given individual, finding the most specific concept it belongs to. For instance, given an instance of the "Medical image" class, it may not be useful to qualify it with an intermediate concept in the hierarchy of medical image classes, such as MR image. It is much more useful to qualify it with the most precise concept such as its proper modality, "T1-weighted-MRI" for instance, taking into account that all abstract super-classes are also considered when asserting that a medical image belongs to the class "T1-weighted-MRI".
- *Retrieval*: which finds all the individuals belonging to a given concept. This reasoning task is particularly important when searching information into a knowledge base. We can easily imagine some basic queries aiming at finding all T1-weighted MRIs registered into the knowledge base.
- *Satisfiability*: the underlying idea is similar to satisfiability at the concept definition level (*TBox*) in the sense that a newly created instance should not lead to any contradiction concerning other instances or concerning the concepts of the *TBox*. For instance if we consider the *acquiredBy* property of figure 2.4 with its cardinality constraint (maximum one device), linking an instance of a medical image with two acquisition devices would lead to an unsatisfiability issue.

Unlike OWL, RDF(S) does not define any Description Logics, but the semantics specification of RDF(S) [Hayes and McBride, 2004] defines a full set of valid inferences with regards to an RDF graph or its underlying RDF Schema. Thirteen rules define the semantics of RDFS and they can be applied to infer new RDF triples through the inferred closure[4] of the source graph.

2.2.3.4 Rule-based reasoning

The use of rule engines, also known as inference engines is a general data-driven and declarative paradigm to deduce new conclusions from a set of models, data and inference rules. An inference rule is made of two parts, the *antecedent* and the *consequent*, and is generally expressed through an *If* clause and a *Then* clause. Inference engines are the core of reasoners, since inference rules are internally used to perform reasoning tasks such as *subsumption*, *classification*, *instance checking*, etc.

Forward chaining is one of the two main strategies to perform reasoning based on inference rules. Forward chaining consists in starting from the whole available data and applying iteratively the inference rules. The produced data is then merged into the original set of data and the set of available rules is again executed to produce additional data. This process continues until a termination goal, or saturation is reached.

Unlike forward chaining which iterates over the *antecedent* of the rules to reach one of the goals (the *consequent*), backward chaining starts from the goals, and search for a rule which has an *antecedent* known to be true. If it is not the case, this *antecedent* is added to the list of goals and the engine goes for a new iteration. Being goal-driven, the backward chaining strategy is well adapted to prove facts. On the other hand, forward chaining is more considered as a data-driven approach allowing the inference engine to produce new knowledge, based on the existing data set.

SWRL [Horrocks et al., 2004] is a Semantic Web Rule Language combining OWL with RuleML[5], the Rule Markup Language. SWRL rules represent implications between the *antecedent* and the *consequent* clauses. Each clause is made of a set of atoms interpreted as a conjunction. Each atom may represent (i) the belonging of a variable to an OWL class, (ii) the existence of an OWL property between two variables, (iii) similarity or difference between two variables. A variable represents either an OWL individual or an OWL data value. More recently, the W3C proposed the Rule Interchange Format (RIF) [Kifer and Boley, 2010] aimed at addressing interoperability of rule languages and engines.

We have seen that forward chaining engines iteratively produce new facts into the knowledge base through the computation of its inferred closure. But in order to be queried or retrieved, the results of this inferred closure must be made available. Most of the time, semantic stores provide such a capability generally known as *materialization*.

[4] which consists in iteratively saturating the graph with all possible inferred statements
[5] http://ruleml.org

With the assumption that adding new facts to the knowledge base never implies that existing facts are retracted (also known as *monotonic* entailments) some tools provide a *total materialization* strategy which consists in computing the inferred closure after each modification of the knowledge base. The main benefits are the consistency of the knowledge base and efficiency of queries (no reasoning needed at query/retrieval time) but on the other hand, addition, updates or deletions of new facts are costly in terms of computing resources.

2.2.4 Semantic data persistency

RDF being the *de facto* standard for expressing semantic data, several possibilities can be envisaged to permanently store RDF triples. RDF stores are generally built on top of (i) file storage, or (ii) relational database management systems (RDBMS), or (iii) native graph model storage. Via their proper API, RDF stores generally propose in-memory RDF statement management, but since it is a non-persistent storage, this capability will not be addressed in this section.

File-system back-ends might be useful for development purpose because of an easy deployment and the simplicity of storing a small set of RDF statements into a unique file. But this approach requires to manually manage RDF files and would not be suitable in the context of concurrent access or large set of files. RDF stores usually operate a native storage back-end with optimized storage and loading procedures.

RDBMS back-ends appear to be a convenient and scalable approach with regards to the continuous growing amount of produced RDF triples. Indeed, RDF stores can be independent from underlying RDBMS and may benefit from replication, clustering, or simple access control mechanisms. But RDF stores become also impacted by the different performances of these RDBMS. The counterpart of RDBMS back-ends appears when considering reasoning tasks, especially using rule-based engines. These reasoning engines dynamically generate database accesses to propagate the effects of inference rules which dramatically impact the performances of the reasoning engine.

With native storage back-ends, RDF stores are able to implement optimizations strategies to more efficiently store or retrieve RDF triples. Some of them natively adopt a graph structure which better fits the way of representing RDF statements. RDF storage is achieved through dedicated and optimized data structures thus increasing the overall performances.

2.2.4.1 Existing stores

The following section briefly describes existing semantic data stores and focus on their underlying physical storage, their support for reasoning and their performance.

Jena framework [McBride, 2002, Company, 2009] is a widely adopted framework covering most of the concerns related to the development of Semantic Web applications. Jena provides a unified programming environment to address the management of RDF(S) and

OWL ontologies. Moreover, it provides some interesting features regarding the persistence of semantic data and the reasoning through its inference engine, or external state-of-the-art engines.

Jena provides two subsystems to persist RDF and OWL data, namely SDB and TDB. While TDB focuses on high-performance through a native indexing and storage engine in a local filesystem, SDB is built on top of a classical relational database and benefits from non-functional properties provided by the database engine such as authentication, logging, clustering, etc. These two components do not let the user specify or tune a schema for storing semantic data but use their own schemas to efficiently perform storage or retrieval. Different levels of caching are used to transparently let the user manage its Jena model without taking care of the persistency layer.

Jena can enable or disable reasoning capabilities. For each category of reasoning, a dedicated ontology model is available. These ontology models cover ontology languages such as OWL-{Lite, DL, Full}, RDFS, or DAML+OIL[6], with different kinds of reasoners (transitive class-hierarchy inferences, or rule-based reasoners). Based on these reasoning capabilities, Jena distinguishes between the asserted model (or base model, *i.e.* without inferences) and the inferred model, mainly because depending on the application, it is not always useful to store (materialize) inferred statements[7].

Some limitations of the Jena SDB persistence engine have been pointed out in [Kim et al., 2008b].

Joseki is a web server aimed at accessing semantic data through the SPARQL RDF query language [Prud'hommeaux and Seaborne, 2008] and the SPARQL protocol [Grant Clark et al., 2008]. The SPARQL protocol consists in defining a standard web service interface for SPARQL query processors. Joseki implements both SPARQL/Query and SPARQL/Update [Seaborne et al., 2008] protocols and thus provides a web service interface to query and update RDF graphs. RDF data are internally managed through the Jena framework.

Owlgres [Stocker and Smith, 2008] is a Description Logics reasoner (DL-Lite family [Calvanese et al., 2007] of Description Logics, with its corresponding OWL2 QL Profile recommended by the W3C [Motik et al., 2009]), with a relational database backend. This engine does not show recent development activities (first and last release in May 2008). The engine is released with a simple Joseki endpoint and some scripts to load ontologies into the relational PostgreSQL database. This semantic repository has some limitations regarding its reasoning capabilities (no transitive properties, no cardinality restrictions, no individual equality assertions) but performs queries with logarithmic time with respect to the size of the data.

[6] the OWL predecessor
[7] which could be considered as volatile statements

Virtuoso is a commercial[8], general-purpose, data and application container, with extensive SPARQL and RDF support [Erling and Mikhailov, 2007]. It additionally provides a Jena interface. Virtuoso addresses scalability for both semantic data storage and reasoning tasks. In its commercial distribution, Virtuoso also addresses information distribution and heterogeneity issues through the setup of a unified data model on top of its Virtual Database Engine.

With regards to the inference scalability [Orri and Ivan, 2009], whereas common approaches consists in beforehand materializing deduced facts, Virtuoso performs as much as possible runtime and on-demand inferencing. To guarantee responses in a fixed time, incomplete results may be returned through partial evaluation of SPARQL queries. Virtuoso also provides a realtime implementation of the *owl:sameAs* property, and supports transitivity through the rewriting of transitive SPARQL subqueries. While one subquery starts from the source, the second subquery starts from the target, and the first intersection of paths produces a first partial result.

Allegrograph is a commercial product providing a high-performance disk-based storage with standard HTTP and SPARQL interfaces. Multiple RDF triple stores, can be assembled into a virtual single store, thus enabling the federation of distributed stores. Allegrograph provides some limited but effective and practical reasoning through its RDFS++ engine. RDFS++ supports the following predicates: *rdf:type*, *rdfs:subClassOf*, *rdfs:range*, *rdfs:domain*, *rdfs:subpropertyof*, *owl:sameAs*, *owl:inverseOf*, *owl:TransitiveProperty*, and *owl:hasValue*. This engine can be topped with a Prolog interface allowing to perform rule-based reasoning. The Prolog engine can also be used on top of RDF data to allow more complex deductions based on domain-specific reasoning.

Sesame Sesame [Broekstra et al., 2002] is an open-source, community-supported framework to store and query RDF(S) data. It can be connected to several storage mechanisms such as standard file systems or relational databases through its SAIL API (Storage and Inference Layer). The repository can be queried through the standard SPARQL query language and provides reasoning capabilities through the support of an RDFS inferencer (also provided by SAIL). Sesame additionally provides a standard web service interface which implements the SPARQL protocol. Sesame is extensible through a plugin architecture.

In the context of the Sesame engine development, a specific query language SeRQL (Sesame RDF Query Langugage) has been proposed. According to its authors, some original features has been proposed such as the support for graph transformation, basic set operators (*union*, *intersection*, and *minus*). SPARQL and SeRQL have been developed during simultaneous initiative but merge efforts have been achieved from both sides to benefits, in SPARQL, from features of SeRQL and conversely.

Alibaba is an extension to the Sesame RDF repository which allows to bind Java objects and classes to RDF triples and RDFS/OWL classes. Alibaba also exposes them

[8] and open-source distribution

through an HTTP server. A federation feature is additionnaly provided with the set up of virtual RDF stores based on multiple distributed RDF data sources. Based on the SPARQL Protocol, these federations are therefore read-only.

OWLIM [Kiryakov et al., 2005] is a high-performance semantic repository published as an extension of Sesame through a SAIL implementation. Two versions are available, SwiftOWLIM, the free edition and BigOWLIM, the "enterprise" edition (usable for free in a research context). Both editions provide a persistence strategy guaranteeing data preservation and consistency, and support for reasoning with OWL dialects, covering most of OWL-Horst[9], OWL-Lite, RDFS, and OWL2 RL[10]. SwiftOWLIM is dedicated to prototyping since it loads the full content of the knowledge base in central memory and performs fast reasoning and query evaluation. On the other hand, BigOWLIM scales over billions of triples, performs query and reasoning optimizations, but still, at the cost of slower performances. OWLIM products have recently been renamed [11]. OWLIM-Lite corresponds to a fast in-memory semantic repository, OWLIM-Lite SE provides better scalability and robustness, and finally OWLIM-Enterprise provides a semantic repository possibly deployed on a replication cluster and allows for load balancing and scalable query performance.

Tupelo [Futrelle et al., 2009] is a middleware supporting large-scale e-Science through a decentralized semantic data repository. Tupelo has been driven by general principles to enhance metadata interoperability: (i) when data or metadata are re-located, they should retain their meaning (underlying model/ontology), (ii) metadata should be automatically interpreted in order to scale , and (iii) the knowledge on how data has been processed might be more valuable than the data itself.

To implement these principles, Tupelo leans on semantic web technologies for representing metadata, on content management system technics to handle data and on the open provenance model [Moreau et al., 2009] to record complex processes and data provenance. Tupelo offers two categories of operations, covering both metadata management – through the assertion, the deletion, and the search of statements – and data management allowing users to read, write and delete large binary objects (BLOBs) globally identified by their URIs. Some interfaces are provided to connect to RDF triple stores (Jena, Sesame), local or remote file systems (SSH), or HTTP servers.

Distribution issues are handled through the ability of globally identifying resources across several data and metadata providers. Semantic data can be linked across repositories through different identification policies and mappings. Moreover computational inferences are available through the extension of the middleware with dedicated plugins, similarly to the Jena framework.

[9] a decidable rule extension to OWL
[10] an OWL2 profile dedicated to scalable reasoning (polynomial time) through a rule engine
[11] http://www.ontotext.com/owlim

	Storage			*Reasoning*				
Repositories	*In-memory*	*Relational DB*	*Native*	*RDFS*	*Desc. Logics*	*Rule-based*	*Materialization*	*Performance*
Jena	✓	✓	✓	✓	≈OWL-Lite	✓	dynamic	⊖
OWLGres	-	✓	-	✓	no trans., no card.	-	?	⊕
Virtuoso	✓	✓	✓	subsumption	same/equivalent		dynamic	⊕⊕
AllegroGraph	-	-	✓	RDFS++	< OWL-Lite	✓	dynamic	unavailable
Sesame	✓	✓	✓	✓	-	-	-	⊕
SwiftOWLIM	✓	✓	-	✓	≈OWL-Lite	✓	static	⊕
BigOWLIM	-	✓	-	✓	≈OWL-Lite	✓	static	⊖

Table 2.1: Comparison of several existing stores regarding their storage back-end and reasoning capabilities

2.2.4.2 Stores comparison

Rohloff and co-workers [Rohloff et al., 2007] and Bizer & Shultz [Bizer and Schultz, 2009] evaluated the most popular RDF Stores in terms of capabilities and performances over large set of semantic data. The LUBM [Guo et al., 2005] methodology and datasets is dedicated to benchmarking RDF stores and has been used to measure both load and query[12] time.

Rohloff and co-workers chose to evaluate, among others, Sesame, {Swift,Big}OWLIM, and AllegroGraph through the LUBM benchmark. Their results confirmed that SwiftOWLIM was not designed to address large datasets, and Jena/Sesame with a MySQL backend was not suitable for handling large datasets (more than 100M triples). "Winners" are Sesame + BigOWLIM, Sesame + DAMLDB[13] and Jena + DAMLDB. The authors also noticed that Jena + DAMLDB provided faster answers to low complexity queries.

From the single client use case, Bizer & Shultz conclude that Sesame has good performances for small datasets (1 Million triples), and Virtuoso is faster for large datasets (25M to 100M triples). For simultaneous queries from multiple clients, Bizer & Shultz showed that Virtuoso outperforms its competitors, namely Sesame and Jena SDB/TDB. Regarding the overall performances, it is confirmed that Jena SDB/TDB is out-of-competition.

Based on [Rohloff et al., 2007] and [Bizer and Schultz, 2009], table 2.1 summarizes the capabilities and the overall performances of popular semantic repositories. In particular, we highlight the kinds of reasoning and their limitations with regards to the OWL-* families, and their materialization strategy. Virtuoso shows good performance but has limited reasoning capabilities. Sesame + OWLIM shows a good coverage of storage and reasoning capabilities (with static materialization) with in one hand, high performances in the context of small datasets, and on the other hand a slower behavior in the context

[12] the set of queries also cover reasoning soundness (returns correct responses to a query) and completeness (returns all of the correct responses to a query)

[13] dedicated and optimized database for storing DAML content, predecessor of OWL content

of large datasets. Performance of AllegroGraph could not be evaluated in the study of Rohloff and co-workers mainly due to a buggy early release. Finally Jena shows slow performances but is highly documented, extensible, and can be plugged to state-of-the-art DL reasoners.

Through this section, we briefly described standards and technologies aimed at representing, querying, reasoning and storing semantic data. These elements can be considered as raw material to address the challenging issue of semantic data integration in the context of collaborative e-science platforms. Section 2.3 focuses on state-of-the-art approaches addressing the distribution of SPARQL queries over multiple remote data sources.

2.3 Virtualized data integration

Contrasting with data warehousing approaches (also known as materialization based approaches) where the content of distributed data sources is replicated into a centralized data warehouse, distributed querying consists in pushing the query to the distributed data sources and gathering the results from the querier side. Distributed query processing approaches are generally presented as virtual integration since from the querier point of view, distributed data sources are virtually integrated as if a single data source was queried.

2.3.1 Distributed query processing approaches

Querying Linked Data [Bizer et al., 2009a] is presented in [Ladwig and Tran, 2010] as a kind of federated query processing since datasets are hosted by distributed data sources with established links across these hosted datasets. RDF, introduced in section 2.2.1.2 is the *de facto* standard to represent linked dataset over the Web. In the remainder of this thesis, we use the terminology commonly associated to the Linked Data area, namely *triple patterns*, *basic graph patterns (BGPs)* and *data sources*.

Definition 3 *A triple pattern (s,p,o) formalizes a statement involving a subject resource (s) linked to an object resource (o) through a predicate (p), where s, p or o can be either variables or values. In Conceptual Graphs (or Knowledge Graphs), triple patterns may be used to represent edges forming the knowledge graphs.*

Definition 4 *A Basic Graph Pattern (BGP) is a set of triple patterns linking together resources through subject or object labels, and predicates. A basic graph pattern may form a graph.*

Definition 5 *A data source, identified with a Universal Resource Identifier (URI), represents a set of triple patterns. In the context of distributed systems in general, but with Linked Data in particular, data sources may be geographically distributed over a network.*

Linked data querying present new fascinating challenges [Ladwig and Tran, 2010] related to the nature of a quickly evolving Web of data. These challenges are close to the characteristics of data sources. *Scalability* is a major issue since the volume of exposed data through a single source may increase constantly, in a context where data sources are multiplying. *Dynamicity* is also challenging since due to the nature of the web, there is no guaranty that data sources are always available. Moreover, the content of a data source may be changing rapidly. Even more challenging, data sources are considered as autonomous entities and present thus real *heterogeneities* in terms of access protocols, content size, source description, underlying data models, schemas, or ontologies.

2.3.1.1 General principles

The general phases for query processing have first been introduced in [Haas et al., 1989] and are re-used in [Kossmann, 2000] through an extensive state of the art over distributed query processing (DQP) approaches. DQP approaches are composed, in general, with the following steps :

(i) *Source selection* depends on the part of the query to distribute. It consists in selecting the most appropriate data source contributing to the result set. As a corollary, it may sometimes consists in determining irrelevant sources, thus preventing the engine to send distributed queries to sources that will not contribute to the result set. This phase can be generally considered as a static optimization since it is performed before the effective evaluation of the query with an a-priori knowledge on the content of data sources [Quilitz and Leser, 2008, Alexander et al., 2009, Harth et al., 2010, Görlitz and Staab, 2011]. Recently, source selection has also been proposed at query runtime to correct source ranking [Ladwig and Tran, 2010], thus addressing the dynamic aspect of Linked Data.

(ii) *Query rewriting:* this step consists in building a set of sub-queries that will be individually sent to each data source contributing to the result set. Some optimization may be realized in order to minimize the communication cost or maximize the processing that can be achieved remotely by each contributing source.

(iii) *Query planning:* given a set of sub-queries, the query planner searches for the optimal sequence of sub-query to be sent to remote data sources. Dynamic programming (or "iterative dynamic programming") is a classical approach [Selinger et al., 1979] widely used in query planning. It consists in iterating over all possible query execution plans and, given a cost estimation function, the algorithm selects the optimal plan.

(iv) *Query evaluation:* given a query execution plan, the evaluation consists in following the sub-query sequence to incrementally construct a final result set. The query evaluation requires joining intermediate results possibly resulting from distributed data sources. Distributed join can be realized either through *nested-loop joins*[14] or through

[14] which naively iterates over the distributed data sources

bind joins [Haas et al., 1997] and are more precisely described through the following distributed query processing approaches.

2.3.1.2 State-of-the-art approaches

DARQ [Quilitz and Leser, 2008] tackles semantic data integration through a federated approach. It aims at transparently federating multiple distributed and autonomous RDF data providers. The approach is based on *service descriptions* to allow accurate data source selection, and on a *query optimization* algorithm to reduce the cost of sub-query distribution. The general process consists in (i) parsing the initial SPARQL query, then (ii) based on the *service descriptions* the initial SPARQL query is decomposed into sub-queries (query planning), (iii) the resulting query plan (sequence of sub-queries) is optimized and finally (iv) distributed to all data providers (query execution).

Based on *service descriptions*, The DARQ query planner searches for relevant sources contributing to the query results and then builds a set of suitable sub-queries. *Service descriptions* are represented in RDF and encompass:

(i) *data description* through predicate *capabilities* in the form of existing predicates and their associated value constraints on subjects or objects ;

(ii) *access pattern limitations* [Florescu et al., 1999] to represent, from the data source point of view, triple patterns that must be included into a query in order to provide results ;

(iii) *statistical informations* describing the total number of triples N, and optional information describing the capability of the source for a given predicate. This optional information covers the number of triples for a given predicate $n(p)$, and two triple pattern selectivities, the first one considering that the subject is bound ($sSel = 1/n(p)$ by default), the other one considering that the object is bound ($oSel = 1$ by default).

Once available, these *service descriptions* are used by the query planner first, to perform source selection and second, to build sub-queries. For each basic graph pattern (BGP) constituting the initial SPARQL query, the source selection consists in matching all triple patterns against the predicate-based capabilities of the sources. These predicate-based capabilities constitute a limitation of DARQ since it is not possible to select a source if a predicate is unbound in the basic graph pattern. For example, queries containing triples patterns like `(<aPerson>, ?p, <aLocation>)` with `?p` a variable predicate, cannot be processed through DARQ.

Once sources have been selected, the query planner decompose the initial basic graph patterns into sub-queries with the following principle. A set of triple patterns is associated to each data source. If a triple pattern matches exactly one data source, it is added to the set associated to the data source, and this set is finally processed to build a single sub-query to be sent to this precise data source. On the contrary, if a triple pattern matches multiple data sources, it must be sent through independent sub-queries to all matching data sources.

The second contribution of DARQ consists in a set of logical (query rewriting) and physical optimizations of the sub-queries produced through the query planning phase. When FILTER expressions are available, the value constraints are reused to replace variables by constants. Moreover, based on the rules proposed in [Pérez et al., 2006, Pérez et al., 2009], the query optimizer merges several basic graph patterns forming the initial SPARQL query. Finally, when possible, value constraints from the FILTER expressions, are incorporated to the sub-queries to avoid transferring useless intermediate results. The physical optimizations consist in finding the query plan that will reduce the size of transferred data, thus achieving faster global querying. The DARQ optimizer is based on a dynamic programming algorithm to find the optimal query plan. The DARQ engine implements both a *nested-loop join* and a *bind join* [Haas et al., 1997]. While the *nested-loop join* naively iterates over bindings provided by the outer relation with bindings provided by the inner relation, the *bind join* aims at exploiting already known bindings to ultimately replace join variables by their already known values. Finally, sub-queries containing bound join variables are sent multiple times, and allow to reduce the size of transferred results.

SPLENDID [Görlitz and Staab, 2011] is a query optimization strategy aiming at transparently federating distributed SPARQL endpoints, by exploiting statistical data describing their content. Statistical data are provided by the Vocabulary of Interlinked Data (VoID) first introduced in [Alexander et al., 2009]. VoID has been designed from both the data producer and the data consumer perspectives, and aims at describing the content of a dataset (a set of RDF triples published through a single provider), its relation to other datasets (interlinking information) and the vocabularies used in the dataset (RDFS/OWL classes or properties). SPLENDID relies on two main components, namely the *index manager* and the *query optimizer*.

The *index manager* is responsible for associating each VoID description to the corresponding SPARQL endpoint through an index structure. Basically, two indices are provided, the first one describing the cardinality of a given predicate (or RDFS/OWL property) for a given endpoint, the second one describing the cardinality of a given type (or RDFS/OWL class) for a given endpoint.

VoID descriptions are exploited by the query optimizer to reduce the cost of the overall distributed querying. After the query rewriting step, the optimizer performs the (i) data source selection, and (ii) a join re-ordering optimization. The data source selection consists in, for each triple pattern, retrieving a matching data source for the type/predicate indices. If a predicate is unbound in the triple pattern, all data sources are associated since no information is available from the indices. The source selection is refined for triple patterns having bound variables because they are supposed to be located into a single data source. But VoID descriptions do not cover this particular case. To overcome this issue, SPARQL ASK queries are sent to all data sources to determine which one(s) actually host(s) the triple. Indeed, the result of the `ASK` query (`true/false`) indicates if a matching triple pattern can be found in the target data source. Except in the case of exclusive groups introduced in [Schwarte et al., 2011] and aiming at grouping triple

patterns that are exclusively hosted in a single data source, triple patterns are sent individually to the data sources to not miss any result that would be joined across distributed data sources.

Once the data sources have been accurately selected, the query optimizer performs the join re-ordering, based on a dynamic programming algorithm.

At query runtime, two strategies are used to distribute joins: *parallel joins* and *bind joins*. While parallel joins consist in sending requests in parallel to the distributed data sources and finally joining locally the results (a strategy adapted to selective joins, leading to small result sets), bind joins exploit the results obtained from the first join operand into the second operand.

SemWIQ [Langegger et al., 2008] is a collaborative knowledge sharing platform, targeting, in particular, e-Science communities which adopt ontologies to semantically describe scientific data. SemWIQ is based on a *mediator-wrapper* approach for handling data heterogeneity and provides a unified query interface through an extended SPARQL engine. The general approach consists in (i) analyzing a global SPARQL query designed with respect to a global schema (generally modeled through ontologies), (ii) selecting in a data source registry, relevant data sources based on the concepts referred to in the query, then, (iii) a canonical query plan is optimized through a federation optimizer which groups sub-queries through relevant data sources, and finally (iv) the global query plan is executed over possibly wrapped data sources.

The heterogeneity of data sources is addressed through the DR2-Server, for relational data sources. For non-RDF and non-relational data sources, a local wrapper may allow remote sub-query plans execution by performing native data access and transformations.

The distributed query processing is achieved as follows. SemWIQ relies on a concept-based data integration approach, thus requiring for all data to be described as instances of ontology classes. Based on the data source registry, source content statistics (RDF-Stats [Langegger and Wöß, 2009]), and declarative rules (*JBoss Rules*[15]), simplified Basic Graph Patterns (BGPs) are grouped together, in a first step, to be sent towards appropriate data sources through*Service* operators. Moreover, *Service* operators may be combined through (i) *Union* operators, allowing for querying several target data sources for a same BGP, and (ii) *Join* operators allowing for combining data resulting from several triple patterns sharing the same variables. In a second step, the resulting global query plan is re-ordered trough an iterative dynamic programming algorithm, aiming at finding the cheapest query evaluation plan.

SemWIQ has some limitations with regards to the expressivity of SPARQL. Indeed, (i) only SELECT queries are supported, (ii) subjects of triple patterns must be variables, and (iii) the distribution query processing does not support for unknown data types, *i.e.* ontology classes of data instances must be known statically (asserted data types), or implicitly deduced through description logic constraints.

[15] http://www.jboss.org/drools

FedX [Schwarte et al., 2011] is another SPARQL distributed query processor addressing the querying of federated SPARQL endpoints. The originality of FedX is that no assumptions are made over the content of the distributed datasource, and no metadata or statistics are needed to perform static sources selection. Similarly to the approaches presented above, and following the general distributed query processing model proposed in [Kossmann, 2000], FedX performs in a first step data *sources selection*, followed by a set of static join optimizations (triple pattern grouping and join re-ordering). Final dynamic join optimizations are performed to exploit intermediate results at query runtime.

On-demand data sources selection is realized without any *a priori* knowledge over the content of the federated data sources. Indeed, each triple pattern forming the basic graph pattern is annotated with a relevant data source through the execution of a SPARQL `ASK` query which populates a local cache preventing from re-executing the `ASK` when not necessary.

Once annotated with the relevant data source, triple patterns are grouped together when belonging to *exclusive groups*. This notion is introduced to characterize triple patterns that can be matched into a single data source, and thus do not require a costly distributed join. This grouping technique can drastically reduce the number of remote invocations and thus the number of transferred results through the network.

Then joins are re-ordered through a rule-based join optimizer. The proposed algorithm is based on the variable counting technique [Stocker et al., 2008] to evaluate the cost of the joins. This heuristic consists in estimating the number of free variable considering that *subject* variables are more selective than *object* variables, themselves more selective than *predicate* variables.

Finally, joins are optimized at query runtime by exploiting the intermediate results provided by the previously computed joins. The FedX engine is based on an optimized nested-loop join technique. More precisely, in classical bind join techniques presented above, mappings resulting from the left part of the join are pushed individually to the right part of the join leading to a huge number of remote invocations if the amount of intermediate results (mappings) is high. To avoid this issue, the proposed optimization consists in grouping a set of mappings into a single sub-query through SPARQL `UNION` constructs, thus avoiding numerous remote invocations. The single sub-query finally need to be post-processed to correctly associate the results of this `UNION` query to the global result graph. FedX additionally provides a parallel implementation through a pool of worker threads, and a pipelining strategy to exploit intermediate results as soon as they are available.

SPARQL-DQP. Buil-Aranda et al. propose in [Buil Aranda and Corcho García, 2010] to federate SPARQL queries over a set of SPARQL endpoints based on relational database distributed query processing (DQP) techniques. SPARQL-DQP relies on the transformation of a subset of SPARQL queries to their equivalent SQL queries. The distributed query processing is then supported by the OGSA-DAI and OGSA-DQP [Lynden et al., 2009] framework.

To address multiple RDF data sources querying, the SPARQL 1.0 language has been

slightly extended (SPARQL-D) through possibly multiple FROM clauses (included into SELECT queries and referring to a SPARQL endpoint) allowing to determine the source endpoint to which the query must be evaluated. After being parsed, SPARQL-D queries are transformed into a set of SQL queries (based on a relational algebra for SPARQL [Cyganiak, 2005]) which forms a logical query plan (LQP). The query plan is optimized and evaluated through OGSA-DQP. Queries can be evaluated through both *pipelined* and *partitioned* parallelism. While *pipelined* parallelism relies on a multi-threaded implementation of iterators, allowing to provide tuples as soon as they become available, for directly being processed by the next operator, the *partitioned* parallelism relies on several distributed nodes being queried in parallel and providing blocks of result tuples finally being merged by the DQP coordinator.

The RDF data publication is realized through a specific OGSA-DAI data resource, implementing the WS-DAIOnt-RDF(S) specification, and allowing for wrapped RDF data to be accessed through OGSA-DAI/OGSA-DQP.

Finally, a set of OGSA-DAI data-centric workflows are generated from the optimized, partitioned, Logical Query Plan (LQP). These workflows are connected through their inputs and outputs and enacted to return SPARQL results from the multiple virtual RDF data sources.

Optimizations for SPARQL 1.1 Federation. Well-designed graph patterns have been introduced in [Pérez et al., 2009] to restrict the usage of SPARQL OPTIONAL clauses thus leading to more effective query evaluations. Based on these results, Buil-Aranda et al. propose in [Aranda et al., 2011] optimizations based on query rewriting in the context of distributed SPARQL 1.1 Federation[16] query evaluations. The main idea consists in reordering the initial SPARQL query so that the most selective operators are executed first and OPTIONAL clauses – OPTIONAL being the most expensive SPARQL operator [Pérez et al., 2009] – are executed the latest. These optimizations, and a well-designed graph patterns checker, have been implemented in the SPARQL-DQP framework.

Dynamic source discovery Approaches presented above are characterized as top-down [Ladwig and Tran, 2010] since they rely on a statically fixed set of distributed data sources, for which it is possible to build *a priori* knowledge on the content of data sources. To adapt to dynamically evolving data sources, whose availability is unpredictable, Ladwig and co-workers propose a hybrid approach which consists in dynamically ranking sources at query runtime. The bottom-up query evaluation strategy consists in (i) extracting data sources from the query, (ii) starting to evaluate the query while discovering new data sources from intermediate results at runtime, (iii) evaluating the query against the new data sources, and (iii) terminating the evaluation when all possible data sources have been explored. A hybrid strategy consists in refining, at query evaluation time, an optimized query plan based on partial knowledge on the content of data sources. The

[16] which introduces the SERVICE and VALUES clauses to distributed sub-parts of a SPARQL query over a set of SPARQL endpoints.

proposed implementation rely on non-blocking join operators as introduced in [Hartig et al., 2009] and [Ladwig and Tran, 2011]. This approach is particularly promising since it allows for static optimization, while still considering the dynamicity, and the volatility of data sources distributed over the Web.

2.3.2 Discussion

In the previous section, we described general purpose semantic data federation approaches. These approaches are mainly focusing on performance issues faced when performing semantic data federation. Table 2.2 provides a synthesis of the static and dynamic optimizations aiming at enhancing the efficiency of distributed SPARQL queries.

	STATIC OPTIMIZATIONS			DYNAMIC OPTIMIZATION
	Source selection	*Query rewriting*	*Query planning*	*Query evaluation*
DARQ	Service capabilities	Triple pattern grouping Filter pushing	Dyn. prog.	Nested-loop join Bind join
SPLENDID	VoID descriptions SPARQL ASK	Triple pattern grouping	Dyn. prog.	Nested-loop join (parallel) Bind join
SemWiq	Source registry RDFStats	Triple pattern grouping Filter/Optional pushing	Dyn. prog.	Nested-loop join Bind join Dyn. join re-ordering
FedX	SPARQL ASK	Triple pattern grouping	Variable count.	Block bind join (pipelined)
SPARQL-DQP	–	Triple pattern grouping	OGSA-DQP query partitioner	OGSA-DQP parallelism & pipelining
Ladwig *et al.*	Partial knowledge	–	Refined dyn. prog.	Corrective src. ranking Join reordering Non-blocking joins

Table 2.2: Static and dynamic optimizations proposed by state-of-the-art federated querying approaches.

All these approaches address the *scalability* challenge for data integration through performance-oriented distributed query processing techniques. They aim at lowering the impact of massive semantic data querying on end-users.

The SPARQL-DQP approach, does not address transparent semantic federation, and rely on SPARQL *Service* clauses. These clauses help in implementing distributed query processing when the content of data sources is well partitioned and known at query design time. However, this assumption does not hold in many real use cases, and specially in the context of life-science collaborative platforms. This approach is not suitable in the context of dynamic knowledge base federations in which pre-designed SPARQL 1.1 queries must be adapted to take into account the data source availability.

Addressing transparent[17] semantic federation, DARQ, Splendid, SemWiq, Fedx, and Ladwig and co-workers , address performance issues and tend to reduce the amount of data communication between the federation querier and the multiple remote data sources. Precise data source selection thus becomes crucial to prevent unnecessary communications through optimized distributed joins. Whereas Splendid and DARQ are based on *a priori* knowledge on the content of data sources to perform source selection, this task is dynamically achieved through SPARQL ASK queries in the FedX approach

[17] from the query designer point of view

which provides better flexibility and prevents from maintaining additional data source content descriptions. However Ladwig and co-workers propose an interesting hybrid approach in which source selection can be driven by *a priori* source content descriptions and a dynamic refinement, thus allowing to adapt to the dynamic nature of data sources federated over the web.

With respect to the *knowledge* challenge for life-science data federation, all these approaches are based on SPARQL distributed querying, thus allowing, in theory, to exploit ontologies and possible inferences from federated query processing. However all approaches suffer from limitations with respect to the expressiveness of the supported SPARQL features. All of them only support SELECT queries. They may impose constraints with regards to the basic graph patterns supported. For instance, SemWiq cannot federate triple patterns with bound subjects, and all instances must be associated to an ontology class.

Towards a better exploitation of domain ontologies through life-science collaborative platforms, it is critical to find a good balance between expressiveness and performance when federating semantic queries at large-scale.

2.4 Data sharing through Life-science collaborative platforms

Several e-Science collaborative platforms resulted recently from large-scale projects addressing collaborative e-Health activities. @neurIST, ACGT, CaBIG and BIRN, aimed at providing large-scale distributed computing and storage resources to ease the setup of collaborative e-Health activities, and tending to accelerate scientific discoveries.

The remainder of this section describes state-of-the-art collaborative e-science platforms with a particular focus on the methodologies adopted to tackle data sharing through autonomous, distributed and possibly heterogeneous scientific data sources.

2.4.1 Life-science collaborative platform examples

CaBIG. The Cancer Biomedical Informatics Grid (CaBIG) results from a joint initiative of the US National Cancer Institute and the UK National Cancer Research Institute. It aims at providing a software infrastructure, CaGRID [Saltz et al., 2006] adopting a service-oriented and model-driven approach, dedicated to the management and the analysis of multi-source heterogeneous biomedical data. In CaGRID, structural metadata, giving the form of data units, are implemented as UML[18] models. CaGRID data services expose native data (raw files or relational databases) as object-oriented UML models. Data can be queried through the CaGRID object-oriented Common Query Language (CQL). CQL queries, expressed in terms of objects-attributes-associations, are navigational queries in the sense that data retrieval is specified through traversing UML class diagrams which reflect metadata structure. On top of that, domain metadata are concep-

[18]UML stands for the Unified Modeling Language, an industrial software design standard supported by the Object Management Group (OMG).

tualized through the NCIt ontology [Hartel et al., 2005], whose concepts are associated to structural UML models through linkage *referent* metadata.

However CaGRID CQL queries do not benefit from the semantic annotations of UML models, intended to address semantic inter-operability. Beltran & al. propose in [González-Beltrán et al., 2010, 2012] to let end-users concentrate on the domain entities rather than understanding the several underlying fine-grained UML class diagram. The proposed Cancer ONtology QUErying SysTem (COnQueSt) for CaBIG semantic data federation follows the LAV approach since the NCIt ontology is considered as the global schema and queries over the global-schema are rewritten following the data source local-schema. COnQueSt is based on two components, an *OWL generation service* and a *Semantic query service*:

- the *OWL generation service* is responsible for (i) generating OWL classes and properties from the annotated UML models exposed through Data services, and (ii) for importing the NCI thesaurus ontology.

- the *Semantic query service* is responsible for mediating the different abstraction levels from ontology-based queries. More precisely, this service rewrites ontology-based queries in CQL, or DCQL (a distributed extension of CQL allowing for basic multi-source data federation).

BIRN. The *Biomedical Information Research Network* (BIRN) is a national initiative from the USA, aiming at supporting Biomedical research through multi-source data sharing and online collaborations. The BIRN mediator [Ashish et al., 2010] targets heterogeneous and distributed biomedical data integration while still coping with the autonomy of data providers. More precisely, data sources participating into the BIRN network do not need to adapt their legacy environment, and keep the control over their hosted data.

Data sharing is achieved in BIRN through the concepts of *virtual organizations* (VOs), communities of data providers and data users sharing some specific objectives through data access, sharing or analysis ; and *domain models*, which consist in agreed views on data at the scale of the VO, useful in the context of this VO.

Three components target data integration in BIRN:

- The *mediator*, follows the GAV approach (with envisaged support for LAV and the OWL2 query language (OWL2-QL) in future works) to data federation and exposes multiple data sources as a single virtual database. Users query the virtual database by using terms of the domain model. The mediator is then responsible for selecting appropriate data sources, based on the description of their content, and for rewriting domain-level queries into source-level queries.

- The *distributed query evaluation engine* is based on the OGSA-DAI/OGSA-DQP framework to evaluate the source-level queries resulting from the mediation step. While OGSA-DAI [Antonioletti et al., 2007] propose streaming data-centric workflows to access multiple wrapped data sources, OGSA-DQP [Lynden et al., 2009]

extends the OGSA-DAI framework with an engine capable of building distributed query plans and performing optimized (aiming at pushing the workload to the remote data sources) distributed evaluations.

- *Source wrappers* are based on OGSA-DAI resources which include a set of predefined connectors to access either relational databases, XML data bases or raw file systems.

With respect to security, a critical concern for biomedical applications, BIRN relies on the Grid Security Infrastructure [Lang et al., 2006] (GSI) which provides encryption through Public Key Infrastructures (PKIs) and thus enforces data privacy, and a logging and auditing component letting data source providers and administrators in general, track data access.

@neurIST. The @neurIST European Integrated Project [Benkner et al., 2010] provides an infrastructure dedicated to biomedical research addressing (i) the secured management of heterogeneous and distributed clinical and research datasets, and (ii) their complex processing through high throughput computing services. The clinical application of the project targets the investigation and the clinical treatment of cerebral aneurisms. Clinicians and researchers benefit from seamless secured access and computations over a federation of distributed and heterogeneous aneurism data sources. The @neurIST middleware is organized through two main components. While *@neuInfo* provides data management services through a generic data management and integration framework, *@neuCompute* provides an autonomous grid middleware topping the Fura middleware [Arbona, 2009] which allows for distributed data processing over heterogeneous computing infrastructures through jobs dispatching, parallelization and monitoring.

The *@neuInfo* data management layer relies on a mediator-based, global-as-view (GAV), approach to cope with data source heterogeneity. On top of the OGSA-DAI/OGSA-DQP [Antonioletti et al., 2007, Lynden et al., 2009] frameworks, *@neuInfo* defines a Data Access Services (DAS) and a Data Mediation Services (DMS) which both provide, through a virtual schema (also known as "global schema" in GAV/LAV approaches), a transparent access to possibly heterogeneous data sources. DMSs are responsible for (i) translating queries expressed through the global schema, into local schemas specific queries, (ii) accessing remote data sources, and (iii) translating back local result data according to the global schema. The underlying mediation engine [Koehler and Benkner, 2009] is responsible for generating the mappings between the global schema and local schemas, based on the @neurIST ontology [Boeker et al., 2007]. Data mediation is achieved through the following steps:

1. *Virtual query plan generation:* this first step consists in parsing the initial relational query based on the virtual schema to generate a virtual query plan. This step is achieved through the OGSA-DQP query parser.

2. *Concrete query plan generation:* this step addresses the data heterogeneity between the virtual schema and the several concrete schemas (also known as "local schemas" in the

GAV/LAV terminology.). It is achieved through the proposed Data Mediation Service (DMS) in the following phases:

(i) Virtual query parsing identifies for each OGSA-DQP operator forming the virtual query plan, the corresponding mapping rules in the mediation schema ;

(ii) Partial mediation query plan generation consists in, for each OGSA-DQP operator, building the corresponding concrete query plan resulting from the application of the mapping rules ;

(iii) Mediation query plan assembling finally consists in substituting the OGSA-DQP operators in the virtual query plan with the partial mediation query plans.

3. *Concrete query plan execution* is achieved through the OGSA-DQP engine, which distributes the query plan partitions to appropriate evaluation services.

With regards to security concerns, @neurIST preserves the autonomy of each participating data source. Since @neurIST supports both research on aneurism and clinical decision, patient data privacy becomes a high-order priority. Due to the distributed nature of the @neurIST platform, the security architecture targets the authentication and the authorization of several stakeholders across multiple domains and border. Security enforcement has been implemented through (i) the *Public Key Infrastructure* (PKI) to authenticate users into a single *virtual organization* (VO), and to guaranty message- and transport-layer security, and (ii) authorization policies (based on role and location attributes) and a distributed access control mechanism. The distributed access control consists in, at client-side, (i) requesting a security token to the local security token service, then (ii) incorporating the received token which includes the role and location attributes to the service request, and finally (iii) performing the service call. From server-side, the remote access control is performed as follows : the first step consists in calling the remote security token service, then checking locally the authorizations, and finally, depending on the security decision, invoking the service. This distributed access control has been implemented through the Security Assertion Markup Language (SAML) and the WS-Trust OASIS[19] standard.

ACGT. ACGT was an EU funded project (2006-2010) aiming at improving Medical Knowledge Discovery through a semantically rich infrastructure supporting multicentric and post-genomic clinical trials [Bucur et al., 2011]. Post-genomic data covers a wide variety of natures, going from pathology data, genomic data, micro- and macro-imaging data, to clinical data collected through Case Reports (symptoms, treatments, etc.). A large heterogeneity in data sources must be tackled to achieve post-genomic clinical trials. Moreover, external source of knowledge are generally considered in the context of post-genomic trials, which require coherent and efficient querying over distributed data sources.

[19] OASIS stands for the Organization for the Advancement of Structured Information Standards, providing standards for e-business and web-services.

The ACGT Master Ontology on Cancer [Martín et al., 2008] has been built as the semantic interoperability framework for the Data Access Layer of the ACGT platform. The Data Access Layer cope with (i) syntactic interoperability through database wrappers based on the OGSA-DAI framework, and (ii) semantic interoperability through a dedicated Semantic Mediator [Martín et al., 2007] based on a Local-as-View (LAV) approach, which adapts to continuously reaching new data sources:

- the proposed Data Access Service offers, through an OGSA-DAI web service interface, a single query interface based on the SPARQL language. Relational database and DICOM[20] wrappers have been developed through the D2RQMap [Bizer and Seaborne, 2004] language. They allow SPARQL querying over virtual RDF graphs.
- the proposed Semantic Mediator is based on a Local-as-View approach in which the ACGT MO ontology acts as the global schema and provides (i) a mean to setup the local views and (ii) the semantic referent over which user queries can be formulated. A SPARQL query rewriting mechanism has been developed to adapt global SPARQL queries into source-specific SPARQL queries. Results are then accumulated into a single OWL document representing ontology instances and returned back to the querier.

2.4.2 Discussion

We briefly introduced above four popular life-science collaborative platforms resulting from US or EU large initiatives, namely CaGRID, BIRN, @NeurIST, and ACGT. All these platforms share a lot of commonalities to address the main challenges of life-science data integration.

Towards data source *autonomy* preservation, these four platforms adopted federation approaches and thus distributed query processing techniques. They all consider distributed and heterogeneous data sources. *Distribution* and *heterogeneity* sub-challenges are addressed by @NeurIST, ACGT and BIRN through an extensive use of the OGSA-DAI/OGSA-DQP frameworks and platform-specific mediation services based on data source wrappers and query/data adapters. The ACGT approach is based on a *Local-As-View* mediation and thus provides better *scalability* when adapting to new data sources reaching the collaborative platform. Indeed, it is the responsibility of the new participating site to provide site-specific schema mappings to resolve the heterogeneity with the global schema. The global schema does not need to be adapted and the other participating sites are not impacted. BIRN and @NeurIST mediators are based on a *Global-As-View* approach. They provide less scalability since the global schema may evolve to adapt to new data sources thus impacting the other participating data sources. With respect to these *distribution* and *heterogeneity* sub-challenges, CaGRID differs since it is based on a model-driven approach, and relies on project specific query languages and query processing engines (CQL/DCQL) to address, possibly distributed, object-oriented querying over UML models and the underlying data units stored as raw files or relational data.

[20] DICOM stands for the Digital Imaging and Communications in Medicine standard.

Even if CaGRID relies on the NCIt domain ontology, participating data sources must describe their data in terms of CaGRID specific structures (Common Data Elements), and CaGRID alters consequently the data source *autonomy*.

The *security* sub-challenge is part of the *autonomy* challenge since life-science data sources are responsible for the sensitive biomedical data they host, and must comply to strict security policies. Although all described platforms rely on Grid Security Infrastructure (GSI) and the underlying Public Key Infrastructure (PKI), they address differently data privacy and access control in the context of distributed and autonomous life-science data sources. The most advanced and suitable approach has been developed by the @NeurIST project and relies on distributed security token providers and distributed access control enforcement points.

All four initiatives rely on domain ontologies to capture and conceptualize knowledge associated to the integrated data, thus constituting a semantic referent (*knowledge* challenge for data integration). However, several levels of ontology exploitation are provided through these platforms. The BirnLEX ontology (which evolved through NIFSTD ontologies and NeuroLEX wiki [Imam et al., 2011]) aims at providing a controlled vocabulary to annotate BIRN data sources, but its usage in the context of BIRN data source federation is not clear, even if multi source data integration is envisaged in [Imam et al., 2012]. More precisely, [Ashish et al., 2010] focus on providing unified relational querying capabilities, and it does not seem possible yet to express semantic queries over NeuroLEX concepts to retrieve distributed data from federated BIRN data sources. Similarly, a specific ontology resulted from the @NeurIST project but only relational federated querying can be achieved through OGSA-DAI and a mediation service. More recently, semantic querying on top of federated CaBIG data sources has been studied [González-Beltrán et al., 2012]. However, the proposed approach follows an *Extract-Transform-Load* data warehousing approach to populate a semantic repository from distributed CaBIG data sources. But this approach does not appear to be scalable in the context of continuous increases of data amounts, and does not cope with the autonomy preservation of data sources managing hardly relocatable data. Finally ACGT is the only platform providing semantic federated querying through a SPARQL query interface, thus allowing to exploit the ACGT Master ontology, and possible inferences, directly when querying distributed ACGT data sources.

The most challenging tasks for collaborative life-sciences appear thus as (i) being able to preserve the autonomy of participating distributed data sources while still providing sufficient security guarantees, and (ii) exploiting domain ontologies at query design-time and run-time over autonomous distributed and heterogeneous life-science data sources.

2.5 Conclusion

In this chapter, we introduced the main concepts related to data integration and knowledge sharing needed to build collaborative life-science platforms. Addressing knowledge data management, section 2.2 briefly introduced standard languages and reference tools

dedicated to semantic data representation, querying, reasoning, and persisting. State-of-the-art approaches dedicated to semantic data federation have been described from a general-purpose and performance-oriented perspective in section 2.3, and from an life-science application perspective in section 2.4.

The *security* sub-challenge for life-science data integration is addressed in chapter 3 as a contribution to support the setup of secured multi-centric studies.

To address the *autonomy* preservation of distributed and heterogeneous data sources participating in life-science collaborative platforms, we propose in chapter 4, methods and algorithms to achieve transparent semantic data federation with a particular focus on both language expressivity and performances.

Towards a better exploitation of ontologies in the context of life-science data integration (*knowledge* challenge for life-science data integration), chapter 4 aims at bridging the gap between generic performance-oriented approaches described in section 2.3 and collaborative life-science platforms described in section 2.4, through distributed query processing strategies balancing both performance and expressivity. We also propose a set of experiments in chapter 8 evaluating our approach at large-scale (section 8.2), and over distributed and heterogeneous neuroscience data sources (section 8.3).

Key Points

- *Life-science data repositories cover a wide variety of data natures which leads to interoperability issues in collaborative activities.*
- *Semantic data representation have gained a lot of interest to address interoperability issues and to perform highly expressive graph-based querying/reasoning.*
- *Data integration approaches based on a centralized data warehouse are not suitable in the context of autonomous life-science data source, which manage hardly relocatable data.*
- *Local-As-View approaches for virtualized (federated) data integration better adapt to dynamic federations (scalability) but require complex query rewriting mechanisms.*
- *General purpose and performance-oriented semantic data federation approaches still suffer from expressivity limitations.*
- *Although collaborative life-science platforms rely on domain ontologies, they generally only support relational distributed querying and do not allow semantic querying over distributed/heterogeneous data sources.*

CHAPTER 3

Secured collaborations in a life-science platform

Contents

3.1 Introduction

Federating data sources is increasingly needed to address many challenging societal issues such as large scale epidemiology or rare diseases studies. Grid technologies are appealing to deal with the distribution of life-science data providers [Breton et al., 2005]. However, the take up of grid technologies is slowed down by the difficulty to manipulate sensitive medical data in a distributed environment on the one hand, and the reluctance of medical centers to openly deliver valuable data sources which acquisition is a costly process on the other hand. In pre-clinical or clinical research, the fear to loose control over local data sources often counter balance the temptation to share data in spite of the growing need for collaborations through multi-centric studies.

For example, the NeuroLOG project, introduced in section 1.4.1, aimed at building a federation of collaborative neuroscience sites to address challenging public health problems. The sites participating to the federation have both collaborative interests and competitive activities. In addition, each site has set up a local and satisfying working environment which should be preserved. Consequently, it is needed to offer a data federation

system which does not compromise sites data beyond the objects of collaboration decided and which does not interfere with the normal autonomous operation of the sites.

3.1.1 Motivations

As an example of valuable neuroimaging data, we consider here the setup of neuroscience experiments which rely heavily on the availability of brain image databases. The setup of these experiments is generally a long and costly process in which neuroscientists are involved to (i) select a target (and control) population, (ii) design specific acquisition protocols by parametrizing the image acquisition device to obtain signals or reconstructed images relevant to their studies, (iii) post-process the acquired data to finally obtain as much as possible homogeneous brain databases (*e.g.* intensity normalization, data registration). Producing such valuable databases is a long and costly process but remains the first requirement to launch further scientific activities. Considering this effort, the set-up of such databases is often the source of collaborations between scientific partners sharing similar objectives.

However, sharing life-science resources (data and application codes) in computational sciences still remains challenging and face two major obstacles.

Regulatory obstacles. The sharing and processing of personal medical data for healthcare or bio-medical research must strictly conform to national and supra-national (for european member countries) policies. Rahmouni and co-workers provide in [Rahmouni et al., 2010] several examples of privacy requirement at the european scale. Most of european data protection laws ban the processing of personal data, except if data is priorly anonymized (or de-personalized). It consists in eliminating all pieces of data possibly leading to the identification of the corresponding person. Another common regulation is that data must be collected in specified, explicit and legitimate purposes (Article 6b of the EU directive 95/46/EC, 1985), while still allowing for further scientific exploitation provided that Member States present safeguards.

As an example of national legal issues, the CNIL[1] – the french national commission for information technology and civil liberties – provides several guidelines helping information technology providers or users to conform with laws. Some of these guidelines are dedicated to health data management. For instance, each health practitioner or each person responsible for files must comply to security obligations. He or she must take the necessary precautions to guarantee the confidentiality of data and ensure that they are not communicated to non-authorized parties. Thus, sensitive health data must be accessible to people granted with the necessary permission only. As more practical guidelines, some technical measures must be taken in the context of health applications deployed over a network. For instance, a connection to the system by several users using the same login and password must be forbidden, connections and data usage must be journalized,

[1] http://www.cnil.fr/english

personalized data should be partially or fully ciphered (as well as communications), data access must be restricted to authorized users, etc.

Social obstacles. Stodden provides in [Stodden, 2010] a literature review leading to the conclusion that decision on sharing is based on self-interested and communitarian motivations. In addition, Stodden proposes an empirical study[2] of sharing behavior of researchers manipulating both data and application codes (*i.e.* computer scientists). Her study reveals the main obstacles hampering data and codes sharing in computational sciences communities. On one hand, the top reasons motivating scientists to share data and codes are *"encouraging scientific advancement"*, *"encouraging sharing and having others share with you"*, and *"being a good community member"*. On the other hand, the top reasons motivating to prevent sharing are *"the time it takes to clean up and document for release"* and *"the possibility that your data/codes, may be used without citation"*, and to a lesser extent "*competitors may get an advantage*" or *"the potential loss of future publications based on this data/codes"*. While reasons for sharing appear as driven by communitarian considerations, reasons for not sharing appear as driven by personal interests.

In line with this study, Goble and co-workers [Goble et al., 2011] depict scientists as self-interested (as any other group of persons) mainly motivated by finding funding (and sufficient resources to achieve there research) and building their reputation. In this context, sharing happens when scientists are rewarded for it, and when both sharing risks (loosing competitiveness) and sharing costs (time and effort to prepare reusable data/codes, or potential sustainability/support constraints) are minimized.

These barriers have been faced during the NeuroLOG project in which distributed neuroscience partners required for the setup of collaboration while still coping with (i) competitiveness and ownership concerns and (ii) regulatory issues applying to sensitive biomedical datasets.

3.1.2 Related works

We briefly study in this section the existing approaches to enforce security in health-oriented distributed systems.

3.1.2.1 Basic notions of security in distributed information systems

Authentication is a process guaranteeing the identity of parties when accessing/exchanging information. Public key infrastructures [Housley and Polk, 2001] (PKIs) and the X509 norm provide a standardized way for authentication in telecommunication systems. A certification authority (*CA*) is a component empowered to deliver certificates as proof of identity. In addition, these certificates are signed by the *CA* so that a certificate is non-repudiable and its provenance is traceable. The signature process can be configured

[2] based on interviews of conference attendees for NIPS (*Neural Information Processing Systems conference*) and ICML (*International Conference on Machine Learning*)

hierarchically so that the signed certificate is also empowered to deliver certificates, and thus becomes a "*sub-CA*". A certificate is composed of descriptive information such as the name of the owner, its organization unit, the identifier of the certificate issuer, etc. A pair of public/private keys is associated to the certificate and used for both asymmetric encryption, and owner authentication.

Authorization is the process of verifying that an authenticated user has the privilege (is authorized) to perform a certain operation or more generally to access a certain resource. Note that it is mandatory to check beforehand, that the user is authenticated. Such combination of authentication and authorization is also known as *Access Control*. As an example, the user "John" requests for the removal of a dataset, the authorization procedure will check the security policy to assert that removal privilege is granted to "John" for this dataset. If not, the removal operation will fail.

Matrix authorization rules, also known as Access Control Lists (ACL) consist in basic rules which assign a specific permission from a subject to a protected resource. Such a rule can be seen as a triplet (s, a, r) in which s is the subject, a is an access type (permission), and r the resource to protect. The main disadvantage of this model is that the security policy (the set of rules) is made of "per user" rules, implying an high cost of management.

Whereas permissions are assigned to users in ACL, they are assigned to roles (the user possible functions) in role-based access control (RBAC) [Sandhu et al., 1996] in such a way that security policies become lighter and more understandable. The cost of policy management is then reduced. The concept of group is often distinguished in this kind of patterns. A group is considered as an identified set of user (as in operating systems) whereas a role, identifying a particular function assigned to users, is rather a collection of permissions.

Confidentiality guaranties no leaks of sensitive or private information in the system. It usually relies on cryptography to assert that only authorized people can access to confidential resources. *Encryption* is closely related to confidentiality. It relies on cryptography techniques to protect sensitive or private information. This process transforms a message with an algorithm that makes it unreadable (*ciphering*). People which possesses the deciphering key can access the message. Symmetric algorithms, AES for instance, use the same key for encryption / decryption whereas asymmetric algorithms, RSA for instance, use a public key for encryption and a private key for decryption.

The Secured Socket Layer (SSL) is based on asymmetric cryptography to guaranty confidential network communications. It uses X509 certificates and their associated private keys. A client C owns an X509 public certificate C_{cert} and an associated private key C_{key}. A server S recognizes a number of client certificates that are stored locally in a trust store, or certificates which have been preliminary signed by a trusted certification authority. When a client connects, the server and the client proceed with an hand-shaking: the server checks the identity of the client. If C_{cert} is valid (i.e. it is not outdated and it was

signed by a known certification authority) the socket is opened and the communication begins. Otherwise the server closes the connection. An SSL channel can be configured to perform dual hand-shaking: both the client and the server authenticate to each other.

3.1.2.2 State-of-the-art approaches

Widely adopted in production grids (*e.g.* EGI, the European Grid Infrastructure), the Virtual Organization Management System VOMS [Alfieri et al., 2003] is an authorization service, delivering an extended proxy certificates to an authenticated user. This extended certificate, (i) enables single sign-on across the grid, (ii) enables the authentication to various Virtual Organizations (VOs), and (iii) binds groups and roles to the user through attributes embedded into certificates.

To tackle the centralization issues of VOMS, the Shibboleth's [shi] decentralized approach to authentication and authorization, is gaining adoption in grid communities. The central idea is that remote authentication is delegated to the user's referee authority. In other words, the question *"Where are you from"* is asked to the client. Hence, the user's authority authenticates him/her, and generates a security assertion including user's security attributes. The main advantage of this approach is the decentralized authentication and attributes assignment through Shibboleth authorities.

Chadwick and co-workers propose PERMIS [Chadwick et al., 2008], a modular authorization infrastructure suitable for grid environments. This flexible and distributed framework provides (i) credential and hierarchical RBAC policies management and (ii) an access control decision engine, including credential validation.

Sinnott and co-workers analyzed the management of roles through large scale and dynamic virtual organizations (VOs) and the impact of both centralized and distributed approaches [Sinnott et al., 2008]. Centralized VO roles management is well suited in the context of static virtual organization in which roles and users do not change rapidly. The management of roles is simpler and easier since a single VO administrator is responsible for role assignment to users. But it can also be seen as a disadvantage, especially in the context of life-science collaborative platforms requiring for autonomy. These approaches are also considered as less scalable since the VO administrator needs a detailed knowledge on all VO users and roles. On the other side, decentralized role management allows for more dynamic and scalable collaborations (particularly needed in the context of this work). It also provides a better autonomy since sites make their own local decisions on their resources, which is decoupled from the role assignment to users. However, in decentralized approaches, roles may be scattered over several collaborating sites, leading to non-trivial role usage and advertisement thus hampering their adoption.

An in-depth review of federated access control through grid authentication and authorization technologies have been proposed by Jie and co-workers [Jie et al., 2011]. They promote Shibboleth as (i) a single-sign-on scalable user-friendly authentication solution and (ii) a strong basis to support, coupled with PERMIS, fine-grained authorization in grid environments. However, Jie and co-workers note that Shibboleth cannot be considered as a universal solution since clinical sites/hospitals or commercial organizations

may not trust academic identity federations, and security standards are still flux (W3C, IETF, OASIS standards).

Data protection is a major requirement in health environments. Elaborated solutions are proposed to protect from the "insider abuse" issue such as the use of Shamir secret sharing scheme in [Seitz et al., 2003] and in the Medical Data Manager [Montagnat et al., 2008a] leaning on the Secure Storage Service provided by the gLite middleware [sec, 2005] [Scardaci and Scuderi, 2007]. Blanquer and co-workers propose in [Blanquer et al., 2009] an enhancement which consists in also protecting decryption keys from "insider abuse".

Stell and co-workers describe in [Stell et al., 2007] an in-depth analysis of security requirements for Neuroscience dedicated e-Infrastructures: *"On the one hand, the local administrator must find a way to translate their local policies to a common interface that can be understood by remote users as local. On the other hand, there must be a way for the remote user to gain fine-grained and secure access to individual data fields and parameters"*.

This last statement illustrates the partly conflicting needs for (i) autonomous data providers, responsible for the sensitive medical data they host, and (ii) the need of end-users to access distributed data in the context of multi-centric collaborations. The proposed security measures in the context of life-science data integration (see the security challenge introduced in section 2.1.3.2) are thus non-trivial to finely balance these conflicting interests.

3.1.3 Requirements for secured life-science collaborations

In this chapter, we propose a security model addressing collaborative life-sciences driven by partly conflicting requirements, namely

(R_1) medical data protection ;

(R_2) distributed control over data sources with prevailing local policies ;

(R_3) support for multi-centric studies involving data sharing ;

(R_4) autonomous sites administration.

R_1 is achieved through classical data encryption techniques. R_2 and R_3 are achieved through a novel distributed data access control policy described in details in this chapter. In the system proposed, there is no global authority nor any super-administrator with specific access rights, thus guaranteeing to the system users that they solely control the access to the data they own. The solution proposed addresses R_4 through minimal centralization. It was designed for simplicity of use and lightened administration overhead.

Although collaborative life-science platforms involve several data representations, such as raw file, relational databases, or semantic data, the proposed security model only target raw files. Addressing these non-trivial security requirements over other data representations such as tables (relational databases), or graphs (semantic data) would open interesting perspectives.

3.2 Life-science distributed security model

We consider in this chapter life-science collaborative platforms in which the contributing centers have set up working environments over years that are well suited for their site-specific objectives. Although federating resources is generally considered as an high-order priority, the system proposed has to adapt to these legacy environments and most importantly, it should not interfere with the normal autonomous operation of the sites (R_4). The proposed security model is based on thoroughly validated methods to identify collaborating partners. The user credentials are managed at the application level in such a way that the end-user authenticates only once (single sign-on).

3.2.1 From independent to collaborative trust domains

Each center involved in the federation is responsible for registering its own local users, as it is usually the case in autonomous entities. It is administrating a local certification authority (Site CA), empowered to deliver and sign site user certificates, thus delineating a local trust domain. The widely adopted X509 certificates used contain descriptive information such as the name of the owner, her organization unit, and the identifier of the certificate issuer. They are non-repudiable proofs of identity all over the trust domain.

To enable multi-centric studies (R_3) in a secured environment (R_1), the research centers involved have to interconnect their trust domains. Our system is relying on a root certification authority taking part in a coordinating node named *Federation Registry*. The *Root CA* is delivering certificates to all participating sites (*Site-CA* certificates). It delegates certification ability to each site administrator for local users. End users receive individual certificates that are thus resulting from a 2-levels signature chain (*Root CA* and *Site CA*). Being ultimately signed by the *Root CA*, the user certificate of a particular site is used to establish a secured communication with another participating site.

Figure 3.1 depicts the use of the 2-levels trust chain to establish secured communications, either locally to one participating site, or to another site of the federation. A user of *Site A* owns a signed certificate used to establish secured communication with all services. When she attempts to retrieve a data from *Site B* for instance, the certificates signature chain makes identity control possible at *Site B* without prior registration of the user. The trust chain resulting from the signature of both *Root CA* and *Site A* is used by *Site B* to determine that the request comes from a legitimate user at the scale of the federation.

3.2.2 Data protection

As outlined in R_1, data protection and access control are critical in any healthcare dedicated information system. Another advantage of X509 certificates is that they enable secured communications between users and service providers within and across local trust domains through Secured Socket Layers (SSL). SSL makes an extensive use of asymmetric cryptography to perform mutual authentication and to negotiate communication

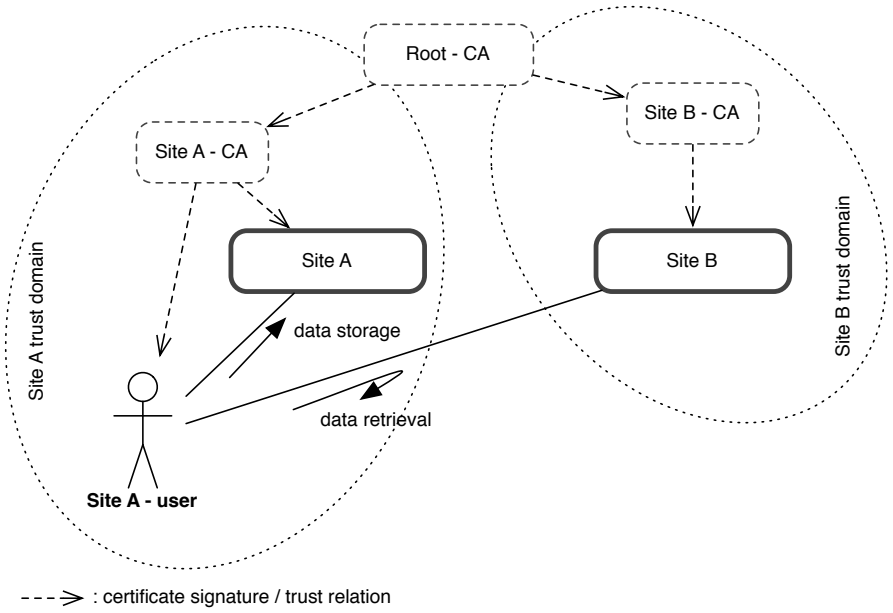

Figure 3.1: Bridging independent sites trust domains.

encryption keys. We use the AES encryption algorithm to protect data content. Two protection policies, corresponding to two levels of privacy, are implemented in our system: (i) minimally, all communications are protected so that data is encrypted during transfers only, (ii) potentially, the system can be configured so that data is also encrypted on disk. The first policy is acceptable in a research context where the data manipulated is de-personalized prior to its importation in the system. It facilitates data manipulation locally while guaranteeing that data is protected when transferred outside. The second policy addresses the stronger data protection that must be considered in the context of clinical deployment or sites on which all local users should not access to the complete data base.

3.2.3 Decentralized access control policy

The key contribution of this work is a decentralized data access control policy where local policies prevail for controlling access to local data (R_2) in a collaborative environment (R_3). Our access control mechanism is based on traditional Role-Based Access Control (RBAC) [Ferraiolo et al., 2001] for ease of use. The remainder of this section exposes a decentralized role-based access control across the federation in which multi-centric study instigators independently assign roles to users, and site administrators manage the sharing of resources by assigning them roles and permissions.

The proposed RBAC access control policy decouples and distributes two administration tasks: (i) the assignment of roles to users and (ii) the definition of access rights for

each role. Most existing RBAC systems are centralized and these two tasks are simultaneously managed in a single system.

Data access control policy. On each site, at least one administrator has special capabilities to register the site users and to maintain the site's access control policy. To satisfy our requirements, the assignment of user's role and the definition of roles access rights are decoupled. In the distributed environment, users may belong to different trust domains (a system administrator usually has no complete view of the potential users of the system) while the access rights to data of a given domain has to be ultimately controlled by this domain's administrator. The compromise to enable collaborative work while ensuring site-wise access control to local data is as follows:

- Site administrators are capable of creating federation-wide roles, as many as needed to describe their access control policies;
- The creator of a role controls the assignment of all system users to that role: the management of a particular role is centralized on one of the sites.
- Each site administrator controls the assignment of federation-wide roles to permissions related to their local resources.

A user is granted access to a data item if she belongs to at least one role that is locally authorized to access this item. This policy framework ensures that sites solely control access to their data: only a site administrator can bind some role to her data. It also ensures that each role is well defined and administrated: only the role creator can bind users to that role. It implies a collaboration between the data owner and the role creator: the data owner agrees to make some data accessible for a particular role (*e.g.* in the context of a particular multi-centric study) ; the role administrator is trusted and recognized as the administrator for this particular study. Any user in the federation can collaborate to the study. Through the certification chains, the role administrator can validate the identity of any user before assigning the role to her. Finally, the roles are guaranteed to be unique federation-wide through the coordination registry.

3.3 Results and implementation

The proposed distributed access control policy is illustrated in the context of a secured, cross-sites data sharing and retrieval use case in the context of the NeuroLOG project.

3.3.1 Use case: secured sharing of datasets through decentralized RBAC

Let us consider three sites A, B and C participating in the federation, given a collaborative study initiated by site A, the goal of site B is to share a set of owned data with partners involved in that study. Various actions are involved in this secured data sharing and a minimum of coordination is needed to achieve this goal, such as agreeing on the name

of a collaboration role. Figure 3.2 illustrates the interactions between administrators and site services (data storage/retrieval and access control services), and between site services and the coordination registry.

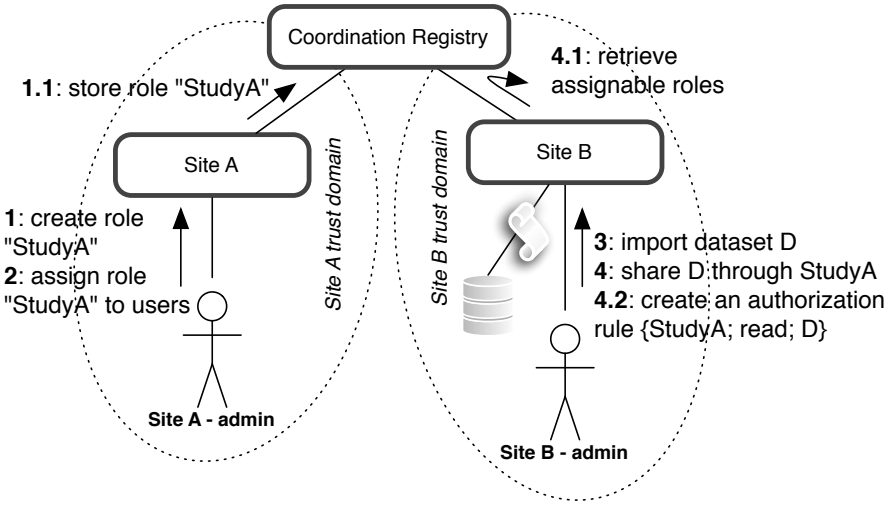

Figure 3.2: Activities involved in coherently sharing data

1. The administrator of *Site A* creates a new role named *StudyA* corresponding to the multi-centric study, to which datasets intended to be shared can be attached (step 1).

2. The administrator shares effectively the study by:

   (a) declaring that role on the Registry, associated to its initiator site (step 1.1);

   (b) potentially granting permissions onto hosted data for this shared role through the local access control policy (study read permission, for instance).

3. The administrator registers any user (local or foreign) participating in the study by assigning her the role previously created (step 2). If the user is unknown from the site, she is registered with her distinguished name (DN of her certificate) after validation of her certificate signature chain.

4. Finally, after importation of dataset *D* within *Site B* (step 3), the administrator of *Site B* is able to decide to share *D* with participants in *StudyA* (step 4), involving the retrieval of assignable roles for data access control (step 4.1) and the creation of a local authorization rule (step 4.2).

Let us now consider a user of *Site C*, involved in *StudyA* who needs to retrieve a shared data hosted by *Site B*. Figure 3.3 presents how the access control is performed thanks to the delegation of roles checking.

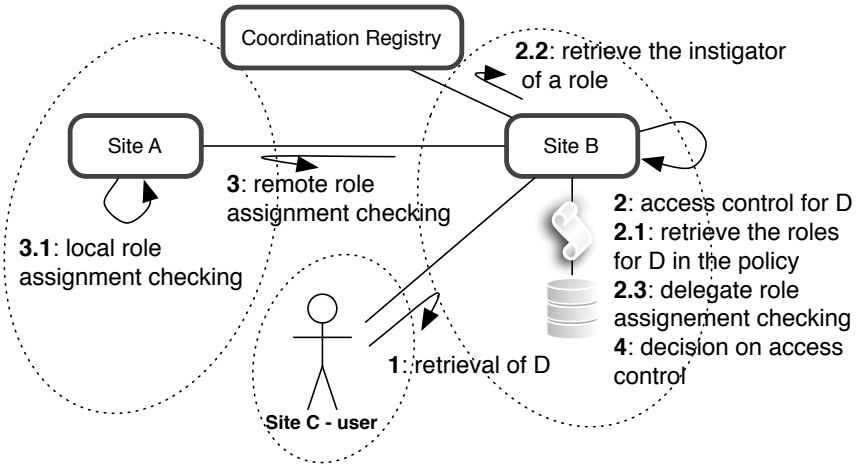

Figure 3.3: Decentralized access control

1. The user from *Site C* requests from *Site B* the retrieval of dataset *D* (step 1);

2. *Site B* controls the permissions of the user over *D* (step2) by:

   (a) retrieving in the local access control policy, the collection of roles attached to *D* (step 2.1);

   (b) retrieving the initiator sites for the resulting collection of roles (step 2.2);

   (c) delegating to each initiator site, the checking of role assignment to the user (step 3), in other words "does the user own the role *StudyA* ?" (step 3.1);

3. If one of the collaborating sites validates the role assignment checking, then the access control is performed regarding the permission associated to the data and the role.

3.3.2 Implementation

The security model exposed in this chapter has been implemented within the core of the NeuroLOG middleware. The remainder of this section details how our security concerns have been operationalized. The NeuroLOG platform involves a *registry* dedicated to the coordination of the platform, multiple intercommunicating *site servers* which are implementing most of the middleware functionality and distributed *clients* from which users authenticate and connect to the system. To ensure scalability and sites independence, the administration of the platform is distributed and handled site-wise. In particular, sites are responsible for (i) granting access to the platform for their users and (ii) managing a local access control policy over hosted resources.

3.3.2.1 Middleware and Application services

Figure 3.4 illustrates the main software components involved in the NeuroLOG middleware and where have been implemented the distributed access control. A NeuroLOG user is able, through its *Client* application (and a dedicated graphical user interface), to perform queries against the *Relational data manager* on its site server. It provides a unified view over the distributed and heterogeneous data published by the other collaborating sites through the commercial *Data Federator* integration engine. As it has been introduced in section 3.1.3 relational (or semantic) data are considered in this platform as public whereas raw files are considered as sensitive and their access need to be strictly controlled. End-users are able to retrieve either local files or remote files through secured communication channels. For each participating site, sensitive raw files are accessed by a dedicated *Raw data manager* which is responsible for access control through a policy enforcement point (*PEP*). Then the distributed access control is performed by the *Site manager* through a dedicated policy decision point (*PDP*) as described in section 3.3.1. Finally, based on the access control decision, raw files might be directly communicated to the end-user.

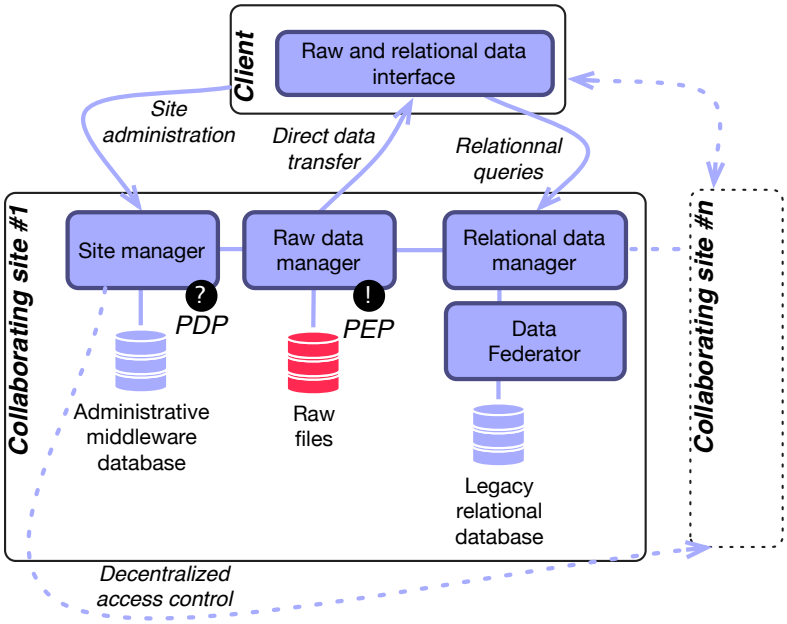

Figure 3.4: Policy decision and enforcement points in the NeuroLOG middleware.

The software stack uses standard frameworks such as MySQL database, EJB3/Hibernate API to handle object/relational mappings, and JAX-WS web-services to expose application and middleware services.

Transverse security concerns are addressed through OpenSSL, Java Secure Socket Extension, Java Public Key Infrastructure and Java Cryptography Architecture. OpenSSL is involved in the setup of the certification authorities as mentioned in section 3.2.1 and in the delivering and signing of X509 certificates. Java Security frameworks ease the estab-

lishment of secured communication channels through SSL.

3.3.2.2 Authentication service

Through the establishment of trust chains over the federation, each participant can communicate through SSL channels. We had to cope with an authentication issue, related to using high-level protocols (*e.g.* SOAP protocols) over pre-established SSL connections. Once an SSL communication is established, those protocols do not provide control, at server side, over the identity of the caller emitting high-level protocol messages. For instance, while opening an SSL channel with a Site Server, the identity of a user is validated through SSL handshaking. However, several clients may similarly connect to the same server. Thus, the Site Server receives requests without knowing who effectively performed each of them. This issue brings a major security requirement at application level that is authentication (re-identification) of the caller of a specific operation in the service.

Regarding web services, the security context depends on the service container. A web server transporting messages over HTTPS will benefit from SSL handshaking to identify the caller. However, with simple stand alone web services, re-identifying the user connected is not possible and an authentication token is needed in the WS operation call to perform access control at the service level. Alternatively, web services can be hosted in a container such as Apache Tomcat. In this case, the container provides the functionality needed to retrieve the SSL identity of any caller, thus enabling re-identification without any additional operation.

3.3.2.3 Authorization services

The authorization component makes use of the same software stack. To manage the sharing of roles over the federation, the coordination registry handles the persistency of roles and their associated initiator sites. Authorization services deployed in the Site Servers handle the persistency of users and their assigned roles, and authorization rules. In order to collaborate with their peers, as proposed in the decentralized access control use case, they expose a service responsible for validating that a user owns a role. They finally expose an access control service which asserts that given (i) an authentication token, (ii) a resource identifier, and (iii) a permission, the identity extracted from the token owns the permission over the requested resource.

The flexibility of the proposed system allowed to extend this distributed access control policy to also protect the invocation of data processing services. This requirement has been gathered through the NeuroLOG community. It is related to the social motivations preventing sharing introduced in section 3.1.1. As an example, partners developing image processing tools also wanted to protect the access over tools whose algorithm were not fully exploited, in terms of scientific publication. During their deployment as web service through the JGASW [Rojas Balderrama et al., 2010] web service wrapper, data processing tools have been instrumented with a policy enforcement point preventing for

non-authorized invocations.

3.4 Discussion and conclusion

Considering the analysis of Sinnott and and co-workers in [Sinnott et al., 2008], we proposed in this chapter an hybrid security model dedicated to health-oriented distributed platforms. Indeed, the proposed security model allows for distributed access control while still considering prevailing local policies. In addition, the coordination registry eases the agreement on roles at the scale of the federation, and thus copes with the disadvantage of fully decentralized VO role management systems in which roles may be scattered over participating sites.

Compared to VOMS, the NeuroLOG platform is based on a lighter infrastructure and the certificates are decoupled from the authorization framework, thus enabling more dynamic permission control. In addition, with VOMS, user, group and role management are centralized at the VOMS server level. Conversely, NeuroLOG authorization mechanisms are decentralized over the participating sites in the sense that (i) a site administrator registers her users by providing them a valid certificate, and (ii) authorizing users, either local or remote, through role assignments.

The Shibboleth approach is not really suitable in the context of the NeuroLOG platform (R_2 and R_3). Indeed, despite a similar authentication pattern, the attributes management (roles in our terminology) is quite different. A key contribution in our work is the role management, under responsibility of the site which initiates a multi-centric study, and exclusively that site. On the contrary, with the Shibboleth approach, data sharing in the context of multi-centric clinical studies, would not be solely under the responsibility of the study initiator site but distributed to all administrators of participating sites, which serves badly the required autonomy (R_4) of participating sites.

PERMIS enables the definition of various security policies, involving distribution of attributes in site-wise repositories whereas our approach propose a non-trivial security policy where federation-wide unique roles are recognized and coherently managed in the federation. In the NeuroLOG platform, local policies are understandable all over the federation through availability and unicity of federation-wide roles. Moreover we propose a compromise between fine-grained and coarse-grained policies with authorization rules scaled to clinical studies and their related datasets.

Rajasekaran et al. identified in [Rajasekaran et al., 2008] similar security requirements in the design of the @neurIST platform, aimed at improving research and clinical care of cerebral aneurisms. Solutions appear to be close to our approach but at this stage, available documents do not provide sufficient information to compare the role management at the scale of the virtual organization and how data sharing can be performed in the context of multi-centric studies.

We summarize the positioning of our approach with respect VOMS, Shibboleth and PERMIS as follows. On the one hand, VOMS relies on a centralized management of users, groups and roles. On the other hand, PERMIS provides a fully distributed management

of authorization policies with roles, uniquely defined at the scale of a site. We propose a hybrid approach in which the management of roles is centralized at the scale of the federation. This choice is a trade-off which eases the agreement on roles at the scale of the federation, and avoids complex mappings between site-specific roles needed if each site deploys a prevailing local PERMIS authorization framework. Moreover, in the context of a multi-centric study, Shibboleth would hamper the autonomy property of participating sites since data sharing would be under the responsibility of all data-providers. Our hybrid model allows to have a single study coordinating site which manages users participating in that study, and at the same time allows each data provider to control the access over the data they publish through the federation-wide roles.

Providing a security infrastructure for medical applications running in a wide-scale, open grid infrastructure is difficult due to (i) the sensitivity of medical information but also (ii) the conflicting requirements of medical users to be able, at the same time, to share data and to still control their data resources. The foundational security layers integrated in the lower stack of most grid middlewares is addressing the first point but the second one requires elaborated and domain-specific security control policies.

In this chapter, we proposed a distributed security framework that addresses neuroscientists needs for enabling multi-centric studies by federating existing, heterogeneous site environments. The security layer is founded on state of the art security tools. In addition, we propose a distributed access control management policy that enables data sharing while respecting local data access policies (thesis research question RQ_2). The proposed infrastructure enables autonomous sites operation and it does not require a centralized administration. This distributed security policy is validated through an implementation ; practical set up problems related to client-server communications are taken into account.

This work addresses the access control over distributed raw data files and data processing services. It opens interesting perspectives since valuable life-science data relies more and more on complex and rich data representations such as knowledge graphs. Indeed, life-science data providers would be open to share their domain models (ontologies) but would be much more reluctant to share their research semantic data (ontology instances/individuals) without a strong access control. A lot of research activities are currently being conducted to address the access control over semantic data [Flouris et al., 2010], [Sacco et al., 2011]. Context-aware access control for distributed semantic data stores [Costabello et al., 2012] would be particularly relevant in the context of federated semantic queries (chapter 4) addressing distributed health data shared through linked data principles. Providing a flexible enough security framework still coping with the autonomy of semantic health data providers is an interesting perspective to foster the adoption of semantic web and knowledge-oriented technologies by clinical research communities.

Key Points

- *Even if foundational security framework have been available for a long time in distributed computing infrastructures, they do not suit application-level security requirements in the context of life-science resource sharing.*
- *Life-science resource sharing faces two partly conflicting concerns: setting-up collaborative multi-centric studies while still letting data providers operate local access control policies.*
- *Decentralized access control policies decoupling (i) the assignment of roles to users and (ii) the assignment of permissions to resources through roles, allows for facing secured enough life-science data sharing.*
- *Similar security requirements should be considered to address secured collaboration involving semantic data representations in collaborative life-science platforms.*

CHAPTER 4

Semantic data and query distribution

Contents

4.1 Introduction

Aimed at easing the setup of large scale scientific collaborations, collaborative life-science platforms provide data sharing and processing facilities while still considering partners as autonomous data providers. Preserving the autonomy of participating organizations, as introduced in section 2.1.3.2, is a prime-order constraint since life-science data are hardly relocatable for ethical or legal constraints, and life-science data providers must keep the control over their owned data. Such constraint strongly motivates for handling distributed querying over multiple data sources.

Through *materialized* or *virtualized* approaches, data integration systems (introduced in section 2.1.3) are used to coherently provide a uniform access over these distributed data sources. Materialized approaches consist in (i) extracting data from distributed data sources, (ii) possibly performing data transformations and finally (iii) populating a homogeneous and centralized data warehouse. These materialized (or *data warehousing*) approaches are not suitable in the context of autonomous data sources managing possibly sensitive data, hardly relocatable for ethical or legal reasons. An alternative solution consists in considering virtualized (or *federated*) approaches which allow pushing queries

over distributed data sources, without requiring for prior data migration, through Distributed Query Processing (DQP). Query results are finally integrated at querier side.

In the context of virtualized data integration, data may be partitioned with or without constraints on their nature. Two distribution setups may be envisaged: a first one in which each data source is specialized for a specific kind of data, and a second one in which no assumption can be made on the content of data sources. This second setup is the more appropriate to preserve the autonomy of data sources participating into life-science collaborative platforms. Moreover, this usage scenario is relevant in the context of collaborative platforms allowing semantic data sharing through a common ontology. This imply that all data sources possibly host any kind of data represented through the ontology, and consequently no assumption can be made on data sources content. This hypothesis on data partitioning requires "transparent" data federation strategies coping with the lack of prior knowledge on data sources content.

Moreover, we assist nowadays to a continuous increase in the number of initiatives aimed at publishing on the web open databases through the Linked Data principles [Bizer et al., 2009a]. These published data are generally semantically described with ontologies and are published through standard semantic data representations (OWL/RDF(S)). Collaborative life-science platforms being also supported by domain ontologies, there is an opportunity to benefit from knowledge exchanges in both ways: a collaborative life-science platform could benefit from "open" knowledge originating from Linked Open Data sources, and conversely publishable platform data or models as Linked Open Data, would enrich the Linked Open Data sources available over the Web.

Semantic data representations and techniques are well spread into both Linked Data initiatives, and collaborative life-science platforms. But as discussed in section 2.4.2, few of the recent large scale collaborative life-science platforms benefit from semantic data representation and querying when performing data integration, and most of them only rely on relational data federation. This lack of ontology exploitation in data integration thus prevents from performing highly expressive queries and automated reasoning.

Recently, the W3C published a pre-recommendation[1] [Prud'hommeaux et al., 2011] addressing the federated querying of distributed knowledge bases through a set of language extensions for SPARQL 1.1, the SERVICE and VALUES clauses. But it requires prior knowledge on how to distribute the subsets of the query to particular semantic data providers. For instance the following query $Q_{4.1}$ searches for person metadata distributed over two knowledge bases site1 and site2, where data is partitioned as follows. The first data source hosts foaf:name properties and the second one hosts dbpedia:birthDate properties. This particular setup requires for the query designer to have this prior knowledge on which site hosts which kind of properties in order to drive the semantic query evaluator through the SERVICE clauses.

In the context of life-science collaborative platforms hosting homogeneous semantic annotations, *i.e.* data semantically described through a common ontology, it is not re-

[1] http://www.w3.org/TR/sparql11-federated-query

Listing 4.1: SPARQL 1.1 federated query performed against two distributed SPARQL Endpoints

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dbpedia: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?x ?name ?date WHERE {
        SERVICE <http://people.site1.org> {
                ?x foaf:name ?name .
                FILTER (CONTAINS (?name, 'Bobby␣A'))
        }
        SERVICE <http://people.site2.org> {
                ?x dbpedia:birthDate ?date
        }
}
```

alistic to let the query designer decide on which site a sub-set of the query should be pushed. Moreover, considering the previous example, we could easily envisage that site2 also hosts foaf:name properties. The query then must be adapted in order not to miss any person described through site2. More generally, this approach is not suitable in the context of dynamic knowledge base federations. If we envisage a new site joining the federation, the pre-designed SPARQL queries must be consequently adapted to take it into account. In the same way, an unreachable partner site would lead to a necessary adaptation of the federated queries.

Several research activities have been recently conducted to address transparent semantic data federation [Quilitz and Leser, 2008, Görlitz and Staab, 2011, Langegger et al., 2008, Schwarte et al., 2011, Ladwig and Tran, 2010]. They have been more precisely discussed in section 2.3.2. They all highlight the cost of performing distributed query processing over geographically spread semantic data sources. Through performance-oriented approaches (possibly involving parallelization), they tend to minimize the amount of network communication needed to join data coming from distributed data sources. However these approaches generally only cover a limited subset of the SPARQL querying language, thus preventing their adoption in the context of demanding collaborative life-science platforms.

Since collaborative life-science platforms strongly rely on autonomous data source providers, their underlying data integration systems generally face the issues of federating heterogeneous data sources (heterogeneity challenge described in section 2.1.3.2). We propose in this chapter to (i) rely on abstract knowledge graphs as a unifying model to represent, query and reason on semantic data while still coping with possible heterogeneous data structures, and (ii) a set of optimization strategies to implement efficient distributed query processing over geographically spread abstract knowledge graphs. The remainder of this chapter, addressing both research questions RQ_1 and RQ_3, proposes a set of static and dynamic optimizations in section 4.2 and a first evaluation of the proposed algorithms in section 4.3. Results are finally summarized and discussed in sec-

tion 4.4.

4.2 Strategies for semantic query distribution

Based on the notion of abstract knowledge graphs, this section proposes strategies to statically and dynamically optimize the distributed query processing for geographically spread semantic data sources.

4.2.1 Abstract knowledge graphs

Abstract knowledge graphs have been introduced in [Corby and Faron-Zucker, 2010] through the KGRAM framework. They result from an initiative [Dieng-Kuntz and Corby, 2005] aimed at easing the design of semantic web applications through conceptual graphs (CGs) [Sowa, 1984]. They are particularly well suited in the context of collaborative life-science platforms since (i) they allow representing and querying semantic data (such as standard RDF data and RDFS controlled vocabularies) and (ii) they allow addressing structural data source heterogeneity as soon as a graph-based view on the underlying data structure can be provided.

In spite of subtle differences [Dieng-Kuntz and Corby, 2005], the mapping between RDF data and Abstract Knowledge Graphs was almost straightforward:

RDF(S)	*Abstract Knowledge Graphs*
RDFS class	Concept as a node
RDF(S) property	Relations as directed labelled edges
RDF graph	Abstract Knowledge Graph
RDF resource (subject or object)	Node
RDF predicate	Node
RDF triple	Directed labelled edge

Table 4.1: Mapping between RDF(S) and abstract knowledge graph representations

The Knowledge Graph Abstract Machine [Corby and Faron-Zucker, 2010] (KGRAM) is a framework aimed at representing, querying and reasoning over semantic data represented as abstract knowledge graphs[2]. KGRAM interprets an abstract language which generalizes SPARQL 1.1, thus benefiting from its high expressivity (aggregate functions, subqueries, negations and property paths).

To mediate several, possibly heterogeneous data sources, KGRAM introduces a set of abstract operators manipulating abstract graph data structures (abstract *Nodes* and *Edges* forming abstract *Graphs*). *Graphs* are navigated through *Producers* responsible for the iteration over *Nodes* and *Edges*, thus acting as graph mediators. For each kind of data source, a specific *Producer* is needed that abstracts its representation provided that it is able to produce a graph view of hosted data.

[2]KGRAM is part of the Corese [Corby and Faron-Zucker, 2010] Semantic Web Factory.

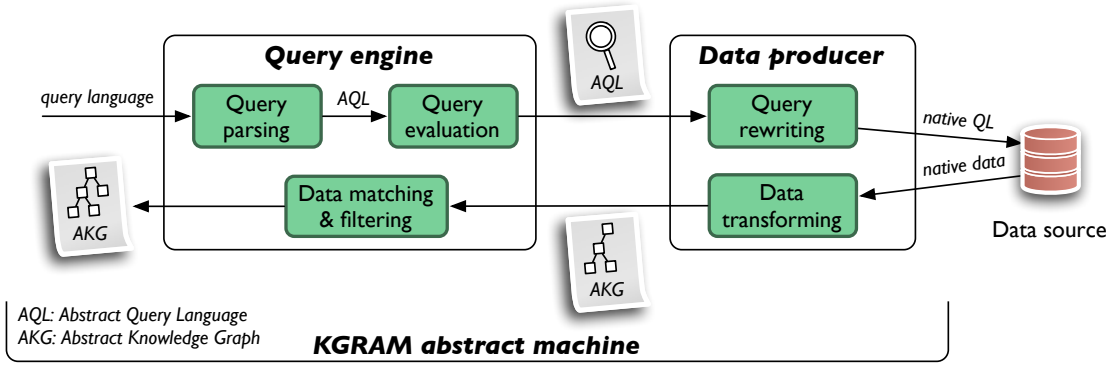

Figure 4.1: Graph-based querying with KGRAM.

Figure 4.1 illustrates graph-based querying with KGRAM. After being parsed, a query is transformed through an abstract query language (*AQL*) generalizing SPARQL 1.1. The query evaluation consists then in sending *AQL* queries to data *Producers*. Queries are rewritten through the native query language to address the specificities of a data source. When native data is returned, the *Producer* transforms it back into its abstract knowledge graph representation (*AKG*) to the query engine. Data is finally matched and filtered by the query engine and returns an *AKG* as result.

Abstract knowledge graphs, and graph *Producers* provide thus a well adapted model to study the performance issues of distributed query processing over autonomous (distributed and heterogeneous) data sources, and to adapt to their potential structural heterogeneity.

4.2.2 Distributed Query Processing principles

We consider semantic data as knowledge graphs, fragmented over multiple semantic data stores. The distributed query processing (DQP) of a semantic query consists in identifying graph patterns into a virtually unified graph, based on the appropriate sub-querying of multi-sources graphs. We adopted a federated approach to achieve querying over distributed knowledge bases while coping with site autonomy and scalability constraints. Federated approaches rely on a central federator component, and a set of federated data providers. From a unique query, the federator is responsible for the coherent sub-querying of the federated data providers and for unifying all results into a global result set.

Based on the sample query $Q_{4.2}$, this section provides both an informal description of the distribution principles and a naive algorithm to spread a SPARQL query over a set of distributed semantic data stores. Results are retrieved as if all resulting data were virtually gathered into a centralized knowledge base.

The sample query $Q_{4.2}$ aims at searching for people (designated by the two variables ?x and ?z) who share a common friend (variable ?y). The body of the query is composed

Listing 4.2: Sample SPARQL query distributed over remote knowledge bases

```
1 PREFIX foaf: <http://xmlns.com/foaf/0.1/>
2 SELECT DISTINCT ?x ?z WHERE {
3         ?x foaf:knows ?y .
4         ?y foaf:knows ?z .
5 }
```

by two edge requests (line 3 and line 4) forming a Basic Graph Pattern (i.e. a set of triple patterns). Graph edges are composed of a *subject* node, an *object* node, and a *predicate* linking together the *subject* to the *object*. For the ?x foaf:knows ?y edge request, the subject ?x and the object ?y are linked together through the predicate foaf:knows. We consider the case where friendship relations are fragmented over distributed knowledge bases. Then, several evaluation strategies can be envisaged:

(i) *Sequential evaluation*, drafted in algorithm 1, consists basically in iterating over each edge request ; for each one, the query processor iterates over each distributed knowledge base to push the current edge request. This strategy implies to wait for the response of each knowledge base at each iteration step.

(ii) *Fine-grained parallel evaluation* consists in exploiting the parallel querying of distributed knowledge bases and wait, through a synchronization barrier at each edge iteration, for the complete set of results.

(iii) *Coarse-grained parallel evaluation* consists in limiting the number of remote edge requests by grouping them together through independent basic graph patterns and enabling a parallel evaluation for each independent basic graph pattern. This strategy could only be considered if triples from the same independent graph pattern are not spread over distributed knowledge bases. This strategy aims at limiting the number of remote queries to federated endpoints.

(iv) *Pipelined evaluation* consists, for a given edge request, in exploiting partial results as soon as they are available into the remaining edge requests. Although promising, this strategy is complex since it requires a completely asynchronous evaluation engine.

In a first step, let us consider the sequential distributed query processing of the query $Q_{4.2}$ as it is proposed in Algorithm 1. The federator first iterates over the two edge requests ?x foaf:knows ?y and ?y foaf:knows ?z, (line 1). Then, for the current edge (line 2), the federator iterates over the remote producers and performs a remote evaluation of the current edge (line 3). Results are finally merged into the federator (line 3). This proposed algorithm 1 could be easily enhanced by exploiting the parallelism of distributed data providers such as proposed in the fine-grained parallelism strategy *(ii)*.
The remainder of this chapter mainly focuses on fine-grained evaluation, suitable when data is scattered without any consideration of the data model (ontology or controlled

Algorithm 1: Sequencial distributed query processing, waiting after each remote invocation.

Data: $Producers$ the set of SPARQL endpoints,
$EdgeReq$ the set of edge requests forming the SPARQL query,
$scheduler$ a thread pool allowing for parallel executions.
Result: $Results$ the set of SPARQL results.

```
1 foreach (e ∈ EdgeReq) do
2     foreach (p ∈ Producers) do
3         Results ← p.getEdges(e) ;
```

vocabulary), and on coarse-grained parallel evaluation, suitable when data is partitioned considering several data models.

4.2.3 Query rewriting optimizations

Algorithm 1 illustrates the basic principle envisaged to transparently query distributed knowledge bases without specifying, into the query, any distribution directive so that no additional knowledge on the content of the distributed stores is required. The main drawback of this approach is that, for all federated endpoints, we need to send remote queries for all edge requests constituting the initial query. The number of matching edges returned to the federator might be high and they are finally joined or filtered by the federator itself. The main idea under the proposed optimization strategies consists in minimizing the amount of edges to be processed by the federator, and maximizing the processing performed by remote federated endpoint. To achieve distributed query processing of SPARQL queries, we consider the following four query rewriting strategies :

(i) Naive: consists in segmenting a global SPARQL SELECT query into a set of elementary SPARQL queries, one for each edge request. Since SELECT queries return results as a set of of pairs $\{variable, value\}$ (SPARQL RESULTS format[3]), it is preferable to generate CONSTRUCT queries because results are directly returned as RDF data. In that format, results do not require any transformation process[4] in order to be incorporated back into the federator.

(ii) Filter: consists in incorporating an applicable FILTER expression to a generated edge request, in order to delegate the filtering of irrelevant results directly to the remote federated endpoint.

(iii) Binding: consists in exploiting partial results gathered from previous edge requests. More precisely, the value of variables are exploited as soon as they are known into the generated edge requests. They become thus simpler since variables are replaced

[3] http://www.w3.org/TR/rdf-sparql-XMLres
[4] that would be needed if results were transferred as $\{variable, value\}$ pairs

by their corresponding values. They additionally reduce the amount of transferred results.

(iv) Full: consists in combining both the *filter* and the *binding* strategies.

4.2.3.1 Filter rewriting strategy

While the *naive* query rewriting strategy consists in simply encapsulating an edge request into a SPARQL CONSTRUCT query, the FILTER rewriting strategy consists in aggregating to the select query, an applicable FILTER expression which represents a value restriction for one or more variables. The FILTER expression is extracted from the global SPARQL query. The idea is the following: in order to prevent from transferring results that will be locally filtered, ultimately, by the federator, it should be more effective to filter results directly at the source, behind the federated endpoints.

For example, let us consider the following query $Q_{4.3}$ which aims at searching for persons whose name contains "Bobby A" and their birth date.

Listing 4.3: Full SPARQL query distributed over remote KGRAM endpoints

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dbpedia: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?x ?name ?date WHERE {
        ?x foaf:name ?name .
        ?x dbpedia:birthDate ?date .
        FILTER (CONTAINS (?name, 'Bobby␣A'))
}
```

This query is decomposed into two edge requests: ?x foaf:name ?name, and ?x dbpedia:birthDate ?date. Searching for the first edge would be traduced by the query $Q_{4.4}$ in the *naive* strategy and it would be traduced by the query $Q_{4.5}$ in the *filter* strategy which is much more effective as it will be shown in the experiment results.

Listing 4.4: Generated SPARQL query encapsulating a single edge request through the naive rewriting strategy

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
CONSTRUCT {?x foaf:name ?name} WHERE {
        ?x foaf:name ?name.
}
```

This naive query $Q_{4.4}$ will lead to the transfer of all foaf:name properties for all federated endpoints. This will impact both the network load and the federator since it will ultimately filter all irrelevant results. Query $Q_{4.5}$ allows for filtering directly all irrelevant results behind federated endpoints, and thus limit both the size of transferred results and the load of their processing by the federator.

Listing 4.5: Optimized SPARQL query encapsulating a single edge request through the filter rewriting strategy

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
CONSTRUCT {?x foaf:name ?name} WHERE {
        ?x foaf:name ?name.
        FILTER (CONTAINS (?name, 'Bobby␣A'))
}
```

4.2.3.2 Binding rewriting strategy

The VALUES keyword proposed by the W3C in the SARQL 1.1 federated querying prerecommendation allows the federator to constrain federated endpoints with already known results. Based on this idea, we propose to systematically and transparently exploit these intermediate results (i.e. mappings between variables and their associated values). The benefits are twofolds since (i) the query does not need to be adapted (no need to incorporate SERVICE or VALUES clauses) to take into account the partial results, (ii) the federated endpoint will process simpler edge requests, leading to less expensive joins, from the federator point of view, because of less results to be transferred.

Let us consider that during the evaluation of the same query $Q_{4.3}$, the federator has already performed the first edge request ?x foaf:name ?name and gathered a set of intermediate results. Let us then consider one of these results in the form of the following mappings: { ?x $\rightarrow$ http://dbpedia/bobbyauyeung ; ?name $\rightarrow$ Bobby Au-Yeung }. We propose to exploit these results to replace the variables by their associated values (already known through the evaluation of previous edge requests). The following query $Q_{4.6}$ illustrates an optimized encapsulated edge request exploiting an intermediate result. It enables faster execution by the federated endpoint since only one variable need to be searched for, and only one triple should be produced from its evaluation.

Listing 4.6: Optimized SPARQL query encapsulating a single edge request through the binding rewriting strategy

```
PREFIX dbpedia: <http://dbpedia.org/ontology/>
CONSTRUCT {<http://dbpedia/bobbyauyeung> dbpedia:birthDate ?date} WHERE {
        <http://dbpedia/bobbyauyeung> dbpedia:birthDate ?date
}
```

The experiments presented in section 4.3 will show that the global query evaluation time is drastically reduced when combining these two query rewriting strategies.

4.2.3.3 Data source selection

In our work, KGRAM is provided with a cache index, dynamically created with SPARQL ASK queries. This cache mechanism prevents from unnecessary communications. For

each KGRAM EDGE request, this index stores the identified data sources providing candidate edges and thus potentially contributing to the result set. By using its index, KGRAM is able to send remote EDGE requests only to data sources hosting candidates. This strategy answers Linked Data querying scenarios where remote data sources and their associated producers are specialized into one kind of data. The proposed cache index thus allows saving the cost of unnecessary remote communications.

4.2.3.4 Coarse-grained parallelism from data source selection

As it has been drafted in section 4.2.2 point *(iii)*, the main idea consists in grouping together edge requests into a single sub-query that will be performed against a single source. This is possible if candidate edges are not spread over multiple data sources. Data source selection indexes, introduced in the previous paragraph, are used to determine the edge requests to be grouped together, based on the content of data sources. Finally, through query rewriting, a new SPARQL 1.1 query is generated that includes SERVICE clauses to group, when possible, edge requests.

Coarse-grained parallelism is achieved through the following steps:

Step 1. *Indices initializations*. This first step consists in populating two indexes needed to determine which edge request can be performed against a single data source. $idxEdgeSrc$, implemented with a {key,value} map, associates to each edge request a set of data sources hosting edge candidates and $idxSrcEdge$, similarly implemented, associates to each data source, edge requests which are exclusively hosted by this data source. This second index is used as input to generate a set of SPARQL 1.1 Service clauses.

Step 2. *Service clauses generation*. Service clauses are generated by simply navigating the $idxSrcEdge$ index and constructing a Service clause per indexed data source (the keys of the index).

Step 3. *Query rewriting*. Finally, the query rewriting step consists in, from the initial query, distinguishing the edge requests that will be included in Service clauses, from the edge requests that must remain outside Service clause (because possible edge candidates are spread over multiple data sources). Then the new query is assembled by re-associating the ungrouped edge requests and the edge requests grouped through Service clauses.

Sample rewritten query The following paragraph illustrates the coarse-grained parallelism through a sample SPARQL query ($Q_{4.7}$) evaluated in a distributed setup. We consider in this example two knowledge bases. The first one exposes DBpedia [Bizer et al., 2009b] statements though the sample http://dbpedia endpoint, and the second one exposes NeuroLOG statements through the sample http://neurolog endpoint.

Listing 4.7: Initial SPARQL query distributed over two remote KGRAM endpoints.

```
1 PREFIX foaf: <http://xmlns.com/foaf/0.1/>
2 PREFIX dbpedia: <http://dbpedia.org/ontology/>
3 PREFIX linguistic−expression: <http://www.irisa.fr/visages/linguistic−expression−owl−lite.owl#>
4 SELECT DISTINCT ?x ?y ?name ?date WHERE {
5         ?x foaf:name ?name .
6         ?x dbpedia:birthDate ?date .
7         ?y linguistic−expression:has−for−name ?z .
8         FILTER (CONTAINS (?name, 'Bob'))
9 }
```

The initial query $Q_{4.7}$ involves three edge requests: foaf:name (line 5), dbpedia:birthDate (line 6), and linguistic-expression:has-for-name (line 7). The data source selection strategy proposed in section 4.2.3.3 already prevents unnecessary requests. It is indeed useless to send DBpedia edge requests to the NeuroLOG endpoint, and conversely. Moreover the proposed coarse-grained parallelism strategy allows grouping DBpedia edge requests into a single "Service" subquery, and NeuroLOG edge requests into a similar "Service" subquery. The following query $Q_{4.8}$ illustrates the rewritten query including Service clauses.

Listing 4.8: Automatically rewritten SPARQL query using SPARQL 1.1 Service clauses.

```
1 PREFIX foaf: <http://xmlns.com/foaf/0.1/>
2 PREFIX dbpedia: <http://dbpedia.org/ontology/>
3 PREFIX linguistic−expression: <http://www.irisa.fr/visages/linguistic−expression−owl−lite.owl#>
4 SELECT DISTINCT ?x ?y ?name ?date WHERE {
5         SERVICE <http://dbpedia> {
6                 ?x foaf:name ?name .
7                 ?x dbpedia:birthDate ?date .
8                 FILTER (CONTAINS (?name, 'Bob'))
9         }
10        SERVICE <http://neurolog> {
11                ?y linguistic−expression:has−for−name ?z .
12        }
13 }
```

By grouping edge requests into SPARQL Service clauses, the evaluation of this optimized query saves the communication of some intermediate results. Indeed, whereas three distributed joins are needed when evaluating $Q_{4.7}$, only two are needed for the evaluation of $Q_{4.8}$. The join between lines 6 and 7 of $Q_{4.8}$ is performed by the http://dbpedia endpoint.

Edge grouping algorithms The following paragraphs detail the three main steps required to achieve the coarse-grained parallelism strategy. Algorithm 2 describes how is built a first index associating to each edge request, the possible data source hosting the

corresponding edge candidates. In the previous example (query $Q_{4.7}$), we would obtain the following index :

$$
\begin{aligned}
idxEdgeSrc \; = \{ \;& \text{foaf:name} \rightarrow \{\text{http://dbpedia}\}; \\
& \text{dbpedia:birthDate} \rightarrow \{\text{http://dbpedia}\}; \\
& \text{linguistic-expression:has-for-name} \rightarrow \{\text{http://neurolog}\} \; \}
\end{aligned}
$$

The algorithm consists in iterating over all expressions forming the initial query (line 1). In the case of an OPTIONAL clause (line 2), a recursive call is performed to index all optional edge requests (line 5). In the case of a UNION clause, similar recursive calls are performed to index the content of the UNION arguments (lines 8 and 9). Finally when the current expression is an edge request (line 10), SPARQL ASK queries are sent in parallel to remote producers to determine the target data source hosting edge request candidates (line 12).

Algorithm 2: initIdxEdgeSrc(exp**,** $idxEdgeSrc$**)** initializes an index associating an edge request to a set of data sources, based on SPARQL ASK queries.

Data: $idxEdgeSrc$ an empty index storing for each edge request, the set of data sources hosting edge candidates.
Result: $idxEdgeSrc$ the populated edge to sources index.

```
1  foreach (exp ∈ Query) do
2      if (exp.isOptional()) then
3          option ← exp ;                    // adds exp to the option set (+=)
4          foreach (optionalExp ∈ option) do
5              initIdxEdgeSrc(optionalExp, idxEdgeSrc) ;          // recursion
6      else if (exp.isUnion()) then
7          union ← exp ;
8          initIdxEdgeSrc(union.getArg(0), idxEdgeSrc) ;          // recursion
9          initIdxEdgeSrc(union.getArg(1), idxEdgeSrc) ;          // recursion
10     else if (exp.isTriple()) then
11         foreach (p ∈ Producers) do in parallel
12             scheduler.submit(p.askEdge(exp)) ;
13         wait for scheduler ;
```

The resulting index is then used as input to initialize the second index $idxSrcEdge$ more usable in the context of SPARQL SERVICE clauses. Indeed, algorithm 3 aims at reversing, somehow, the previous index, to provide a description of the content of data sources, sorted by data sources. If we come back to the previous example, algorithm 3 would lead to the following index :

$$idxSrcEdge = \{ \begin{array}{l} \text{http://dbpedia} \rightarrow \{\text{foaf:name ; dbpedia:birthDate}\} ; \\ \text{http://neurolog} \rightarrow \{\text{linguistic-expression:has-for-name}\} \} \end{array}$$

Algorithm 3 consists in iterating over the keys of the first index, *idxEdgeSrc* (line 1). For each edge, the optimizer selects exclusively edges hosted in a single data source (line 3). For each selected single source edge, it is added to the set of edges associated to the corresponding data source (lines 6 or 10).

Algorithm 3: Initialization of an index associating data sources to a set of hosted edges. The initialization is based on the reversed index previously introduced in algorithm 2

Data: $idxEdgeSrc$ a populated index associating for each edge request, the set of data sources hosting edge candidates,
$idxSrcEdge$ an empty index associating for each data source, the set of hosted edges.

Result: a populated sources to edges index.

```
1  foreach (edge ∈ idxEdgeSrc) do
2      sourceSet ← idxEdgeSrc.get(edge) ;
3      if (sourceSet.size() = 1) then
4          url ← sourceSet.get(0) ;
5          if (idxSrcEdge.containsKey(url)) then
6              idxSrcEdge.get(url).add(edge) ;
7          else
8              edges ← new Set<Edges>() ;
9              edges.add(edge) ;
10             idxSrcEdge.put(url, edges);
```

Once these two indices have been initialized, thus completing the **step 1** of the coarse-grain parallelism strategy, the optimizer can now generate a list of SPARQL SERVICE clauses by exploiting the $idxSrcEdge$ index. Algorithm 4 details how these SERVICE clauses are constructed, thus fulfilling **step 2** of the strategy.

Algorithm 4 iterates over all data sources indexed in $idxSrcEdge$ (line 1). For each one, all associated edge requests are retrieved and gathered (line 3) into a basic graph pattern (line 4). Applicable filters are extracted (line 5) from the initial query in the same way as presented above, in section 4.2.3.1. Finally, a SERVICE clause is generated (line 6) for each indexed data source.

Once the SERVICE clauses are available, the final query rewriting can be performed (**step 3**).

Algorithm 4: Generation of SPARQL SERVICE clauses, based on the $idxSrcEdge$ associating data sources and the corresponding hosted edges.

Data: $idxSrcEdge$ an empty index associating for each data source, the set of hosted edges,
$queryFilters$ the list of query filters.
Result: $services$ a set of SPARQL Service clauses.

```
1 foreach (url ∈ idxSrcEdge) do
2     bgp ← BasicGraphPattern.create() ;
3     triples ← idxSrcEdge.get(url) ;
4     bgp ← triples ;
5     bgp ← getApplicableFilters(queryFilters, bgp) ;
6     bgp ← Service.create(url, bgp) ;
```

Algorithm 5 describes this final query rewriting step. It consists in iterating over the expressions forming the initial query (line 1). In this first version of the algorithm, optional statements have not been addressed, and they are thus kept outside SERVICE clauses (line 3). Similarly to Algorithm 2, UNION clauses lead to two recursive calls (lines 8 and 9). Then, when the current expression is an edge, if it does not exist in the data source index ($idxSrcEdge$), it is kept outside the SERVICE clauses (line 13), otherwise it has to be integrated within SERVICE clauses (line 15). The query rewriting is finally performed from line 16 to line 21. SERVICE clauses (retrieved from line 18) are first added to the new query (line 19) and "non-Service" clauses are concatenated to the new query (line 21).

This coarse-grained parallel strategy is interesting because from very simple indices, it allows saving communication costs by grouping together query edge requests, if they are hosted by a single data source.

However, Algorithm 5 introduces combinatory complexity resulting from order issues when re-assembling the optimized query. We chose in this algorithm to first process "non-service" expressions, but we will see in the first experiments presented in section 4.3 that in some cases, this strategy can be particularly inefficient. Still, such coarse-grained parallelism opens new optimization perspectives in terms of cost based query planning through automated SERVICE clauses generation.

4.2.4 Federator parallelism optimizations

We have proposed in the previous section a set of distributed query processing optimizations based on query rewriting. We will describe now distributed query processing enhancements through parallelism exploitation in the context of remote data sources.

Algorithm 5: rewriteQueryWithServices(exp,$idxSrcEdge$)) introduces SPARQL 1.1 SERVICE clauses in a SPARQL query based on the previously introduced index.

Data: $idxSrcEdge$ the index associating for each data source, the hosted edges.
$query$, the initial query to be rewritten with SPARQL SERVICE clauses.
$toKeep$, an edge set to be kept inside SERVICE clauses.
$toDrop$, an edge set to be kept outside SERVICE clauses.
Result: a rewritten query with SPARQL SERVICE clauses.

```
1  foreach (exp ∈ query) do
2      if (exp.isOptional()) then
3          toDrop.add(exp) ;
4      else if (exp.isUnion()) then
5          union ← exp ;
6          union.set(0,rewriteQueryWithServices(union.getArg(0),idxSrcEdge)) ;
           // recursion
7          union.set(1,rewriteQueryWithServices(union.getArg(1),idxSrcEdge)) ;
           // recursion
8          toDrop.add(exp) ;
9      else if (exp.isTriple()) then
10         if (¬ existsInSourceIndex(exp, idxSrcEdge)) then
11             toDrop.add(exp) ;
12         else
13             toKeep.add(exp) ;
14     bgp ← BasicGraphPattern.create() ;
15     if (¬ toKeep.isEmpty()) then
16         serviceClauses ← getServiceClauses(idxSrcEdge) ;
17         bgp ← serviceClauses ;
18     if (¬ toDrop.isEmpty()) then
19         bgp ← toDrop ;
```

4.2.4.1 Parallel-wait strategy

Based on the fine-grained parallelism (strategy *(ii)* of section 4.2.2), the following algorithm 6 illustrates how to distribute the query over a set of federated knowledge bases by exploiting the parallelism of each remote producer. *Results* follow the "SPARQL Result" W3C recommendation[5] and are represented as a set of *Results*, each of them encompassing a set of *Mappings* between variables and values.

Algorithm 6: Fine-grained parallel distributed query processing, with an explicit wait condition.

Data: $Producers$ the set of SPARQL endpoints,
$EdgeReq$ the set of edge requests forming the SPARQL query,
$scheduler$ a thread pool allowing parallel execution.
Result: $Results$ the set of SPARQL results.

```
1 foreach (e ∈ EdgeReq) do
2     foreach (p ∈ Producers) do in parallel
3         scheduler.submit(p.getEdges(e)) ;
4     wait for scheduler ;
5     foreach (task ∈ scheduler.getFinished()) do
6         Results ← task.getResults() ;
```

The principle consists in iterating over each edge request forming the initial SPARQL query (line 1). Then, for each edge request, all federated SPARQL endpoints are queried concurrently (line 3). The federator then wait for all federated endpoints to finish through a synchronization barrier (line 4). Results are finally accumulated for the current edge request (lines 5 and 6) and the next edge request iteration can be processed (line 1). In the following section, we consider reducing the cost of the synchronized barrier by anticipating the post-processing of the results through a classical producers-consumer pattern.

4.2.4.2 Parallel-pipeline strategy

The following set of algorithms is a first step towards an asynchronous distributed evaluation of the SPARQL query. Since this work is based on the KGRAM abstract machine, we do not address a complete asynchronous query evaluation that would require an in-depth redesign of the core evaluation strategy. The main idea consists in enabling the federator to handle results as soon as they are available. Instead of post-processing the results when all endpoints have terminated, already known results can be post-processed sooner. The global evaluation time is expected to be reduced, in particular when endpoints are producing large amounts of intermediate results.

Figure 4.2 illustrates a fictive evaluation for a given edge request (from t_0 to t_3 or t_4, depending on the strategy) pushed to two federated endpoints, *Producer#1* and *Pro-*

[5] http://www.w3.org/TR/rdf-sparql-XMLres/

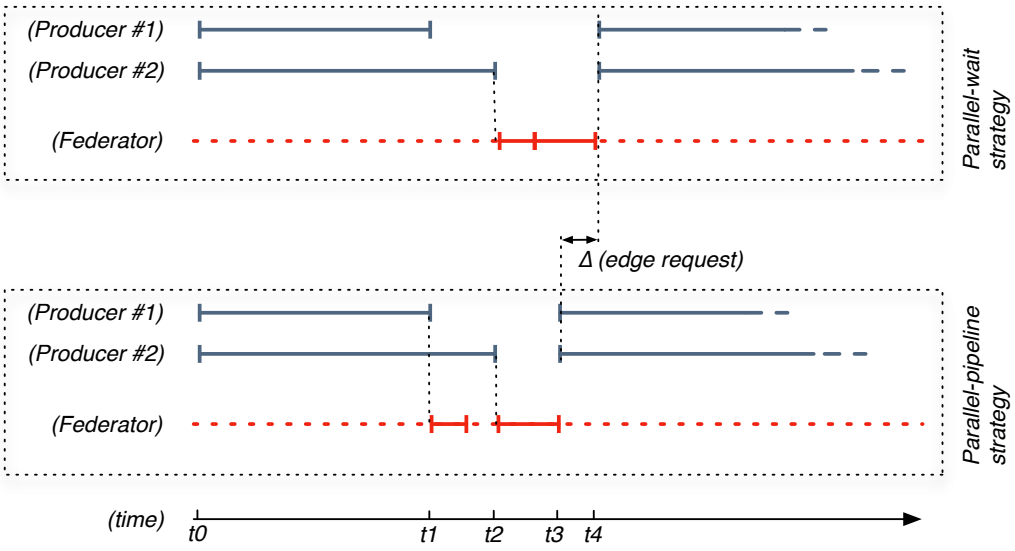

Figure 4.2: Distributed semantic query processing with KGRAM

ducer#2. The top rectangle sketches the query evaluation through the "parallel-wait" strategy whereas the bottom rectangle sketches the "parallel-pipeline". The first edge request is launched by the federator at time *t0* for the two strategies. For the "parallel-wait" strategy, the federator is waiting until the completion of the two remote producers (time *t2*). Then results are centrally post-processed from time *t2* to *t4* and the new edge request is launched at time *t4*. The "Parallel-pipeline" strategy is more effective since results begin to be post-processed at time *t1* as soon as they are available. Similarly, results produced by the second endpoint begin to be post-processed at time *t2* allowing the federator to launch the new edge request sooner, at time *t3*.

This strategy seems to be beneficial for the global query evaluation time in case the post-processing of the results is significant compared to the time spent to remotely process the query, and if one of the remote producers is penalizing for the rest of the federation due to lower CPU performance or network bandwidth.

Algorithms 7 and 8 illustrate the N-producers (federated endpoints) / 1-consumer (federator) pattern used to optimize the parallel evaluation. We removed the explicit wait condition present in the previous algorithm 6 (line 4) and replaced it by an implicit wait condition provided by a synchronized blocking "First-In-First-Out" queue (*syncQueue*). Indeed the queue provides blocking *put* and *take* operations (represented by the $\hookleftarrow\!\!\rho$ symbol). We also extended the queue with a blocking *next* (algorithm 9) and a non-blocking *hasNext* operations (algorithm 10). The blocking *next* operation aims at iterating over partial results and waits until all current results have been consumed. The *hasNext* operation checks if an endpoint is still producing results or if there are still results to be consumed from the queue.

Algorithm 7: Edge producer for the parallel-pipelined strategy

Data: $Producers$ the set of SPARQL endpoints,
$EdgeReq$ the set of edge requests forming the SPARQL query,
$syncQueue$ a synchronized FIFO queue allowing thread-safe put and take of objects.

Result: a synchronized blocking queue aiming at streaming results as soon as they are available to the edge consumer.

```
1 foreach (e ∈ EdgeReq) do
2     syncQueue ← new LinkedBlockingQueue() ;
3     foreach (p ∈ Producers) do
4         do in parallel
5             PartialRes ← p.getEdges(e) ;   // (costly) remote invocation
6             foreach (edge ∈ PartialRes) do
7                 syncQueue ↤ edge
8             syncQueue ↤ STOP
```

Algorithm 8: Edge consumer for the parallel-pipelined strategy

Data: $syncQueue$ the result edges stored in a synchronized FIFO queue allowing for thread-safe blocking ($\leftarrowtail$) put and take.

Result: $Results$ the set of SPARQL results.

```
1 while (syncQueue.hasNext()) do
2     edge ↤ syncQueue.next() ;
3     addToKgramGraph (edge) ;
```

The experiments presented in section 4.3 will show that this parallel-pipeline strategy has a positive effect on small knowledge bases, with queries producing a large amount of intermediate results. However the gain is not significant for large scale knowledge bases, except in the case of federated endpoints showing heterogeneous response times.

4.3 Distributed query processing performance and scalability evaluation

The following section presents some results on the evaluation of the previous strategies for performing distributed query processing over remote semantic data stores.

Implementation In KGRAM, briefly presented in section 4.2.1, the evaluation of SPARQL queries consists basically in searching for matching edges in a knowledge graph. *MetaProducers* have been introduced to handle several sources of edges, and thus

Algorithm 9: the *next()* operation aimed at retrieving the next available object in the synchronized queue, waiting if needed.

Data: $nbPendingTasks$ the number of started concurrent call to remoteProducers (SPARQL endpoints)
Result: $Results$ the set of SPARQL results.

```
1 res ↤ syncQueue.take() ; // gets and removes the head of the queue.
2 while (res instance of STOP) do
3     nbPendingTasks ← nbPendingTasks − 1 ;
4     res ↤ syncQueue.peek() ;              // get the head of the queue.
5     if (res instance of STOP) then
6         syncQueue.poll() ;              // removes the head of the queue.
```

enable for mash-up applications while querying several sources of linked data.

We propose a software extension for the KGRAM engine allowing for semantically querying remote federated KGRAM engines through web service endpoints. Figure 4.3 illustrates the main elements involved in the distributed semantic query processing.

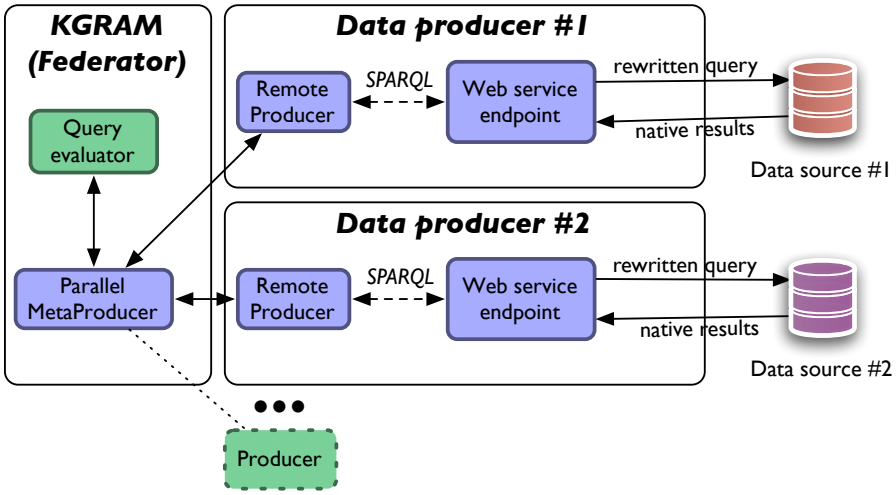

Figure 4.3: Semantic distributed query processing with KGRAM

We provided a web service implementation of the *RemoteProducer* of Figure 4.3 allowing to distribute semantic data stores. *Web service endpoints* are queried over the standard SOAP protocol. The *RemoteProducer* acts as a web service client and is responsible for *(i)* the encapsulation of a KGRAM edge request into a SPARQL CONSTRUCT query that will be pushed to the *Web service endpoint* and *(ii)* returning back the RDF results to the *ParallelMetaProducer*. Based on the algorithm 6, the *ParallelMetaProducer* is responsible for the parallelization of the edge requests over each federated *KGRAM endpoint*. A synchronization barrier makes the *ParallelMetaProducer* wait for the pending results which are

Algorithm 10: the *hasNext()* operation aimed at checking if the synchronized queue has a next element, even if empty.

Data: $syncQueue$ synchronized FIFO queue allowing for thread-safe blocking put and take results
Result: $Results$ the set of SPARQL results.

```
1  next ← queue.peek() ;
2  if (next = null) then
3      if (nbPendingTasks = 0) then return false ;
4      else return true ;
5  else
6      while (next instance of STOP) do
7          nbPendingTasks ← nbPendingTasks − 1 ;
8          syncQueue.poll() ;            // removes the head of the queue.
9          next ← queue.peek() ;
10         if ((next = null) AND (nbPendingTasks = 0)) then return false ;
11         else return true ;
```

finally returned back to the *QueryProcess* module and merged into the federated knowledge base. Several versions of the *ParallelMetaProducer* have been implemented to evaluate the impact of both the query rewriting and federator parallelism optimizations.

Experimental setup. The following experiments have been conducted through the experimental Grid'5000 infrastructure [Bolze et al., 2006]. Grid'5000 is a french scientific instrument dedicated to the study of large-scale parallel and distributed systems. Highly reconfigurable and controllable, Grid'5000 is considered as an experimental and research grid compared to "production" grids, where stability and availability are crucial. The Grid'5000 infrastructure interconnects 9 sites in France and 1 in Porto Alegre, Brazil. It makes available more than 7000 CPU cores for the research community. Sites are communicating through the RENATER network (10Gbps). Through an SSH frontend for each site and the OAR scheduler [Capit et al., 2005], users are able to select and reserve grid resources. The same OAR scheduler is also used to submit either interactive or batch jobs. The experiment setup generally consists in :

(i) reserving the resources from the "Suno" cluster of the Sophia Antipolis site. The "Suno" cluster is composed of 45 nodes, each node has 2 Intel CPUs (2.26GHz) with 4 cores per CPU, 32GB of RAM, and 519GB of disk storage. Generally one node is reserved for the federator and as many nodes as needed are reserved for the federated endpoints ;

(ii) for each federated endpoint, an Apache Tomcat server is installed and the KGRAM web-service endpoint is deployed ;

(iii) each federated endpoint is populated with the testbed knowledge base ;

(iv) from the federator node, a distributed SPARQL query is launched ;

(v) the logs produced are finally analyzed in order to measure the performance and the scalability of the overall setup.

4.3.1 Impact of the query rewriting optimizations

The following experiment measures the time needed to distribute two sample queries over two federated knowledge bases of DBpedia-person [Bizer et al., 2009b] statements, both forming a virtual single repository of 338K statements (169K statements each).

Material and methods. Table 4.2 measures the performance of the evaluation of the following SPARQL queries - "all persons whose name contains 'Bobby A' and their birth date" ($Q_{4.3}$), and "all persons whose name contains 'Bob', their birth date and optionally their birth place" ($Q_{4.9}$) - over two federated KGRAM endpoints. "Transferred triples" measure the number of intermediate results transferred to the federator by the federated endpoints. It is a good indicator to evaluate the cost of the distributed query processing. Finally, the DQP time measures the global time needed to (i) push the query to all federated KGRAM endpoints, (ii) wait for the results, and (iii) integrate them back into the federated knowledge base.

Listing 4.9: Less selective SPARQL query with an optional statement

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dbpedia: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?x ?name ?date ?place WHERE {
        ?x foaf:name ?name .
        ?x dbpedia:birthDate ?date .
        OPTIONAL {?x dbpedia:birthPlace ?place}
        FILTER (CONTAINS (?name, 'Bob'))
}
```

	naive	*filter*	*binding*	*full*
Query 4.3 (10 ws calls, 3 results)				
Transfered triples	169927	115448	54486	7
DQP time (s)	9.730	7.918	4.350	**2.387**
Query 4.9 (944 ws calls, 244 results)				
Transfered triples	17472950	17418766	54835	651
DQP time (s)	518.683	498.416	8.608	**6.981**

Table 4.2: Searching, over two distributed DBpedia datasets,

Results. We can observe that the *full* strategy, which combines both the *filter* and *binding* strategies drastically reduces the cost of the distributed evaluation for both queries (selective and less selective). For the less selective query, the *naive* strategy terminates in more than 8 minutes whereas only 7 seconds are needed to get the results with the *full* strategy.

The following table 4.3 provides some details to understand the contribution of each strategy on the overall distribution process, for a sample query involving a single distributed join.

Query 4.3 (10 ws calls, 3 results)	*naive*	*filter*	*binding*	*full*
DQP#1 {?x foaf:name ?name} (ms)	4424	**2193**	4065	**2265**
DQP#2 {?x dbpedia:birthDate ?date} (ms)	1676	2132	**15**	**14**
DQP#3 {?x dbpedia:birthDate ?date} (ms)	1042	1098	**11**	**12**
DQP#4 {?x dbpedia:birthDate ?date} (ms)	1284	1295	**9**	**10**
DQP#5 {?x dbpedia:birthDate ?date} (ms)	1025	1042	**9**	**8**

Table 4.3: Evaluation of the time needed to perform each edge request. The combination of both *filter* and *binding* strategies shows the best results.

The first edge request (DQP#1) is computed and retrieves 4 matching persons (their name contains "Bobby A"). The distributed join consists in, for each of the 4 resulting persons, retrieving their birth date. It leads to 4 distributed evaluation of the same edge requests, ?x dbpedia:birthDate ?date, over the federated endpoints.

We can observe that under the *naive* strategy, the evaluation of these edge requests (DQP#2, #3, #4 and #5) are expensive and not very clever because they do not exploit the 4 values of ?x retrieved from the first edge request. Ideally, their evaluation should return at most 4 results, but actually, the totality of the dbpedia:birthDate properties are retrieved from the federated endpoints, and the join is finally processed by the KGRAM federator at a high cost. The *filter* strategy improves the results but in a limited extent since it only affects the first edge request (it is not applicable for the other edge requests, since they do not involve the ?name). The *binding* strategy addresses this issue by not impacting the cost of the evaluation of the first edge request, but by drastically reducing the cost of the following edge requests. Finally we can observe that the combination of these two strategies - *filter* impacting the first edge request and *binding* impacting the remaining edge requests - lead to the best results, as shown in table 4.2.

4.3.2 Impact of the federator parallelism optimizations

In this experiment, we show that the pipelined implementation of the federator is positive for non-selective queries, in the context of small size knowledge bases. However, for larger ones, or very selective queries, this optimization has no significant impact, except in the case of knowledge bases having heterogeneous response time.

Material and methods. The experimental setup consists in two queries, a selective one and another less selective (queries $Q_{4.3}$ and $Q_{4.9}$ presented above), a virtual single knowledge base composed by 338K DBpedia-person RDF statements, fragmented over 1, 2, 4, 6 and 8 distributed stores, and two KGRAM federator implementations. Both implementations exploit the *full* query rewriting optimization. The first one implements the *parallel-wait* strategy while the second one implements the *parallel-pipeline* strategy. This experiment is also conducted on the "Suno" cluster of the Grid'5000 infrastructure in which 9 nodes have been reserved allowing to deploy 1 KGRAM federator and up to 8 federated KGRAM endpoints.

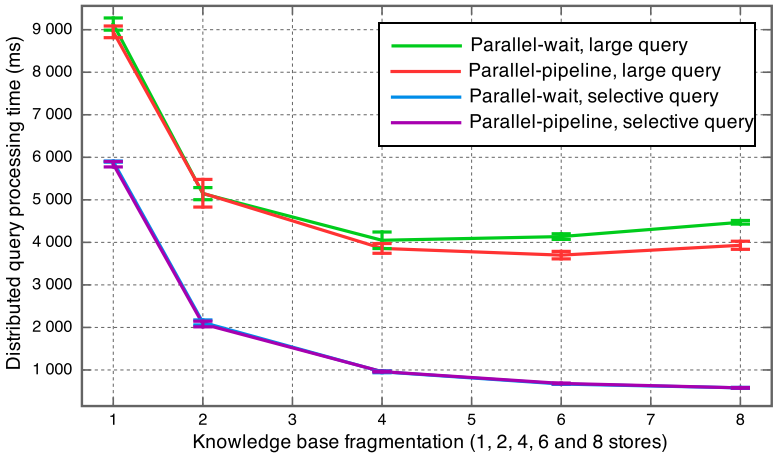

Figure 4.4: Impact of the federator parallelism optimization on a medium size (338K triples) knowledge base, fragmented into 1, 2, 4, 6 and 8 distributed stores. The *parallel-pipeline* strategy is beneficial for queries producing a lot of intermediate results.

Results. Figure 4.4 compares several distributed query processing evaluations for both the selective query and the less selective one. For the selective query, no improvement is measured (blue and purple curves are almost superposed). For the less selective query, a gain is observed starting from 4 distributed stores. Indeed, the more the dataset is fragmented, the more threads are launched in parallel from the federator side, allowing to better anticipate the post-processing of results sent by federated endpoints. We also observe that starting from 4 distributed stores, the global query evaluation time increases. This can be explained by the cost of network communications that becomes penalizing when the number of distributed stores is increasing.

We also observe from figure 4.5 that on the full DBpedia-person dataset (1.7M triples), the pipelined strategy has no impact. Indeed, by being fragmented into sub-parts of the same size, we can imagine that remote edge requests will terminate at the same time, making impossible to anticipate the post-processing while other federated endpoints are still working.

We propose a complementary experiment assessing the interest of the *parallel-pipeline*

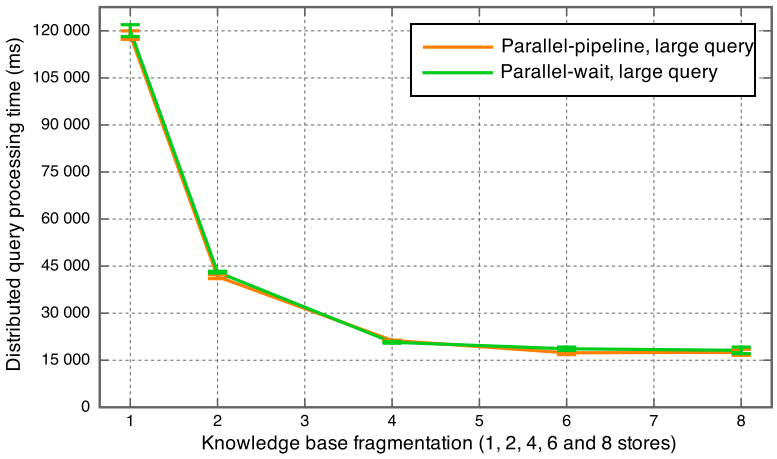

Figure 4.5: Decreasing distributed query processing (DQP) time for a large scale knowledge base (1.7M triples) fragmented into 1, 2, 4, 6 and 8 data stores, under *full* query rewriting strategy.

strategy when federated endpoints that have different response time, due to the size of the hosted knowledge base.

	Parallel-wait	*Parallel-pipeline*
run 1 (ms)	108890	98834
run 2 (ms)	108179	97867
run 3 (ms)	107020	95787
mean (ms)	108029.66	**97496**
std dev (ms)	943.90	1557.01

Table 4.4: Several distributed query processing times under the *parallel-wait* and *parallel-pipeline* strategies for the non-selective query $Q_{4.9}$ performed over 8 inhomogeneous knowledge bases.

Table 4.4 shows the interest of the *parallel-pipeline* strategy when data is fragmented unevenly. Indeed, the fragmentation setups (knowledge base fragmented in chunks of the same size) used in the previous experiments are not realistic in the context of autonomous distributed knowledge bases where really different response times might be observed. This variability could be due to knowledge base hosted through heterogeneous hardware, such as different CPU performance or RAM capacity, or even network bandwidth. Moreover, it is unlikely that each distributed knowledge base has exactly the same size. To reflect this variability, we consider in this experimental setup 8 federated endpoints. These endpoints are deployed through 8 nodes of the "Helios" cluster of Grid'5000 (composed of 56 nodes, each node has 2 AMD CPUs (2.2GHz) with 2 cores per CPU, 4GB of RAM, and 63GB of disk storage) and are hosting respectively 25K, 50K, 100K, 150K, 200K, 250K, 300K and 670K triples. If we observe the mean time needed to perform the distributed querying, we measure a gain of **9.75%**, thus assessing the interest of anticipating the post-processing of results, especially in the context of heterogeneous

distribution setups.

4.3.3 Impact of the dynamic source selection

In this experiment, we compare the evaluation of the seven life-science queries (LS1 to LS7) of the FedBench benchmark with and without data source selection as proposed in section 4.2.3.3. The FedBench benchmark is more precisely described in chapter 8 dedicated to experimental evaluations.

Material and methods. The experimental setup consists in seven identical nodes reserved on the Grid'5000 Suno cluster, the first node is dedicated to the query federation engine, and the following six nodes are dedicated to host the FedBench datasets (totalizing 49 million statements).

Results. Table 4.5 shows a significant gain (up to 15%) for only two queries LS3, and LS5. Indeed, since these two queries are known to generate a lot of intermediate results, thus leading to costly distributed joins, the proposed cache mechanism used in data source selection allows for saving communication costs. However, this optimization has no significant impact on selective queries, being evaluated in less than a few seconds.

	LS1	LS2	LS3	LS4	LS5	LS6	LS7
Federated querying **without** *source selection*							
Eval. Time (s)	0.55	0.74	**44.37**	0.30	**21.36**	1.28	12.61
Std. Dev. (s)	0.00	0.00	**2.96**	0.00	**1.73**	0.04	1.20
Federated querying **with** *source selection*							
Eval. Time (s)	0.59	0.74	**37.62**	0.30	**18.85**	1.31	12.71
Std. Dev. (s)	0.00	0.00	**1.36**	0.00	**0.85**	0.03	3.36

Table 4.5: FedBench results for the KGRAM federation engine with and without source selection.

4.3.4 Impact of the edge grouping algorithm

In this experiment, we (i) validate the interest of the edge grouping strategy and through the automated generation of SPARQL SERVICE clauses, and (ii) illustrate the non-trivial ordering issues introduced by this optimization strategy, possibly leading to sub-optimal evaluations.

Material and methods In this scenario, we consider two distributed data sources. The first one, Src_1 deployed at http://dbpedia, hosts 100K DBpedia-persons statements. The second one, Src_2 deployed at http://neurolog, hosts the 15K statements of the NeuroLOG platform. Query $Q_{4.10}$ involves two edge requests specific to the dbpedia repository,

and one specific to the neurolog repository. Query $Q_{4.11}$ represents the corresponding optimized query including SERVICE clauses.

Listing 4.10: SPARQL query involving both NeurolOG-specific and DBpedia-specific statements

```
1  PREFIX foaf: <http://xmlns.com/foaf/0.1/>
2  PREFIX dbpedia: <http://dbpedia.org/ontology/>
3  PREFIX linguistic−expression: <http://www.irisa.fr/visages/linguistic−expression−owl−lite.owl#>
4  SELECT DISTINCT ?x ?name ?date WHERE {
5          {
6                  ?x foaf:name ?name .
7                  ?x dbpedia:birthDate ?date .
8          } UNION {
9                  ?y linguistic−expression:has−for−name ?name .
10         }
11         FILTER (CONTAINS (?name, 'Bob'))
12 }
```

Listing 4.11: Optimized SPARQL query through edge grouping in SERVICE clauses.

```
1  PREFIX foaf: <http://xmlns.com/foaf/0.1/>
2  PREFIX dbpedia: <http://dbpedia.org/ontology/>
3  PREFIX linguistic−expression: <http://www.irisa.fr/visages/linguistic−expression−owl−lite.owl#>
4  SELECT DISTINCT ?x ?name ?date WHERE {
5          {
6                  SERVICE <http://dbpedia> {
7                          ?x foaf:name ?name .
8                          ?x dbpedia:birthDate ?date .
9                          FILTER (CONTAINS (?name, 'Bob'))
10                 }
11         } UNION {
12                 SERVICE <http://neurolog> {
13                         ?y linguistic−expression:has−for−name ?name .
14                         FILTER (CONTAINS (?name, 'Bob'))
15                 }
16         }
17 }
```

Query $Q_{4.12}$ illustrates a particularly inefficient SPARQL query, by design. More precisely, a statement has been introduced line 6 to search for all classes of instances. We will see in the results table that the "Service-first" heuristic of the rewriting strategy allows for drastically reducing the evaluation cost of this kind of query. Query $Q_{4.13}$ illustrates the corresponding optimized query in which this costly edge request statement (?x rdf:type ?t) has been rewritten (line 11) after the first SERVICE clause (line 6).

Evaluations have been performed on a standard Apple laptop (2.6GHz Core2Duo CPU / 8Go RAM). Two instances of the apache Tomcat container were running on two different ports and hosting each a KGRAM endpoint. Src_1 was implemented by the first

Listing 4.12: SPARQL query involving both NeurolOG-specific and DBpedia-specific statements

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dbpedia: <http://dbpedia.org/ontology/>
PREFIX linguistic−expression: <http://www.irisa.fr/visages/linguistic−expression−owl−lite.owl#>
SELECT DISTINCT ?x ?name ?date WHERE {
        {
                ?x rdf:type ?t .
                ?x foaf:name ?name .
                ?x dbpedia:birthDate ?date .
        } UNION {
                ?y linguistic−expression:has−for−name ?name .
        }
        FILTER (CONTAINS (?name, 'Bob'))
}
```

Listing 4.13: Optimized SPARQL query through edge grouping into SERVICE clauses

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dbpedia: <http://dbpedia.org/ontology/>
PREFIX linguistic−expression: <http://www.irisa.fr/visages/linguistic−expression−owl−lite.owl#>
SELECT DISTINCT ?x ?name ?date WHERE {
        {
                SERVICE <http://dbpedia> {
                        ?x foaf:name ?name .
                        ?x dbpedia:birthDate ?date .
                        FILTER (CONTAINS (?name, 'Bob'))
                }
                ?x rdf:type ?t .
        } UNION {
                SERVICE <http://neurolog> {
                        ?y linguistic−expression:has−for−name ?z .
                        FILTER (CONTAINS (?name, 'Bob'))
                }
        }
}
```

KGRAM endpoint to host the sub DBpedia-persons dataset, and Src_2 was implemented by the second KGRAM endpoint to host the NeuroLOG dataset. The KGRAM federation engine was configured with all proposed optimizations, namely filter rewriting, binding rewriting, source selection and edge grouping.

Results Table 4.6 summarizes the evaluation of the sample queries $Q_{4.10}$ and $Q_{4.12}$ without and with edge grouping in SERVICE clauses (queries $Q_{4.11}$ and $Q_{4.13}$). We measured in this experiment an average time through three runs, and the corresponding number of remote SPARQL queries handled by data source Src_1 and Src_2. ASK queries correspond to the construction of the cache index needed for both source selection and edge grouping strategies. SERVICE queries correspond to sub SPARQL queries, resulting from the edge grouping strategy, evaluated in a single block. CONSTRUCT queries represent unitary edge requests as already presented for filter and binding rewriting. All queries lead to 53 SPARQL results.

		ASK		SERVICE		CONSTRUCT		
	Average time (s)	Src_1	Src_2	Src_1	Src_2	Src_1	Src_2	Remote invocations
$Q_{4.10}$	2.3	3	3	0	0	103	1	110
$Q_{4.11}$	1.2	3	3	1	1	0	0	8
$Q_{4.12}$	118	4	4	0	0	25035	1	25044
$Q_{4.13}$	2.1	4	4	1	1	53	53	118

Table 4.6: Prohibitive evaluations ($Q_{4.12}$) can be achieved in a reasonable amount of time through edge grouping optimizations ($Q_{4.13}$).

Table 4.6 shows that the amount of remote invocations is reduced by a factor 13 when performing edge grouping for the sample query $Q_{4.10}$. In the context of a particularly inefficient query ($Q_{4.12}$), the amount of remote invocations is reduced by a factor of 212. If we focus on the measured average times, the edge grouping saves 52% of evaluation time for query $Q_{4.10}$, and 98% for query $Q_{4.12}$. These results are encouraging, and show that the edge grouping strategy may be useful to drastically reduce the amount of remote invocations.

Discussion The optimized query $Q_{4.13}$ resulting from the "Service-first" heuristic used in the edge grouping strategy allowed saving many remote communications and thus drastically reducing the evaluation time.

It can be implemented thanks to the query reordering achieved when the initial query has been rewritten. Indeed, the costly edge request `?x rdf:type ?t` (let us name it E_1), is positioned after the first SERVICE clause, S_1. The order $\{S_1, E_1, S_2\}$ is particularly well adapted because S_1 leads to few intermediate results (by including a filter value restriction), and these few intermediate results are joined with S_1 by using the dynamic bind joins introduced through the join rewriting strategy (see section 4.2.3.2).

The evaluation time would have been much longer when considering an "Edge-first" heuristic because of the cardinality of E_1, and results would have been similar to those

observed for the evaluation of $Q_{4.12}$. Indeed, $Q_{4.12}$ starts with E_1 and is particularly ineffective for the same reasons. E_1, executed first, leads to a huge amount of result candidates which are then reinjected into distributed joins. 25035 CONSTRUCTS are performed over repository Src_1.

However, as a counter example, we could imagine a single edge E'_1 similarly located on both repositories Src_1 and Src_2, that would be very selective. Then, it would be more efficient to adopt an "Edge-first" heuristic thus leading to the following order $\{E'_1, S_1, S_2\}$. We could also imagine that S_1 and S_2 have different selectivity[6] and their order has an impact on the distributed evaluation of the overall query. Finally, when considering two edges E_2 and E_3 linked together through a common variable, thus benefitting from bind joins introduced in section 4.2.3.2, it might be sub-optimal to group each of them in separate SERVICE clauses because this pattern breaks the bind join optimization.

To conclude, it is difficult to propose a robust edge grouping strategy without considering a full query planner addressing the ordering issues introduced above. Similarly to relational database query optimizer, dynamic programming techniques as well as a cost function based on the cardinality of subqueries (single edge requests, and SERVICE clauses) should be carefully studied to enhance the robustness and the effectiveness of the proposed edge grouping strategy federated querying over distributed knowledge graphs. Moreover, this work could benefit from more sophisticated indices such as the frequent graph pattern indices introduced in [Basse et al., 2010] used to advertise the content of RDF triple stores and thus helping the federated querier to select the most appropriate data source.

4.4 Discussion and conclusion

In this chapter, we addressed semantic data sharing through the federation of distributed multi-source abstract knowledge graphs. We adopted a transparent federated approach to cope with application-level constraints encountered in the life-science area. As a reminder, in the context of distributed neuroimaging data providers, semantic data might be shared through a common domain ontology which prevents from knowing, at query design-time, the potential content of a data source. The other prime-order constraint is the autonomy property of data providers which do not allow data-warehousing approaches. As a consequence, we consider distribution setups involving homogeneous data (sharing the same semantics), with possibly heterogeneous data structures, due to independent/historical data source operation.

To achieve transparent federated querying, and thus collect knowledge from multiple distributed data sources (RQ_1), we considered a fine-grained parallel evaluation strategy. It allows to make no assumption on the content of the distributed data sources, but the main drawback is its cost. Indeed, a naive implementation leads generally to a large

[6] Selective expressions leads to few intermediate results and thus lead to more efficient distributed joins.

amount of network communications, which, in some cases, makes the distributed evaluation prohibitive. We introduced, all along this chapter, optimization strategies aiming at reducing the cost (RQ_3) – in time and volume of network communications – of the distributed evaluation of SPARQL queries over fragmented and distributed knowledge graphs.

The first family of optimizations consists in query rewriting strategies (section 4.2.3), both static and dynamic, which (i) incorporate, when applicable, filters remotely computed at data source, and (ii) reuse intermediate results by using variable bindings (already known values) to finally prevent from recomputing already known values (from the federator side). The combination of these query rewriting strategies showed promising results. However, the impact of the *binding* strategy should further be studied. More precisely, we could imagine some cases, possibly leading to sub-optimal distribution evaluations. Indeed, if we consider two consecutive edge requests, with a first one which is very expensive (leading N intermediate results, N being huge), the evaluation of the second edge request would lead to N distributed evaluations incorporating the intermediate results. Without the *binding* strategy, we would have a single evaluation, possibly transferring more results, and thus more expensive from the data sources perspective. Even if SPARQL queries are sometimes written beginning with the most selective edges first, this question should further be studied to find a tradeoff between joins performed with the *binding* strategy and traditional joins, to avoid sub-optimal evaluations.

In addition to these query rewriting strategies, we proposed a pipelined strategy (section 4.2.4.2) aimed at consuming results as soon as they become available from the federator perspective. We observed that this evaluation is beneficial for distributed query evaluations producing a large amount of intermediate results.

Finally, based on dynamic data source selection with SPARQL ASK queries, we proposed an algorithm (section 4.2.3.4) to achieve coarse-grained parallelism through dynamic edge requests grouping into SPARQL 1.1 SERVICE clauses. In the less favorable query considered in the experiment of section 4.3.4, this strategy allowed to save 52% of the distributed evaluation time, which is promising. However we showed that triple pattern grouping may lead to sub-optimal evaluations, mainly due to ineffective groups and edges reordering. This edge grouping strategy opens interesting perspectives but still needs to be studied through cost functions and query planning approaches. Moreover, advanced indices, such as the ones introduced in [Basse et al., 2010] may be considered to efficiently and precisely describe the content of RDF triple stores, thus allowing edge request grouping based on advanced graph pattern indices.

Results of this chapter 4 open interesting perspectives for efficient transparent querying over distributed knowledge bases, while still benefitting from almost all the expressivity of the SPARQL 1.1 language (except SPARQL 1.0 named graphs).

For short-term perspectives, another enhancement would consist in evaluating asynchronously disjunctive queries. More precisely, the two operands of a SPARQL UNION query could be parallelized. In addition, to complete our approach with a full compati-

bility with SPARQL 1.1, named graph should be operationalized in a distributed setup.
This chapter opens several long-term perspectives:

Efficient DQP. Indeed, still concerned by the efficiency of distributed query processing, query planning techniques could tackle order issues when composing SPARQL SERVICE clauses generated through the proposed edge grouping strategy.

Dynamic semantic alignment. Since it has been possible to address structural heterogeneity by providing a KGRAM *Producer* dedicated to enumerate relational data as graph components, we could imagine another kind of KGRAM *Producer* dedicated to semantic alignment thus addressing a challenging semantic interoperability issue. This alignment component would be responsible to (i) align a semantics into the data model of its associated data source, and (ii) re-align back the matching data to the data model associated to the initial query.

Optimized evaluation for multi-core CPUs. We observe nowadays a stagnancy of CPU frequency. However, CPUs become parallel by involving more and more independent computing units (multi-core, multi-CPU computers). Similar distributed evaluation strategies could be envisaged to perform optimized evaluation on such multi-core, multi-cpu computers. Technically, the Java 7 Fork/Join capability could be used to benefit from multi-core infrastructures.

SPARQL Update in a distributed setup. We have seen in this chapter how it is possible to efficiently collect knowledge from distributed multi-source knowledge graphs. But SPARQL 1.1 includes directives to not only query but also insert new statements into a knowledge graph. In the context of the distributed e-Science platform considered in this thesis, it would be interesting to study how resulting semantic data may be introduced back in a distributed setup. Distributed "updates" opens new opportunities to handle, for instance, semantic data synchronizations in a dynamic distributed setup, where the availability of data sources is not guaranteed.

Elastic SPARQL endpoint. Towards cloud computing areas, the distributed querying techniques envisaged in this thesis could be reused to propose an elastic cloud-based SPARQL endpoint. The idea would be the following. We have seen in Figure 4.4 that starting from 4 distributed endpoints, it is useless to continue fragmenting the initial dataset. However if we consider a growing dataset, at some time, it would be more interesting to increase the fragmentation of data. Autonomic computing techniques could thus be used to monitor the distributed dataset and the distributed evaluation response time for some sample queries. Deployed on a cloud infrastructure (or a cluster) KGRAM-DQP could automatically allocate a new computing node, fragment and redeploy the dataset to offer better scalability.

Key Points

- *Our federation approach does not require, at query design-time, for prior knowledge on the content of the federated data sources.*
- *The proposed federated querying adapts to federation changing such as joining or leaving partner sites and is non invasive with respect to the initial SPARQL queries.*
- *The distributed query processing is efficient through static and dynamic optimizations while still benefitting for almost all the expressivity of the SPARQL 1.1 query language.*
- *Dynamic source selection allows for triple pattern grouping through SPARQL Service clauses but may lead to sub-optimal evaluation plans.*

Part II

Extending knowledge bases through scientific workflows

CHAPTER 5

Semantic services in scientific workflows

Contents

5.1 Introduction

Semantic web services have emerged as a research domain on its own. Semantic web services aim at exploiting semantic technologies to enhance service oriented architectures. In particular, benefits are expected when discovering and composing services, but also in the process of mediating syntactically incompatible services.

These expected benefits are promising in the context of scientific workflows which generally rely on service oriented architectures. In this chapter, we first review, in section 5.2 the main approaches towards the provision of Semantic Web Services. These approaches are directly related to the research questions RQ_3 and RQ_4 addressed in this manuscript towards unambiguous semantic service annotations and user assistance when composing services together into scientific workflows. Then we review in section 5.3 the current approaches for provenance data management in order to enhance the overall quality and reproducibility of e-Science experiments (achieved through scientific workflows) and to ease the exploitation of produced data (RQ_5).

We defend the idea of concise but meaningful provenance statements to finally assist en-user in the exploitation of their experiment results. To achieve this objective we couple semantic services and provenance approaches. Our approach is positioned in the context of these two areas in sections 5.2.4 and 5.3.5.

5.2 Semantic Web Services

Nowadays, organizations heavily rely on *Service-Oriented Architecture (SOA)* to improve their agility in a quickly evolving and competing world. Enterprises need to cost-effectively and almost instantly react to evolving markets. Similarly, scientific communities need to continuously react to novel conceptual or technical approaches to tackle their information processing challenges. Inside or outside organizational boundaries, sharing is also a major issue to foster collaborations. Service-oriented architectures provide some solutions by focusing on loosely-coupled processes, on their description and therefore on reusability.

More precisely, service-oriented architectures provide software infrastructures to publish and consume, over a network, units of information processing, *Services*. The World Wide Web has evolved to allow the distribution and the consumption of such services through Web Service standards, over established web protocols. Web services rely mainly on message exchanges between service providers and service consumers. Messages are encoded in XML and conform to the Simple Object Access Protocol (SOAP) standard. The Web Service Description Language (WSDL) standard focus mainly on describing the SOAP messages a Web Service can understand and produce as results. Web Services are interoperable in the sense that a consumer does not need to know how the provider implemented the service, but only needs to know the service requirements, through its WSDL descriptor, in order to provide the needed SOAP messages for invocation, and for interpretation of results.

But Web Services do not provide any external understanding about the nature of the information processing implemented so that, even supported by service orchestration tools or languages, elaborating a flow of services (workflow), must be down manually with a clear understanding of how the data is processed. The nature of data transformation is highly dependent on the domain in which the service is created or used. Another reason that prevents automation when designing workflows is the lack of description about (i) requirements on domain entities prior to service invocation, and (ii) effects on domain entities when the service invocation is finished. All this additional descriptive information involving domain specific concepts is generally called semantic description, opposed to syntactic description provided by WSDL.

Semantic-Web technologies are good candidates to address issues raised by coherently managing a huge variety of published services over the Web or institutional networks. Semantic Web Services area tackles several challenges, such as the reliability of service search tasks, the assistance when plugging together available services, or the several degrees of (in)compatibility between services either at composition-time, or run-

time. In that sense, the Semantic Web Services Challenge initiative [Petrie et al., 2008] promotes Semantic Web Service technologies and aims at enhancing their common understanding through a set of use cases.

The semantic web services area addresses the following challenges:

Annotation. The *annotation* task is a prerequisite to bridge Web Services and the Semantic Web. Annotating a service consists in using an ontology to bind technical concepts, usually elements describing web services (including their messages), to domain specific concepts. This is a costly task mainly because the expert performing the annotation need both technical and domain specific knowledge. For instance, let us consider a medical image processing tool performing a de-noising operation. From a technical or syntactical point of view, the service might be implemented by an executable binary taking as input a raw file and producing as output another raw file containing the de-noised image. But from a semantic point of view, this de-noising service might implement a kind of algorithm characterizing how the image is processed. The service might additionally require a specific medical image format, and a specific modality of acquisition, for instance ultrasound. Moreover, the resulting de-noised image should conserve the same modality. In other world, even de-noised, the image still has been acquired through an ultrasound device. Achieving automation in the generation of such new annotation is part of the Semantic Web Services challenges.

Discovery. Once annotated, the service can be searched through natural terms for domain experts. This task is generally known as semantic *discovery* process. Capabilities of services can therefore be classified with reasoners so that semantic search engines expose most adequate services. Search requests can also involve the description of information to process, or to be processed, in order to retrieve the service candidates most able to consume or produce this kind of information. Service discovery activities rely on so-called matchmaking algorithms, generally operationalized through inference rules and reasoners.

Composition. Whereas Web Service Choreography languages address complex issues such as global web services coordination and collaboration, the service *composition* challenge aims at providing assistance to the workflow designer when plugging together services. Considering a given semantic web service S, the process of composition rely mainly on discovering either candidate services able to produce semantically compatible data as inputs of S, or able to consume data semantically compatible with data produced by S. Semantic compatibility will be presented later on.

Mediation. The *mediation* challenges consist in, considering two heterogeneous services which are semantically pluggable, taking into account syntactic mismatches. Services able to perform data transformations or conversions to conform to syntactic

constraints are not always available, and it might be not desirable to expose them as services. Some semantic web service approaches therefore consider providing data transformation guidelines or implementations to fulfill syntactic mismatches.

The following sections 5.2.1, 5.2.2, and 5.2.3 describe respectively "top-down", "bottom-up", and domain-specific approaches addressing these semantic web services challenges.

5.2.1 From service models to semantic web services

OWL-S is one of the major initiatives to bring Semantics to Web Services [Martin et al., 2007a, 2004]. OWL-S is a set of notations based on OWL [Mcguinness and van Harmelen, 2004], the Semantic Web ontology language. This framework is composed of three linked ontologies, *(i)* the *profile ontology* which describe what the service does, *(ii)* the *process ontology* which describe how the service is used and *(iii)* the *grounding ontology* which describes the service interface. High-level objectives are to provide a standard[1] representational framework for Web Services, to support automation of Web Services management and to be comprehensive enough to cover the whole lifecycle of service tasks. Key concepts used to describe Services are *inputs*, *outputs*, *preconditions*, and *results* which refer to *effects* (postconditions) on outputs specifications. The goal of the *profile* sub-ontology is to support capability-based discovery. The *profile* defines functional, classification, and non-functional aspects of a service. The functional aspect represents actually the information transformation between the inputs and the outputs, and the domain transformation between the preconditions and the effects. The classification aspect represents the type of a service in a taxonomy. The non-functional aspect represent transverse concerns such as security or quality of service, that can be defined "by extension" using service parameters that refer to another ontology. The *process* sub-ontology is another workflow language based on atomic processes (without internal structure), and composite processes (with both control flow and data flow). The *grounding* sub-ontology links *(i)* each *atomic process* to a deployed Web Service, and more precisely, to a WSDL[2] [Christensen et al., 2001] operation, and *(ii)* abstract *inputs* and *outputs* to WSDL messages. Other works on the applications of OWL-S are shown in terms of service enactment, service discovery, service composition (planning techniques). Related approaches to OWL-S such as WS-BPEL, ebXML, CDL, SWSF, WSMO or WSDL-S are presented in [Ankolekar et al., 2004].

WSMO (Web Service Modeling Ontology) is another major initiative to address Semantic Web Services challenges. Roman and coworkers propose a unified framework composed of a conceptual model for Semantic Web Services, a formal language for their description, and an execution environment hosting semantic web services [Roman et al., 2006]. WSMO is proposed to describe *Web Services*, *User Goals*, and to tackle the interoperability issues through *Ontologies* (potentially domain specific) and *Mediators*. The

[1] from both Semantic Web and Web Services standards

[2] WSDL is the *de facto* standard to describe Web Services, but the grounding sub-ontology of OWL-S can be exploited with other service implementations such as UPnP.

usage of such mediators is specific to WSMO. They capture mappings that cannot be expressed through ontologies and are used as computing elements to resolve mismatches when plugging together WSMO elements. They are able to address the heterogeneity of ontologies, of goals, of web services, and also the one from web services to user goals. WSML is a syntactic framework for a set of layered languages covering the description of ontologies, web services, user goals and mediators. The execution environment WSMX originally designed to be a testbed for WSMO provides support for semantic web services discovery and invocation but also provides an internal workflow engine and a resource manager able to store/retrieve general data produced within the environment.

5.2.2 From legacy web services to semantic web services

Derived from WSDL-S, the World Wide Web Consortium (W3C) released in 2007 the SAWSDL standard [Farrell and Lausen, 2007], aimed at defining Semantic Annotations for WSDL and XML Schema. Martin and co-workers argue that a joint usage of OWL-S and SAWSDL [Martin et al., 2007b] is a good way to tackle Semantic Web Services challenges. SAWSDL is indeed presented as lightweight but too modest to address Semantic Web Services objectives. SAWSDL introduces the annotations *modelReference* (for both WSDL and XSD) and *{lifting,lowering}SchemaMapping* (only for XSD). Some limitations of the *modelReference* annotation are pointed out. For instance, if we consider two WS operations with two input parameters sharing the same XML Schema type but two different semantic referents A and B, the two services become ambiguous and it hampers the discovery process. Indeed the two services cannot be distinguished if the two parameters p_1 and p_2 have for semantic referent respectively A and B for the first service and p_1 and p_2 have for semantic referent respectively B and A for the second service. OWL-S classes (*input*, *output*, and *process*) should be used as an intermediate description between a WS and a domain ontology. The authors conclude by giving a set of recommendations for annotating web services with both SAWSDL and OWL-S.

Similarly to the joint usage of SAWSDL and OWL-S presented before, Vitvar & al. propose WSMO-Lite [Vitvar et al., 2008], a bottom-up service modeling approach to benefit from the SAWSDL standard in WSMO-fashioned Semantic Web Services. This framework is lighter than WSMO: *mediators* are considered as provided by the infrastructure, similarly to *user goals* which are supposed to be provided by the discovery mechanism. Contrary to WSMO, WSMO-Lite does not constraint to use WSML but allows the usage of any ontology written in an RDF syntax. This lightened version no more focuses on semantics of pre/post-conditions and effects as they are domain specific. The main difference between the OWL-S approach to conform to the SAWSDL standard, is that they chose to reduce WSMO to a minimal model for Semantic Web Services.

Derouiche & Nicole propose to extend the Windows Workflow Foundation with SAWSDL documents to guarantee semantic type correctness in scientific workflows [Derouiche and Nicole, 2007]. Their contribution consists in adding a new "Semantic Web Service" activity responsible of annotating web services (WS) annotation. The semantic type checking is performed by extending the binding activity (data flows between services) of

the framework with calls to this "Semantic Web Service" activity. Structural mismatch are solved using *lifting* and *lowering* schema mappings in conjunction with XSLT transformations. The authors also refer to a prototype from the Taverna workbench used to address semantic mismatch detection and resolution in workflows, but the binding of concrete and semantic datatypes is not addressed and prevent the support for workflow enactment.

Lécué & Moreau also tackle semantic and syntactic web services composition in a quite similar approach [Lécue et al., 2008]. Their motivation is based on the large number of approaches dealing with service discovery but the lack of these addressing syntactic message transformation to perform effective mediation. As background information, they describe *causal links* and *data integration* in the Web. Causal links are defined in a formal model [Lecue and Léger, 2006] and represent semantic connections between functional output and input parameters of a service. From an operational point of view, the semantic annotation of these parameters might be realized using SAWSDL, WSMO capability or OWL-S profile. The matching / binding of parameter is described and categorized between five matching types: *exact, plugin, subsume, intersection, and disjoint* which allows at design time, to qualify several levels of (in)compatibilities. Data integration in the Web is considered through the use of XSL technologies to transform XML messages. The proposed semantic matchmaker consists in, being assumed that the control flow is already determined, scoring all causal links between parameters. For syntactic adaptations, no guidelines are provided by the SAWSDL standard. The proposed syntactic engine automatically generates, from the semantic links, an XSL transformation to adapt XML messages between services trough a syntactic and structural analysis of XML schemas [Boukottaya and Vanoirbeek, 2005].

Often syntactically described through WSDL, Grid Services are also desired to be semantically discovered [Selvi et al., 2006]. Thamarai Selvi & al. propose a bottom-up approach leading to a matchmaking system not only based on input/output description of services but also based on their functionalities. The proposed framework is based on OWL-S tools to generate an OWL-S instance of a service from its WSDL. The similarity distance between services is computed through a *Domain ontology* which describes input and output datatypes, and a *Function ontology* which describes the capabilities of services. The authors conclude with a favorable evaluation showing that the elimination of irrelevant services, by considering also their functionalities, is more efficient that by considering only their parameters.

The FUSION project [Kourtesis and Paraskakis, 2008] aims at supporting enterprise application integration through Semantic Web Services technologies, namely UDDI [OASIS, 2004], SAWSDL and OWL. Service discovery requirements cover functional and non-functional properties. *Inputs*, *Outputs*, *Preconditions* and *Effects* (IOPE) are generally considered but FUSION only focus on *Inputs* and *Outputs* (*i.e.* information transformations between services and not state-wise conditions) at both message and schema levels.The classification of the service within a classification scheme is considered as a non-functional property. The proposed architecture is independent from any standard implementation of UDDI, or OWL knowledge bases. Software components UDDI4J,

SAWSDL4J, WSDL4J, OWL-API and Pellet reasoner are the basis of a set of web services used to publish and discover services.

The METEOR-S research project is another major initiative in the Semantic Web Services area [Sheth et al., 2008]. The approach is based on a peer-to-peer middleware to address service discovery and publication. SAWSDL is used for both services annotation (through *modelReferences*) and data mediation (through schema *lifting/lowering*). A new approach, gaining more and more popularity, is introduced as a lightweight manner to address semantic web services. It is based on RESTful web services and microformats for their description. RESTful web services are often presented as simple Web API through HTTP (GET, PUT, POST, DELETE, etc.) without any description language like WSDL which prevents discovery capabilities. This need is therefore addressed through the recommended usage of the hRESTs microformat [Kopecký et al., 2008], which can be directly embedded within web pages and searched through regular web search engines.

Cardoso and co-workers review the state-of-the-art in Web Service Discovery and propose a new algorithm to tackle semantic discovery [Cardoso et al., 2008]. The novelty of the approach consists in (i) considering a judgment process on both similar and distinguishing features (Tversky's model), (ii) not only considering inputs and outputs but also the functionality of the service, and (iii) considering potentially different semantic references. This work is based on the METEOR-S infrastructure. The authors review the three main approaches developed in the Semantic Web Services research area, namely OWL-S, WSMO, and SAWSDL annotations, with a focus on SAWSDL. Four discovery approaches are covered: (i)*I/O matching*, (ii) *Multi-level matching* by exploiting functional and non-functional service information, (iii) *Graph-based matching* based on directed graph representing service descriptions and linked concepts from domain ontologies, and (iv) *Syntactic matching* based on information retrieval techniques. Two tools (Eclipse plugins) resulting from the METEOR-S project are used in this context: Radiant for service annotation and Lumina for service discovery/advertisement.

Built upon the Semantic Web search engine CORESE [Corby et al., 2004], Lo & Gandon propose an approach to address enterprise application integration through Semantic Web Services [Lo and Gandon, 2005]. To annotate services, their approach relies on the *profile* and *grounding* ontologies, and on the *input* and *output* of the OWL-S *process* ontology. From the user point of view, a Semantic Web Portal embedding CORESE is used to discover services trough SPARQL queries. This portal is also dedicated to assist the user during Web Service composition tasks. The CORESE engine can therefore be seen as a semantic UDDI, acting as a Web Service Broker in classical service oriented architectures. Two cases are considered for interactive WS composition addressing inputs availability or desired outputs. Sequences of services (a set of service combinations) can also be discovered to achieve several data transformations between available inputs and desired outputs. The composability is computed by the engine through the execution of *production rules*.

To address semantic mediation between business partners (one of the Semantic Web Services challenges) Küster & König-Ries [Küster and König-ries, 2007] propose the DIANE Service Description (DSD). Contrary to OWL-S, DSD focuses on a clear distinction

between service offers and requests. Instead of instantiating an ideal service and searching through the reasoner for the closest "real-world" service, DSD specifies a service request with a fuzzy set of acceptable services and thus copes with user's preferences. In other worlds, it copes with what is really needed. DSD is used to solve one of the Semantic Web Services Challenge mediation problem in an original way: the mediation issue is viewed as a discovery issue. The DIANE middleware both captures service requests, and service offers, performed by stakeholders, and its matchmaker discovers suitable services to handle the requests. The DIANE middleware can also handle the invocation of services through grounding information of the service description offer and through their proper data lifting/lowering between XML messages and ontological DSD data. In order to promote Semantic Web Services sharing and evaluation, the OPOSsum portal [Küster et al., 2008] has been developed and allows, among others, categorizing services, editing semantic annotations based on several standards (OWL-S, WSML, WSDL).

5.2.3 Applied Semantic Web Services

In the context of the SIMDAT Pharma Grid project, Qu and co-workers propose a semantics-enabled service discovery framework based on OWL-S descriptions and ontology reasoning [Qu et al., 2008]. The service discovery component is a major requirement for biologist end-users to conduct *in silico* experiments, at a high level of abstraction. OWL-S has been chosen for its maturity in comparison with WSMO and for its expressivity in comparison with WSDL-S, from which has been derived SAWSDL. The authors highlight that "non-semantic" information, for instance non-functional infrastructure requirements such as site memory or bandwidth need to be added in an early stage of the service discovery process in order to avoid time consuming reasoning.

Built upon the ^{my}Grid ontology, a bioinformatics service and domain ontology, FETA [Lord et al., 2005] is a service discovery framework characterized by a light-weight semantic support and a semi-automatic approach. Three main actors are distinguished in this framework: both *knowledge engineers* and *service annotators* provide semantic enhanced web services consumed by *scientists*. During the annotation process, services are registered in a UDDI repository (used as an intermediate component within the FETA architecture) and an XML description file is produced to link the service to concepts of the ontology. To cope with the light-weight requirements, they avoid usage of Description Logics reasoning and predefine a set of RDF queries covering the major needs of the bioinformatics community. Feta is part of the ^{my}Grid technologies and referenced in a work focusing on discovery and reusing of scientific workflows [Wroe et al., 2007]. In a bioinformatics context, the authors highlight that for usability purpose they do provide full description logic reasoning. The ^{my}Grid ontology is described in [Wolstencroft et al., 2007] with relevant information on the usability of semantic technologies by scientists to build in silico experiments. In particular, two kind of services are distinguished, namely *domain* services aimed at performing a scientific function, and *shim* services ("helper" services) aimed at gluing *domain* services. While complex Description Logics reasoning is needed to precisely and automatically select *shim* services aimed at resolving mismatches,

simple RDFS descriptions are more suitable to service discovery. Indeed, RDFS descriptions are simpler and it is preferable to retrieve a shortlist of candidate services and let scientists realize the final decision. Authors also reported that OWL created a barrier of adoption for the user communities and that OWL flavors should be hidden to the end-users.

Belhajjame & al. overview in [Belhajjame et al., 2008b] the management of Metadata in the scientific workflow design and enactment workbench, Taverna. Metadata are presented as valuable structured and descriptive information about (i) workflows and their related entities, and (ii) workflow executions. The main objectives are the enhancement of the components of workflow systems and the usage of such systems. The classification of workflow entities with domain specific concepts is described. Annotations are proposed to cover both tasks and parameters of processing units with benefits on focused (domain based) browsing and also on accurate description by domain experts. Additionally, *parameter instances* are proposed to populate a repository of test data to prevent regressions, and assert availability and reliability of processing units. Metadata describing the workflow execution, namely provenance metadata, is also described through both process and data point of view. The metadata life cycle is described through *creation*, *maintenance* and *curation* tasks. It takes into account third-party constraints such as service description importation, metadata storage in RDF/XML repositories, references to external domain ontologies. Semantic Web Services challenges and provenance metadata usage are investigated through an implementation within the Taverna workbench.

Semantically annotating services is a time consuming task. Belhajjame & al. propose an automatic annotation framework based on existing annotated workflows from the ^{my}Grid repository [Belhajjame et al., 2008a]. The derivation of a set of annotations for input or output parameters consist in inferring from existing compatible connections between outputs and inputs, the semantics type candidates. An annotation algorithm is proposed and as a favorable side effect, the detection of conflicts lead to the curation of already annotated workflows, by excerpting errors, in workflows or in annotations. An annotation workbench as been implemented, covering both manual annotation (enhanced by hints with derived annotations), annotation conflicts identification and resolution, and workflow inspection.

The BioMOBY project aims at providing interoperability for biological data centers and analysis center. SAWSDL has been used in this context and this real-world application is one of the few existing initiatives [Gordon and Sensen, 2008]. The focus is on interoperability and therefore on schema mapping annotations of SAWSDL, implemented through XSLT stylesheets. The entry-point is a SAWSDL Proxy servlet, in front of a web service provider, a semantic registry, and a schema mapping server.

Larvet and co-workers propose in [Larvet et al., 2008] a process covering the semantization of legacy web services using SAWSDL. The framework assists the service developer by providing a way to add semantic tags to the service inputs, outputs, or to its goal, avoiding manipulation of ontologies or XML files. Few details are shown about concepts retrieving and services publishing.

5.2.4 Positioning our contributions towards Semantic Scientific Workflows

Approaches	Formalism	Challenges: Annotation	Challenges: Discovery	Challenges: Composition	Challenges: Mediation	Targeted end-users	Year
WSMO	•	x	x	x	x	s	2005
OWL-S	•	x	x	x	-	s	2004
Taverna	•/+	x	x	x	-	d	2008
Lécué & al.	+	-	x	x	x	s	2008
METEOR-S	+	x	x	-	x	s	2008
hRESTs	+	x	x	-	-	s	2008
Cardoso & al.	+	x	x	-	-	s	2008
OWLS-MX	+	x	x	-	-	s	2009
SAWSDL-MX	+	x	x	-	-	s	2009
SIMDAT	+	x	x	-	-	d	2008
FETA	•/+	x	x	-	-	s/d	2005
BioMOBY	•	x	-	-	x	d	2008
CORESE	+	-	x	x	-	s	2005
DIANE	+	-	x	-	x	s	2008
Derouiche & al.	+	-	-	x	x	s	2007
SAWSDL	+	x	-	-	-	s	2007
WSMO-Lite	+	x	-	-	-	s	2008
Yasa4wsdl	•	x	-	-	-	s	2008
Larvet & al.	+	x	-	-	-	d	2008
Vouros & al.	+	x	-	-	-	s	2008
Selvi & al.	+	-	x	-	-	s	2006
FUSION	+	-	x	-	-	s	2008

Table 5.1: Despite most approaches cover annotation and discovery, none of them address all challenges and target the domain experts.

Through a large number of initiatives, this review shows a huge interest in enhancing web services with semantic web technologies. Table 5.1 categorizes the existing approaches by highlighting the tackled challenges. Criteria also cover the use of proper (•) or external (+) formalisms (language, model or ontology), and the targeted end-users: service experts (s) or domain experts (d).

A consensus appears in relying on the SAWSDL standard to bridge the gap between traditional web services and semantically enhanced services. Such a technological consensus could minimize the software development efforts and foster its adoption. How-

ever SAWSDL features cannot, alone, tackle Semantic Web Services challenges. Reasoning engines and precise enough descriptions of services are needed, being originally addressed by OWL-S and WSMO approaches. Domain experts seemed to be reluctant to adopt the WSMO environment and rather preferred the lighter framework OWL-S, which aims at modeling first and which leans on third party OWL reasoners.

Section 5.2.3 focuses on approaches driven by demanding user communities, especially from the Bioinformatics area. A lot of work has been achieved and should be considered for Semantic Web Services applications in the Neuroinformatics or Medical imaging fields. Approaches should rely on SAWSDL and OWL-S for a technical description of services, and additionally, a domain ontology to address at least service classification through their functionality, and parameters compatibility checking at workflow design-time.

Scientists involved in the NeuroLOG and VIP projects are building medical imaging domain ontologies, modeling knowledge on manipulated medical imaging data, simulated images, simulation parameters, and knowledge on their dedicated processing or simulation tools, exposed as services. They also rely on large-scale grid infrastructure to support storage- and compute-intensive tasks described in their workflows similarly to *"in silico"* experiments in the SIMDAT Pharma Grid or workflows in the ^{my}Grid environment.

Our approach, developed in chapters 6 and 8 globally addresses Semantic Scientific Workflows (RQ_4, RQ_5). We are in line with a lightweight top-down approach such as OWL-S to semantically associate domain knowledge to workflow activities through semantic annotations for both the service functionality, and its parameters. However, for ontological integration issues – OntoNeuroLOG and OntoVIP are both grounded to the DOLCE foundational ontology – it has been preferred to ground the service modeling to foundational concepts, which resulted in a small sub-ontology of the OntoNeuroLOG modeling framework, dedicated to Web Services Modeling. Still, our implementation is flexible enough to provide, with limited development effort a service annotator providing OWL-S service descriptions.

5.3 Provenance in scientific workflows

Interest in provenance is growing in the e-Science area since the scientific community agreed on its importance to enhance the overall quality of resulting data and the quality of analysis procedures themselves. If we focus on the neuroimaging area, medical image acquisitions are generally acquired as brain scans and stored inside the PACS[3] of radiology departments. These clinical datasets embed a large variety of descriptive metadata such as nominative information (patient name, birth date, clinician and physician names, eventually a reference to a clinical study identifier, *etc.*) or physical acquisition information describing the acquisition procedure or the physical parameters of the medical imaging equipment. To be used in the context of medical imaging research, these clinical

[3]Picture Archiving and Communication System.

data sets are generally pre-processed and normalized. The data pre-processing usually anonymizes (or pseudonimizes[4]) clinical datasets, and normalizes them so that datasets become comparable in the context of scientific studies. This is an example of information loss that can be retrieved back if provenance information is carefully kept. Moreover, provenance data is needed to maintain connections between both the pre-clinical area (involving scientific data) and the clinical area (involving patient data) and give a chance to associate back resulting scientific data to patients.

Moreover, scientific activities achieved through e-Science platforms are generally considered as competitive, and provenance gained also a lot of interest since it helps in determining the ownership of the resulting scientific outcomes. But beyond these ownership concerns, provenance is a key element towards enhanced data analysis quality, enhanced "*in silico*" experiments reproducibility. Provenance management for e-Science platforms is promising for large and various scientific communities.

5.3.1 Scientific workflows

Scientific workflows are nowadays the *de facto* formalism to represent and conduct *in silico* experiments through e-Science platforms. Bowers reviews in [Bowers, 2012] the main challenges and approaches for the Scientific workflow and Provenance areas (an extensive survey on scientific workflows has been published in [Deelman et al., 2009]). While complex data analysis procedures was traditionally based on scripting approaches (Shell, Python, or R scripts), scientific workflow environments possibly allows for user assistance at workflow design-time, monitoring at workflow run-time, dynamic optimizations while benefitting from data parallelism, or workflow instrumentation to dynamically track, for instance, provenance information. For the most adopted approaches, we could cite the Taverna [Hull et al., 2006] workbench initially dedicated to the bioinformatics area, Kepler [Altintas et al., 2004], Wings for Pegasus [Gil et al., 2007], or the Loni pipeline [Dinov et al., 2009].

Designing scientific workflows can be seen as coarse-grained programming since it requires, for the designer, to compose together data processing activities through data dependency links. Workflows generally adopt graph-based representations where nodes consist in either atomic or composite[5] data processing activities and edges consists in data links connecting together activities through their exposed ports[6], thus providing pipelines in which large amount of data can be processed. Such data-oriented approaches are attractive, specially in the context of distributed computing infrastructure since they offer implicit parallelism, and a way to efficiently achieve compute-intensive experiments.

[4]While anonimization removes all nominative information, and more generally all information that can lead to the patient identity, pseudonimization transforms nominative information so that the dataset is linked to a virtual identity.

[5]While atomic activities wrap a unitary data processing as a black box, composite activities are sub-workflows wrapping other activities and data connection links as a grey-box.

[6]From a processing activity perspective, input ports represent a specific data consumption, and output ports represent a specific data production.

Based on the Gwendia Workflow language [Montagnat et al., 2009], the Moteur [Glatard et al., 2008] workflow designer and enactor proposes innovative iteration strategies to address complex data flows. Iteration strategies [Montagnat et al., 2009, Sroka et al., 2010] define how input data may be combined when they are consumed by an activity through input ports.

Role of provenance in scientific workflows. We have just seen that scientific workflows are a common mean to achieve large-scale *in silico* experiments. They may lead to a huge amount of data processing activities, potentially re-located on large-scale distributed infrastructure. Moreover we have seen that, through possible iteration strategies, scientific workflows may result in complex data combinations and execution plans. For all these reasons, provenance in scientific workflows is crucial to (i) enhance the quality of produced data, (ii) enhance the quality of the scientific workflows themselves, (iii) finally enhance the reproducibility of e-Science experiments through a better interoperability of workflow environments.

5.3.2 Domain-agnostic provenance

The NeuGRID european project aims at providing a computing infrastructure dedicated to the study of neurodegenerative diseases such as the Alzheimer disease. NeuGrid relies on grid infrastructures to achieve storage- and compute-intensive data analysis and processing. To enhance the reproducibility of data analysis/processing and to guarantee the ownership of resulting scientific outcomes, these large-scale analysis rely on the CRISTAL workflow and provenance environment [Anjum et al., 2011]. Through the NeuGRID portal, end-users are able to author LONI [Dinov et al., 2010] pipeline or KEPLER [Altintas et al., 2004] workflows. Workflow descriptions are then translated into the CRISTAL internal format. Workflows are then enacted on the grid infrastructure and synchronized to a virtual instance of the workflow which simulates its execution to finally capture provenance information. Once the execution is terminated, provenance is stored in a relational database. The main disadvantage of this approach is that the NeuGrid-specific model for provenance does not rely on standards and makes it difficult to exchange provenance information outside the NeuGrid boundaries.

[Madougou et al., 2012] propose an e-infrastructure for biomedical research, e-BioInfra, enhanced with a provenance store populated from a *post mortem analysis* of the Moteur workflow enactor logs. The proposed provenance store is loosely coupled to the Moteur workflow environment and thus suitable for other workflow enactors. The proposed system populates a relational SQL backend with OPM provenance statements, queried through the HQL (Hibernate Query Language). In spite of the maturity and scalability of relational database management systems, the graph representation of provenance is lost into a relational representation, and the approach can not benefit from graph-based querying languages such as SPARQL . Moreover, only domain-agnostic provenance is considered in this approach which makes it generally difficult to comprehend, due to its size and genericity, from the end-user perspective. However, the future semantic en-

richment of the e-Bioinfra components (data, services, workflows and executions) would certainly enhance the exploitation of provenance information through domain-specific annotations and queries.

A similar approach is the RDFProv [Chebotko et al., 2010], a relational RDF provenance store focusing on scientific workflows. The proposed system mainly addresses scalable semantic provenance data management by, on one hand, exploiting the expressivity, the reasoning capabilities of semantic web standards, and on the other hand, benefitting from the power, the scalability and the general maturity of relational database management systems. The system adapts to any provenance model expressed as an OWL-DL ontology. The scalability of the approach results from indexing strategies of provenance storage in relational database, and mapping from SPARQL to SQL queries to efficiently query the RDFProv store.

The ProvBase [Abraham et al., 2010] also addresses the scalability of provenance information by considering highly distributed storage and querying techniques. Google's Bigtable technology provides a reliable web-scale cache mechanism. In line with Bigtable, Abraham and co-workers study how the HBase[7] open-source implementation of Bigtable, on top of the Hadoop[8] intensive data communication framework, can provide massive scalability for large provenance datasets. HBase tables differ from traditional relational tables since their columns and rows are stored in separate data structures, allowing to partition rows in regions that are spread over the distributed infrastructure. ProvBase propose a specific provenance storage schema suited for HBase stores and for efficient matching of SPARQL triple patterns. The proposed schema consists in three tables storing provenance statements, and differing in their row keys. The first one has RDF *subjects* as row keys, the second one has *predicates* as row keys and the last one has *object* as row keys. ProvBase proposes in addition an algorithm of basic SPARQL querying based on row and value iterators. The evaluation of ProvBase is promising and new query rewriting optimizations are planned to incorporate already matched data and thus avoid useless data communications.

5.3.3 Provenance and interoperability

The *Open Provenance Model* [Moreau et al., 2011] initiative (OPM) is a community effort aiming at homogenizing the expression of provenance information on the wealth of data produced by e-Science applications. Among other things, OPM enables the exchange of provenance information between several workflow environments. It eases the development of tools to process such provenance information, and finally facilitates the reproducibility of e-Science experiments. OPM is materialized through a natural language specification and three formal specifications: an XML schema (OPMX), an OWL ontology (OPMO) and a controlled vocabulary, with simpler OWL constructs (OPMV). OPM defines directed graphs representing causal dependencies between "things". A Causal dependency is defined as a directed relationship between an *effect* (the source of the edge)

[7] http://hadoop.apache.org/hbase
[8] http://hadoop.apache.org

and a *cause* (the destination of the edge). The nodes of the provenance graph might be either an *Artifact* (immutable, stateless element), or a *Process* (actions performed on *Artifacts* and producing new ones), or an *Agent* (entity controlling or affecting the execution of a *Process*). The edges of the graph represent (i) dependencies between two artifacts (*wasDerivedFrom*) to track the genealogy of artifacts (data lineage), (ii) dependencies between two processes (*wasTriggeredBy*) to track the sequence of processes, and (iii) dependencies between artifacts and processes (*used*/*wasGeneratedBy*) to track the consumption and the production of artifacts through processes. In addition, OPM allows to track the links between processes and their enactor agents through *wasControlledBy* dependencies.

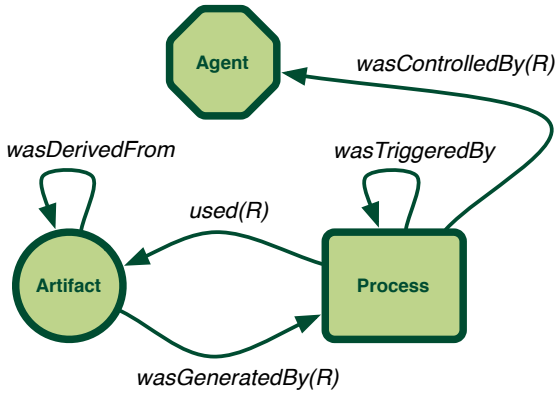

Figure 5.1: The main causal dependencies introduced in the OPM provenance model.

Figure 5.1 summarizes these causal dependencies. *Used*, *wasGeneratedBy* and *wasControlledBy* are parametrized with a label *R*. It is interpreted as a role in the specification which aims at syntactically distinguishing several dependencies of the same kind.

OPM recently made a step further since it evolved through a W3C standardization process towards the Prov-* specifications[9]. PROV-O is an OWL specification of the W3C provenance data model (PROV-DM), currently published as a working draft. It extends the *Open Provenance Model* with some classes and properties. For instance, PROV-DM introduces the notion of *Plan* to describe the context of execution of an *Activity* (the former *Process* OPM concept), which can be seen as a set of instructions, as a recipe, or a workflow.

Missier & Goble propose in [Missier and Goble, 2010] a lossless bidirectional mapping between Taverna workflows and OPM graphs. The objective is to promote the usage of a standard provenance model as a concrete mean towards interoperable workflow environments and thus a better reproducibility of e-Science experiments. The proposed round-trip between OPM and Taverna demonstrates the feasibility of OPM graph transformations directly into Taverna workflows and conversely. This contribution open new opportunities for Taverna to "replay" e-Science experiments enacted on other workflow environments based on their OPM provenance traces, OPM thus providing a pivot model for workflow interoperability. Similar efforts addressing the mapping between OPM and

[9] http://www.w3.org/TR/prov-primer

other workflow environment would considerably enhance workflow interoperability and consequently e-Science experiments reproducibility.

The *Linked Provenance Data* initiative [Ding et al., 2010] addresses (i) provenance representation interoperability by leveraging domain knowledge through the OWL Semantic Web language, and (ii) provenance infrastructure interoperability by relying on Linked Data approaches. More precisely, application-level queries require both domain-agnostic and domain-specific provenance, and potentially multi-source provenance data needed to be interconnected to further be aggregated and queried. Ding and co-workers propose, (i) in line with the OPM 1.1 *Profiles*, a taxonomy of OPM subconcepts to more precisely model domain entities, and (ii) the use of owl:sameAs properties to declare equivalent entities, thus easing the comparison of workflow traces. The *Linked Provenance Data* infrastructure stores directly provenance data in RDF (or converts XML traces) and publishes it on the web through Linked Data principles (RDF and de-referencable URIs). Provenance data may finally be queried through the SPARQL Semantic Web query language.

5.3.4 Meaningful, domain-specific provenance

The Wings/Pegasus environment [Kim et al., 2008a] addresses through semantic reasoning on application-level constraints, the generation of valid and execution-independant workflows, finally enacted over distributed computing infrastructures. In addition, the proposed system is able to produce both application-level and execution provenance. Wings/Pegasus uses a proper OWL ontology to model application-level provenance data and uses a provenance tracking catalog, based on a relational database, to record execution provenance. Two language are thus required to query provenance data, SPARQL for application-level (and thus domain-specific) provenance, and SQL for execution, domain-agnostic, provenance.

Nowadays, a lot of scientific data publication initiatives rely on the RDF Semantic Web standard. The PaCE (Provenance Context Entity) approach [Sahoo et al., 2010] aims at representing provenance of RDF triples directly through the RDF Semantic Web Standard, in a concise form. The RDF reification vocabulary has been initially introduce to track provenance in RDF documents through the following terms : rdf:statement, rdf:subject, rdf:predicate, and rdf:object. Such reification generally makes use of RDF blank nodes which have no sense outside the context of a particular RDF graph, and reified statements makes the reasoning difficult through RDF entailments [Hayes and McBride, 2004]. Moreover, the reification vocabulary is very verbose to express a single relation between two RDF statements. Through the reification terms, 4 statements are needed to describe a single RDF triple. To address these issues, in the PaCE approach, provenance is defined through the Provenir upper-level ontology [Sahoo and Sheth, 2009] modeled with the OWL-DL profile of the Web Ontology Language. Provenir provides the minimal OWL classes and properties (3 classes and 10 relations [Smith et al., 2005]) allowing for RDF and OWL inferencing. Provenir conceptualize the notion of *Data*, *Agent* and *Process* similarly to OPM *Artifact*, *Agent*, and *Process*. e-Science applications can define domain-agnostic provenance through the Provenir ontology, or domain-specific provenance, by

extending Provenir with domain-specific concepts. Compared to the reification approach to track provenance in RDF documents, the usage of the Provenir ontology allows for domain-specific provenance and allows for much more scalable provenance data management.

In *Janus*, [Missier et al., 2010] introduce semantic provenance as technical provenance graphs coupled with domain knowledge. The main objective is to enhance the usefulness of provenance graphs in responding to typical user queries. Semantic provenance was first introduced by [Sahoo et al., 2008], without any concrete implementation. Missier and co-workers propose with *Janus* a domain-aware provenance model by extending the Provenir upper-level ontology [Sahoo and Sheth, 2009] grounded to BFO (Basic Formal Ontology) concepts, and a prototype implementation within the Taverna workflow workbench. In *Janus* the modeling of domain entities relies on four ontologies registered in NCBO, the National Center for Biomedical Ontologies, namely the BioPAX (dedicated to the modeling of biological pathways), the National Cancer Institute (NCI) Thesaurus, the Foundational Model of Anatomy (FMA) and the Sequence ontology. Once web services composed into Taverna workflows are semantically annotated, simple inference rules for each service run are responsible for the propagation of semantic annotations to the produced domain-agnostic provenance, thus providing new domain-specific (meaningful) provenance. To answer provenance queries, a specific transitive closure implementation was proposed based on low-cost SPARQL ASK queries.

5.3.5 Positioning the provenance-based e-Science experiment summaries

The approach defended in this thesis (detailed in chapters 6 and 8) towards Semantic Scientific Workflows and provenance-based experiment summaries (RQ_4, RQ_5), relies on (i) standards models resulting from a community effort (the OPM provenance model), and (ii) Semantic Web standards allowing for the querying and reasoning on execution provenance, to finally infer concise domain-specific annotations on produced data.

Compared to provenance management in NeuGrid and e-BioInfra, our approach preserves the underlying graph structure of provenance through the RDF format, thus allowing for graph-based querying and reasoning. However, the technical provenance storage is possible through the Jena Semantic Web persistence facilities, namely SDB for a relational backend, and TDB for an optimized binary file storage. When using Jena SDB, we thus benefit from the maturity of relational backends while still allowing for graph-based querying through SPARQL . Our approach is in line with RDFProv, but does not require complex mappings to rewrite SPARQL queries to SQL . Moreover since provenance data is stored in RDF and identified through URIs, its publication as Linked Data is facilitated, and opens perspectives to exchange and reproduce experiments outside a particular collaborative life-science communities (NeuroLOG or VIP in the context of this thesis).

To address scalability issues, we propose, through meaningful experiment summaries, concise domain-specific statements inferred from OPM graphs captured at workflow runtime. The main idea is the following. Domain-agnostic fine-grained OPM prove-

nance graphs are useful at workflow design time, or for workflow debugging activities, but such fine-grained information might be annoying and difficult to comprehend for end-users launching *in silico* experiments and expecting their data to be processed, and easily retrieved. Thus we propose to store fine-grained, domain-agnostic OPM graphs, in a short term repository dedicated to workflow design and debugging, and inferred (domain-specific) meaningful experiment summaries in a long term repository dedicated to the exploitation, at long-term, of data produced through e-Science experiments. Our approach is in line with the PaCE which tends to minimize, through domain-specific statements, the size of the provenance graphs. Modern noSQL approaches are promising. This is the case for the HBase schema proposed in ProvBase, and dedicated to the distributed and highly scalable storage of provenance statements. However, complete SPARQL querying needs still to be investigated since only basic querying is supported over HBase tables.

Compared to meaningful provenance approaches, our work provides domain-specific execution provenance whereas the approach promoted in Wings/Pegasus only provides domain-specific provenance annotation at workflow design-time (queried through SPARQL), and relational provenance at workflow run-time (queried through SQL). It is thus not possible to perform rich semantic querying and reasoning on execution provenance. Moreover we rely on a single graph-based representation format, RDF, for both fine-grained domain-agnostic provenance and coarse-grained meaningful provenance. However, reasoning on semantic annotation at workflow design-time is interesting and we plan to couple provenance annotations to conceptual workflows as introduced in [Cerezo and Montagnat, 2011]. Our approach is in-line with the PaCE approach which promotes the usage of the Provenir upper-level provenance ontology, and its extension through domain-specific concepts subsumed by Provenir concepts. Motivated by the community effort promoting OPM as a standard model for provenance, we finally adopted OPM. Relying on domain-specific ontologies grounded to the DOLCE foundational (upper-level) ontology, it would have been interesting to benefit from a provenance ontology also grounded to a foundational ontology, such as Provenir, to provide a more conceptually coherent integration of domain-agnostic provenance and domain-specific provenance.

Janus is definitely the closest approach to our proposal for meaningful e-Science experiment summaries. The main differences are the use of the OPM and medical imaging ontologies (OntoNeuroLOG [Gibaud et al., 2011b] and OntoVIP [Forestier et al., 2011b] grounded to DOLCE [Masolo et al., 2003] as foundational ontology) in our work, compared to Provenir and biomedical ontologies in *Janus*. To address scalability issues, we propose to make a clear distinction between short-term fine-grained domain agnostic provenance and inferred long-term domain-specific provenance through the notion of meaningful experiment summaries. *Janus* extends domain-agnostic provenance with domain specific statements, which requires to manage the large amount of fine-grained provenance along e-Science experiment runs. Moreover we rely in this thesis on the KGRAM Semantic Web query and reasoning engine, fully compatible with the SPARQL 1.1 standard thus allowing to publish and query provenance data captured through the

NeuroLOG and VIP platform as Linked Data.

5.4 Conclusion

We studied in this chapter the current approaches aimed at building semantic workflow environments.

Considering scientific workflow environment as service oriented architectures, we review in section 5.2 how web services can bridge the gap with domain knowledge to enhance their discovery, their composition through orchestrations (workflows), and broadly speaking, their overall usage. We reviewed top-down approaches starting from services models towards technical service descriptions, namely OWL-S, WSMO or FLOWS [Gruninger et al., 2008] ; followed by bottom-up approaches starting from technical service descriptions such as WSDL, towards semantically annotated services, namely WSDL-S, SAWSDL, WSMO-Lite or METEOR-S for the most famous. We finally reviewed application-driven approaches in which a lot of work has already been achieved, specially targeting the bioinformatics area. The application-driven approaches promote both SAWSDL and OWL-S for making web-services benefitting from domain knowledge.

Section 5.3 reviews the current approaches for managing provenance information associated to scientific workflow runs. We reviewed first domain-agnostic provenance approaches, mainly focusing on scalability issues. Then, we described how standardized provenance model can help to enhance the reproducibility of e-Science experiments. We described finally provenance approaches aimed at producing domain-specific information (Wings/Pegasus, Provenir, Janus) representing meaningful information on workflow runs.

Towards Semantic Web Services in the context of *in silico* neuroimaging experiments, we propose through chapter 6 to produce new semantic annotations based on workflow runs. More precisely, we show (i) how semantic annotations associated to service ports can be disambiguated through *Role* concepts, and (ii) how disambiguated service annotations, coupled to fine-grained domain-agnostic provenance and domain-specific inference rules, can lead to the production of new meaningful statements. Chapter 7 describes how the system has been implemented and chapter 8 provides an evaluation of domain-specific provenance, distributed through Linked Data principles, which finally provides experiment summaries to end-users.

Key Points

- *Lightweight ontologies such as OWL-S are popular to semantically annotate services.*
- *The W3C proposes standards for semantically annotating web-services, SAWSDL, and for representing provenance, Prov-DM "derived from" the OPM community effort.*

- *Data processing services involved in MOTEUR workflows, and annotated through the DOLCE-based web-service sub-ontology of OntoNeuroLOG could easily by semantically annotated though the OWL-S service ontology.*
- *Domain-agnostic fine-grained provenance is too much technical, and too much verbose to address end-user concerns. Coarse-grained domain-specific provenance has to be considered with a high interest.*
- *Scalability of provenance is a real concern. When provenance is published as Linked Data, distributed querying should enhance the scalability.*
- *Publication of provenance as Linked Data, and through standards, is an opportunity towards better e-Science experiment reproducibility.*

CHAPTER 6

Semantic scientific workflows for knowledge capture and extension

Contents

6.1 Introduction

Semantic web services, introduced in section 5.2, make the nature of both processed data and applied information processing explicit, thus bridging the gap between syntactic descriptions and domain knowledge. Through enhanced service discovery, composition, or possible mediation between syntactical incompatibilities, semantic web services form a basis towards semantic workflow environments. Rather than focusing on how the design of scientific workflows could be enhanced through semantic services, we address in this chapter the exploitation of precise enough service descriptions to finally populate knowledge repositories based on workflow runs. The long-term objective is to foster data sharing and repurposing when conducting e-science experiments.

However, semantically describing services is critical to precisely identify the functionality of data processing or their input or output parameters. More precisely, general approaches such as SAWSDL, OWL-S, or WSMO provide a mean to attach domain

knowledge to services but may, in some cases, lead to ambiguities when semantically identifying the processed data.

The workflow illustrated in Figure 6.1 represents a typical image registration process commonly encountered in neuroimaging workflows. It consists in superimposing two medical images acquired independently into the same coordinate system. The sample registration process is decomposed into two steps. First, the registration itself consists in calculating, from the input brain MRI and a brain atlas, a geometrical transformation expressed by a transformation matrix. Second, the resampling step effectively aligns the input brain MRI by applying the transformation expressed through the registration matrix.

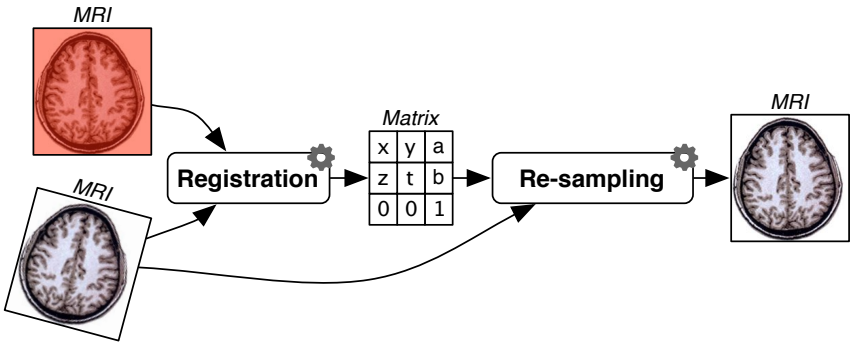

Figure 6.1: A typical neuroimaging workflow mixing several nature of data and processing.

In spite of its apparent simplicity, this workflow mixes several natures of data and processing. It may face ambiguity when semantically distinguishing the two input parameters (left part of the figure) if we only consider the nature of data or processing.

- First, this workflow mixes two services of different nature, whose meaning has been agreed upon within the neuroimaging community. In other words, the knowledge about what kind of underlying information processing is clear for the community but is generally not explicit at the tooling level.
- Second, this workflow consumes and produces data of several natures (medical images, transformation matrix) expressed through raw files at the tooling level. Again, these files have a precise meaning from the user community point of view, with regard to their content.
- Finally, the first step of the workflow takes two files as input, sharing the same nature, both are brain MRIs, but playing different roles from the processing tool perspective. The first one is used as the reference image for the registration process (atlas) whereas the second one is used as the floating (moving) image.

Even if the several natures of data and processing allow to semantically distinguish most of the entities involved in this workflow, it is not sufficient to distinguish the “atlas”

MRI from the "moving" MRI both considered as inputs of the registration step. Precisely characterizing the role of data from a service perspective would allow to disambiguate the semantic annotation of its parameters.

Provenance in scientific workflows has been introduced in section 5.3 as crucial information to enhance the quality of data analysis procedures and produced data through a better understanding of the detailed history of applied processing (also known as data/process lineage). Due to possible data iteration strategies[1], a single workflow execution may lead to a huge amount of fine-grained provenance information. Tracking all this detailed information is required (i) to help in identifying the cause of possible failures or abnormalities, and (ii) to enhance the overall workflow reproducibility. However, provenance produced through scientific workflow environments is generally technical and domain-agnostic, thus making it difficult to comprehend from an end-user perspective. Concise and meaningful information, summarizing complex and fine-grained provenance information, would be valuable for e-Scientists to better interpret, share or repurpose their *"in silico"* experimental results.

The simple workflow of Figure 6.1 also illustrates several level of semantic information:

(i) *domain-agnostic* information, related to the technical description of services or related to the service invocation which can later be used to produce provenance traces. For instance, through an OWL-S profile, the registration service from figure 6.1 would be described with the following properties: one *serviceCategory*, two *hasInput* and one *hasOutput*. Through the OPM provenance model, all causal dependencies represented by the arrows would be tracked as a set of *used* and *wasGeneratedBy* properties. These description are generic and are not yet related to any domain knowledge.

(ii) *domain-specific* information related to the *Nature* of the information processing performed and the *Nature* of the manipulated data. For instance, this would correspond in the previous example to the domain knowledge associated to the content of the data, *i.e.* "Brain MRI" for input and output data, or "geometrical transformation" for the intermediate matrix.

(iii) *domain-specific* information related to the *Role* played by the data involved in the service execution, from the service point of view. This level of information describes the differences between the two input parameters of the neuroimaging workflow, the first one considered as a reference "atlas" and the second one considered as the "moving image". This third level of semantic information allows to disambiguate the semantic annotation of service parameters and is thus needed to precisely reason on the service invocation effects on the processed data.

[1] A single workflow processing step may be executed several times when taking as input, and possibly combining, several data collections.

Leveraging existing ontologies to describe domain-agnostic information (service description and provenance) as well as domain-specific *Nature* of data and processing tools, this chapter focuses on the third level of semantic information to tackle:

- *knowledge capture* through a clarification of the bindings between service descriptions and domain concepts, using a taxonomy of domain-specific *Roles* (RQ_4).
- *knowledge extension* through the automated production of new concise domain-specific statements along e-Science platform usage, *i.e.* workflow runs (RQ_5).

The remainder of this chapter is organized as follows. Our approach is first motivated through a simple running example in section 6.2, and some background information are provided in section 6.3 which describes *Role* modeling and how roles could be considered in semantic web services and e-science workflow environments. Then, we introduce a domain-specific *Role* taxonomy in section 6.4, as an intent to enhance knowledge capture on neuroimaging processing tools. Based on the resulting disambiguated semantic annotations for neuroimaging processing tools, we provide a method in section 6.5 addressing the automated production of new meaningful statements through inferences on workflow provenance traces. Finally, results and perspectives are discussed in section 6.6.

6.2 Motivating use case

We propose in this chapter a methodology for producing and deducing new concise meaningful statements. If we consider the result of the registration workflow presented in Figure 6.1, it would be interesting to associate the atlas used as input in the registration process to the registered image produced. More generally, our approach address the propagation of the effect of services (or sub-parts of workflow) to the produced data. For instance, we would like to automate the generation of a fact saying that "a dataset can be superimposed with another one", because in some cases, processing tools might require that their input data are expressed in the same coordinate system, and thus have beforehand been registered.

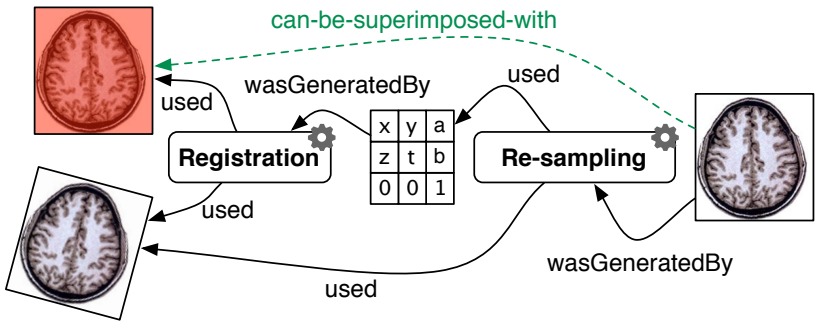

Figure 6.2: Linking data and processes through generic and domain-specific relations.

Figure 6.2 illustrates the causal dependencies established between entities involved in the sample workflow, *i.e.* data and services. Black arrows are relations tracked on-the-fly during each service invocation and forming provenance information. Provenance states data production and consumption. Beyond linking together data and process invocations (*i.e.* capturing provenance information) at a fine-grained level, we want to benefits from a domain ontology and its associated inference rules to automate the production of relevant domain-specific statements. For instance, the dashed arrow represents such information derived through a domain-specific inference rule embedding domain knowledge on the overall registration process.

As it will be shown in Section 6.4.2, this kind of inferences is possible if the semantic description of services is rich enough to properly define the *Roles* of processed data in the context of services invocation.

6.3 Background information

6.3.1 Role modeling

Conceptual or domain modeling generally consists in separating several categories of concepts, for instance those characterizing the nature of an entity from those characterizing their relations to each others. Henriksson and coworkers propose a methodology based on the design of role-based ontologies, extending standard ontologies, to enhance ontology modularization and reusability. They promote a clear delineation between *Natural Types* and *Role Types* [Henriksson et al., 2008] : *"In role modeling, concepts that can stand on their own are called natural types, while dependent concepts are called role types"*. Sowa first introduced *Natural Types* [Sowa, 1984] to describe what is essential to characterize the identity of an individual, and *Role Types* to describe temporal or accidental relations to other individuals. The methodology proposed by Henriksson coworkers consists in (i) identifying the natural types of the domain, (ii) identifying accidental or temporary relationships between individuals and ensuring that role models are self-contained (for reusability) and finally (iii) defining bridge axioms to bind role types to natural types (or to link individuals through properties defined in the role model). This approach is particularly interesting in our context since in life science ontologies, the design effort generally focuses on the first step. Moreover, e-Science experimental platforms are generally data-driven and well supported by ontologies describing the nature of data. However, few efforts concentrate in making explicit the knowledge relating data to their analysis services more deeply than just using information on data nature.

6.3.2 Roles in web service ontologies

Semantically enhanced e-Science experimental platforms usually rely on generic service ontologies to describe data analysis services. The following paragraphs briefly describe how these service ontologies consider the relations between data to service invocations.

The *OWL-S Profile* ontology [Martin et al., 2007a], one of the three ontologies forming the OWL-S proposal, aims at describing what the annotated service does. According to the OWL-S specification, nothing is said regarding how these parameter values are related to the service process and as a consequence, these types should be considered as *natural types* as they are defined in [Sowa, 1984]. To specify the relationship of parameter values to the process, it should be beneficial to rely, through the *parameterType* property, on a role ontology designed according to the methodology proposed by Henriksson and coworkers.

FLOWS [Gruninger et al., 2008, Battle et al., 2005] specifies a first-order logic ontology for Web Services. It aims at enabling reasoning on the semantics of services and their interactions. FLOWS has largely been influenced by OWL-S but in addition, it addresses interactions with business process industry standards such as BPEL. FLOWS differs from OWL-S by properly handling messages as core concepts. Messages are defined in FLOWS by a *message_type*, characterizing the type of the content, and a *payload*, representing the full content. FLOWS also defines three relations to relate atomic process invocations to messages they consume as input or they produce as output: *produces*, *reads*, and *destroy_message*. The relations are very generic and do not characterize more precisely the consumption/production of messages through domain-specific entities. However, FLOWS proposes the *described_by* relation to associate a *fluent* to a message. *Fluents* are used to model "changing" parts of the world. The *described_by* relation aims at providing information on how the content of the message impacts the service invocation while consuming/producing it. Intuitively, since *role types* are defined by Sowa as accidental (or evolving during time) relationships between entities, FLOWS's *fluents* could be a way to model how data are interpreted by analysis services through *Roles*.

WSMO [Roman et al., 2006] is a rich service modeling and enacting framework but it does not cover precisely the characterization of how processed or produced data are related to services in terms of roles. However, since WSMO relies on external ontologies, it remains compatible with any domain ontology designed using a clear separation between *natural types* and *role types*.

SAWSDL [Kopecký et al., 2007] is the W3C recommendation to semantically annotate WSDL and XML Schemas specifying standard Web Services. Again, this specification does not bring anything new to separate the *natural type* of the annotated WSDL message from how it is related to the Web Service (its *role type*). However, depending on the availability of an ontology of roles, *modelReference* attributes could be used to bind *role types* to service parameters.

The BioCatalogue [Bhagat et al., 2010] initiative is a community-driven, and curated service registry aimed at guiding users into a wealth of web services through the registration and annotation of web services and the browsing of resulting annotated web services. Several kinds of annotations are available going from free text, to tags or ontology terms. BioCatalogue allows, among other kind of annotations to *operationally* (e.g. infrastructure, runtime constraints) or *functionally* describe a service. Functional annotation covers information related to what the service does, but also its function and the format of input or output data. The function annotation of data with regard to a given

web service seems to be close to *Role types* previously introduced but few information is available to precisely describe this kind of annotation.

As a continuation of the BioMOBY initiative, the SADI project [Withers et al., 2010] proposes guidelines and best-practices to enhance semantic service discovery at workflow design time. Semantic services are indexed in the catalog through the new set of RDF properties describing the resulting new semantic features associated to input data. The service discovery is based on searches over properties of data consumed as input and over the new properties produced. This approach also aims at reducing ambiguity of search queries through more precise properties, describing the relationships between input and output data.

All these approaches do not really address the distinction of service parameters sharing a similar nature but acting differently from the service perspective. Few approaches tackle this issue through specific representations such as FLOWS *fluents* or BioCatalogue *functions*. We propose, in the following section, to clearly make the distinction between *Role* and *Nature* concepts at domain ontology design time. *Roles* can then be used to disambiguate semantic service descriptions finally enabling reasoning and producing new meaningful statements from workflow runs.

6.4 Knowledge capture in neuroimaging data processing

6.4.1 Supporting ontologies

The OntoNeuroLOG ontology [Temal et al., 2008] is used as a semantic referent to query and retrieve heterogeneous data, as well as to annotate consistently the shared neuroimaging services, by denoting what sort of processing services actually achieve and what data they accept as input and produce as result.

OntoNeuroLOG was designed as a multi-layer application ontology for neuroscience, relying on a number of core ontologies modeling entities that are common to several domains. The whole ontology relies on DOLCE (Descriptive Ontology for Language and Cognitive Engineering) [Masolo et al., 2003], a foundational ontology that provides both the basic entities (at the top of the entities' taxonomy) and a common philosophical framework underlying the whole conceptualization. The ontology was designed according to the OntoSpec methodology, which focuses on the writing of semi-formal documents capturing rich semantics. This is followed by an implementation of a subset of the ontology in OWL, the web ontology language. The definition of this subset and the choice of the relevant OWL dialect take into account the specific needs of the target application. Two subsets of OntoNeuroLOG were used in the context of this work, the ontology of *Dataset* and the ontology of *Dataset processing*, introduced hereafter.

Dataset module. *Datasets* are *Propositions*, *i.e. non physical endurants*, that represent the content of data files used in neuroimaging. The taxonomy of *Datasets* is organized ac-

cording to several semantic axes. The first one denotes what facet of the subject is explored, *e.g. Anatomical datasets* explore the subject's anatomy whereas *Metabolic datasets* explore chemical processes happening in cells. The second axis classifies *Datasets* according to the common imaging modalities, such as *Computed Tomography* (CT), *Magnetic resonance* (MR), or *Positron emission tomography* (PET). This axis includes the numerous submodalities met, *e.g.*, in MR imaging such as *T1-weighted MR dataset*, *Diffusion-weighted MR dataset*, etc. The third axis focuses on Datasets that result from some kind of post-processing, such as *Reconstructed datasets*, *Registration datasets*, *Segmentation datasets*, etc.

Datasets may bear properties of *Representational objects* (since *Propositions* are *Representational objects*), such as *'refers-to'*, which denotes the ability to refer to any kind of *Particular*. This property can be used to refer to the *Subject* (*i.e.* the patient) concerned by a particular *Dataset*. For instance, a property called *'can-be-superimposed-with'* was introduced to relate two *Datasets* that can be superimposed with each other, such as a *Segmentation dataset* (*i.e.* an object mask obtained through a segmentation procedure) and the original dataset from which it was obtained.

Dataset processing module. *Dataset processings* are conceptual actions, i.e. *Perdurants* that affect *Datasets*. The taxonomy of *Dataset processings* covers the major classes of image processing met in neuroimaging, such as: *restoration*, *segmentation*, *filtering*, *registration*, *re-sampling*, etc. Axioms attached to each *Dataset processing* class usually denote which classes of *Datasets* are being processed or result of the corresponding processing. For example, a *Reconstruction* processing *'has-for-data'* some *Non-reconstructed dataset* and *'has-for-result'* some *Reconstructed dataset*, as well as a *Segmentation* processing *'has-for-result'* some *Segmentation dataset*.

Web Services module. In addition to these neuroimaging-specific modules, a domain-agnostic module has been defined to describe Web Services grounded to the DOLCE foundational concepts. It introduces the notions that are classically involved in WS specifications such as interfaces (*ws-interface*), operations (*ws-operation*), and service inputs and outputs (*input/output-variable*). Besides, the model benefits from the *'refers-to'* property (inherited from DOLCE) to establish relationships with the classes of data processing that a particular *ws-operation* implements (such as *rigid-registration* or *segmentation*), as well as with the classes (natural types) of entity that the input and output variables actually represent.

OntoNeuroLOG provides a coherent conceptual framework, grounded on the DOLCE foundational ontology, which allow to semantically represent data and services involved in neuroimaging workflows. Since the Web Service module remains domain-agnostic it can be reused beyond the scope of neuroimaging.

6.4.2 Role concepts

To benefit from expert knowledge conceptualized through domain ontologies (such as the OntoNeuroLOG ontology), services involved in e-Science workflows are manually associated to domain ontology concepts. Semantically annotating a service consists in using an ontology to bind technical concepts, *i.e.* elements syntactically describing services, to domain-specific concepts. Most of semantic web-services initiatives, namely OWL-S, WSMO, or SAWSDL, distinguish the annotation of the functionality of the service from the annotation of the service parameters which consume or produce data.

As an example, let us consider a medical image processing tool performing a de-noising operation. From a technical or syntactical point of view, the service is implemented by an executable binary taking as input a raw file materializing a noised medical image and producing as output another raw file materializing the resulting de-noised image. From a semantic point of view, this de-noising service implements a particular kind of algorithm characterizing how the image is processed. This "how" is described through the annotation of the functionality of the service, *i.e.* a particular class of restoration processing. The service additionally requires a specific medical image format (*e.g.* Analyze), and a specific modality of acquisition (*e.g.* ultrasound) to be successfully invoked. Moreover, the resulting de-noised image should preserve the input modality; in other words, even de-noised, the image still remains an ultrasound image. The service input/output parameters are usually annotated with concepts describing the nature of consumed or produced data. We will see in the following section that such semantic annotation on the nature of consumed or produced data is generally not sufficient to be precisely exploited, at invocation time, to produce new domain-specific annotations.

6.4.3 Differentiating neuroimaging Natural and Role concepts

To precise, and possibly disambiguate, the semantic service descriptions, service annotations should also make explicit how consumed or produced data items are related to the processes. For instance, if we consider the registration service involved in the workflow shown in Figure 6.1, both input parameters should share the same intrinsic nature. Indeed, in this example, the *reference* image parameter and the *floating* image parameter have been acquired both through the same Magnetic Resonance modality (MR) and should be materialized with the same file format. In this geometrical realignment procedure, the two input parameters are not distinguished by their intrinsic nature but rather by their relationship to the registration process, namely *"floating"* and *"reference"*. It is important to note that these two concepts only make sense in the context of a particular kind of image processing, the registration. Without the knowledge of "which data is acting as the reference image" or "which data is acting as the floating image", it is difficult to deduce any meaningful information from the execution of the registration workflow, such as "this resulting image can be superimposed with this reference image", or more generally to retrieve images that have been registered with the same reference and thus, that can be superimposed together. To tackle this issue we propose to distinguish *Natural*

concepts and *Role concepts* when annotating semantic service parameters by relying on a domain-specific role taxonomy.

Figure 6.3 illustrates the taxonomy of roles dedicated to the characterization of the relationships between neuroimaging data and their dedicated processing. *Role concepts* are organized following the main classes of neuroimaging processing similarly to the OntoNeuroLOG dataset processing ontology.

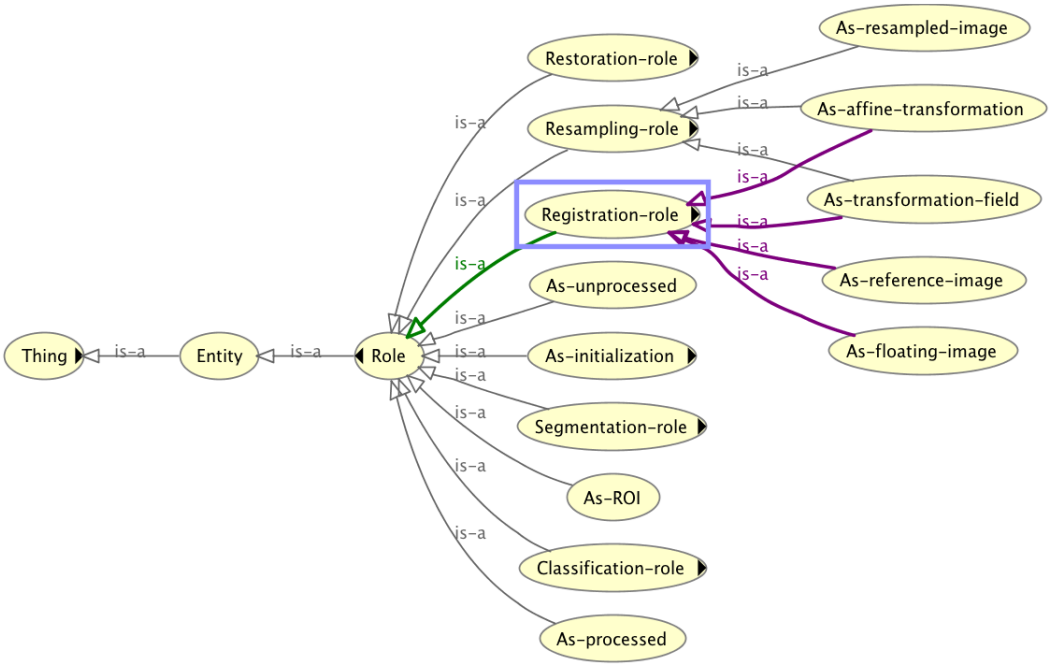

Figure 6.3: A domain-specific role taxonomy characterizing how neuroimaging data can be related to neuroimaging processing tools.

This taxonomy illustrates another example of disambiguation in the context of *resampling* processes. Indeed, the two roles *As-affine-transformation* and *As-transformation-field* precise how a matrix should be interpreted by a resampling process. If we consider two 3×3 matrices, they could share the same nature and representation format. However, one could be interpreted as a set of parameters for translation, rotation and scaling, in the context of an affine geometrical transformation, whereas the other one could be interpreted as a deformation field in the context of a non-rigid transformation.

Relying on this taxonomy of roles, we are now able to precisely annotate the input and output parameters of our image registration service considered in the running example (figure 6.2) with both *Natural* and *Role* concepts. Both input images are characterized by a same *Natural* concept, T1 weighted magnetic resonance image (T1-MR). T1-MR can be considered as a *Natural* concept because it stands on its own and does not characterize how input data are related with any other entities. On the other hand, service input parameters can be annotated with two distinct *Role* concepts to characterize how input data are related to the registration process. The service input parameter interpreting data as floating (the moving data, that will finally be realigned) is annotated with role *As-floating-*

image, and the second service input parameter interpreting data as the geometrical reference is annotated with role *As-reference-image*. Figure 6.4 illustrates the annotation with *Role* concepts for the two services involved in the full registration use-case workflow.

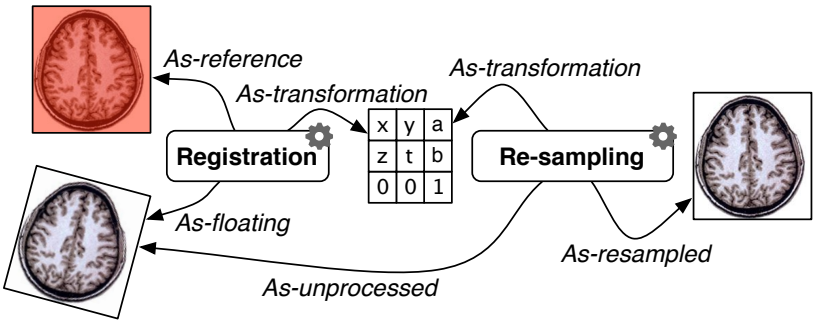

Figure 6.4: Roles involved in the registration workflow.

We have seen how the semantic annotation of services can be disambiguated through the notion of domain-specific *Roles*. We present in the following section how *Roles*, coupled with reusable inference rules (instrumenting domain ontologies) and workflow provenance, enable automated production of new meaningful statements.

6.5 Knowledge extension through semantic workflow runs

6.5.1 OPM provenance ontology

Workflow runs are described through provenance information tracked at runtime. Provenance is expressed through the OPM ontology [Moreau et al., 2011] (presented in section 5.3.3) which provides a modeling framework to represent causal dependencies between *Artifacts*, *Processes*, and *Agents*. But OPM graphs are very generic and verbose. Figure 6.5 illustrates the OPM statements captured from a single execution of our running example workflow. Although the sample workflow involves only two input data and two processes, the produced provenance graph resulting from its invocation is verbose (49 RDF triples) and is difficult to comprehend. *http://Registration* and *http://Resampling* represent the services which have been invoked. *#Registration-3e8e52...* and *#Resampling-b32e7296...* represent their invocation. Finally the medium sized *MyAtlas, MyInput, etc.* nodes represent the data entities involved either as input, output or intermediate data. Due to the reification of the relations in the OPM model, a lot of nodes and arcs are needed to interlink data, process invocations and services, thus leading to a verbose representation even in the case of a simple workflow.

To distinguish several causal dependencies of the same kind, OPM allows to annotate *used* or *wasGeneratedBy* dependencies with syntactic *roles*. A *Role* is defined in OPM as a particular function of an artifact (or an agent) in a process. The

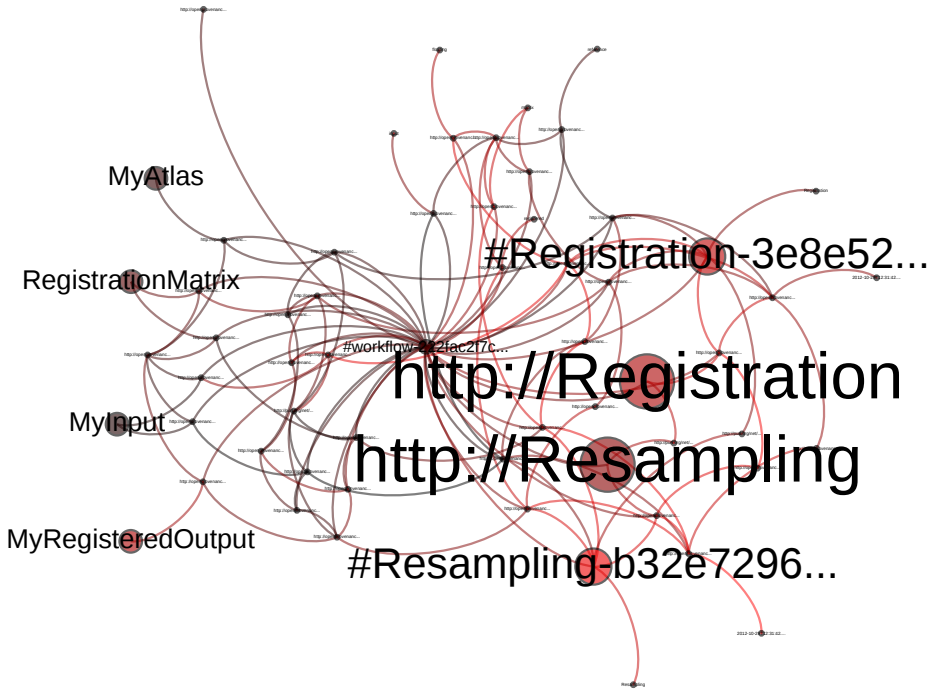

Figure 6.5: OPM provenance statements tracked from the invocation of the running example (Figure 6.1).

OPM model does not formally define roles but allows to associate labels to dependencies between artifacts (or agents) and processes. In OPM the execution of the sample registration process illustrated in Figure 6.1 could be translated with these two causal dependencies: "$\{Process_{Registration}, used(floating), Artifact_{Image}\}$" and "$\{Process_{Registration}, used(reference), Artifact_{Atlas}\}$". The syntactic roles "$floating$" and "$reference$" aim at distinguishing how artifacts are related to processes, but their meaning remains highly dependent on their usage within a given process, and thus, remains highly domain-specific.

In the following section we propose to exploit both the semantic roles introduced in figure 6.3, and the domain-agnostic and verbose provenance statements to deduce new concise and meaningful statements.

6.5.2 Reusable and service independent rules to infer new meaningful statements

The use of rule engines (or inference engines) is a well adopted data-driven and declarative approach to deduce/produce new conclusions/facts from a set of initial statements. In an OPM-instrumented execution engine, the invocation of services generates provenance statements such as the ones simplified in Figure 6.6. The graphical syntax introduced by [Moreau et al., 2011] is reused: *Artifacts* are represented by ellipses and *Processes* are represented by rectangles, *used* and *wasGeneratedBy* causal dependencies,

parametrized with roles, are represented by plain arrows.

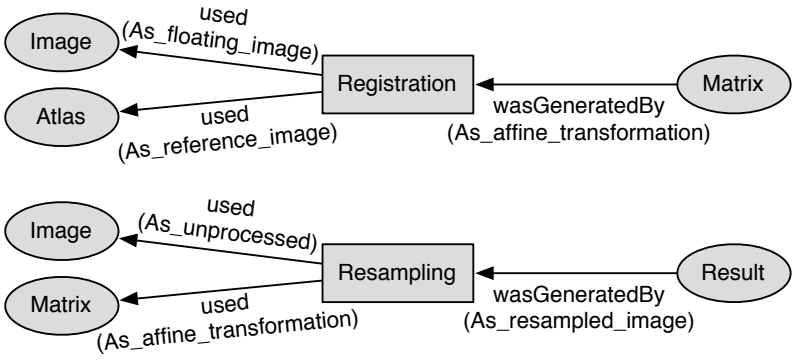

Figure 6.6: OPM statements tracked through the invocation of the registration workflow.

To automate the production of a statement linking the resulting data to the source data through a domain-specific property, an inference rule is proposed using the role-parameterized provenance causal dependencies. As an example, Figure 6.7 illustrates the inference rule deducing the *can_be_superimposed_with* property in the case of the registration workflow. The left part of the implication, the *antecedent*, corresponds to the *If* clause of the inference rule and consists in identifying a conjunction of statements necessary to produce the statements expressed in the *consequent*, at the right part of the implication (the *Then* clause of the rule). The first two lines assert that processes must refer, for the first one, to a Registration dataset processing, and for the second one, to a Resampling data set processing. In other words, the services invoked by the processes should have been annotated with the corresponding *Natural* concepts of the OntoNeuroLOG domain ontology. The two following lines of the *If* clause identify artifacts and processes through their *Role* concepts: the resulting image is identified through the *As-resampled-image* role, the registration matrix is identified through the *As-affine-transformation* role, and the reference image is identified through the *As-reference-image* role. Finally, when the reference image and the resulting resampled image are identified, the rule engine is able to produce a new statement saying that both images can be superimposed (*can_be_superimposed_with* property of the OntoNeuroLOG ontology).

Using *Role* concepts, domain ontologies can be instrumented with inference rules which remain service independent. Such inference rules can be reused in the context of several service implementations realizing a same kind of treatment. Let us consider the deployment of a new registration service, implemented with a new algorithm. As soon as this new service is annotated with *Role* and *Natural* concepts of the same class[2] as the concepts appearing in the registration inference rule, there is no need to rewrite an inference rule specific to this particular service. As a consequence, workflows involving this new service will also benefit from the generation of annotations stating the "superimposability" of data. With this approach, domain experts can enrich their ontologies

[2]or subsumed by

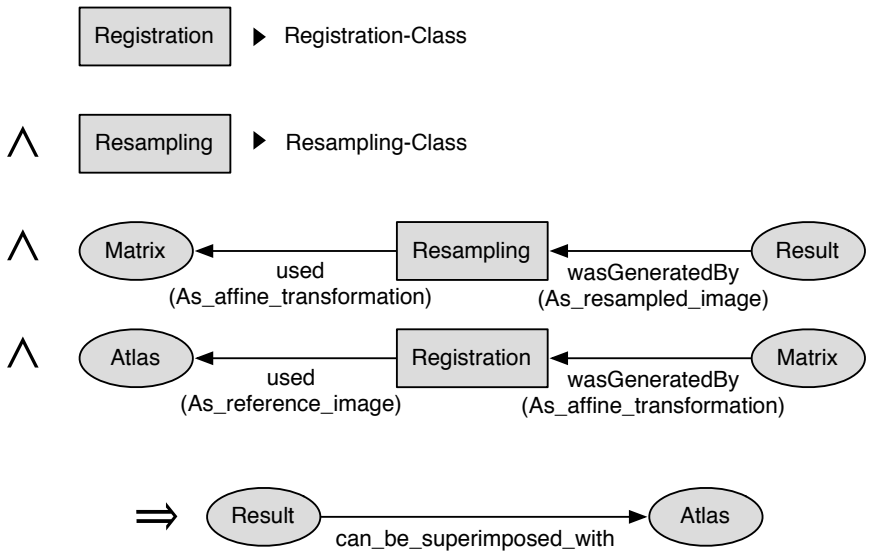

Figure 6.7: Reusable inference rule automating the annotation of superimposable images.

with inference rules that provide meaningful information to end-users independently from the services deployed. Service providers can focus on their services, transparently reusing these service independent inference rules.

6.6 Discussion and conclusion

E-Science experimental platforms strongly rely on service-oriented architectures to assemble data analysis workflows. However, their usability is hampered by the level of expertise required for experiment designers, as they are expected to have a clear understanding of the semantics of the data processing, *i.e.* what kind of data is processed and how they are effectively processed. To improve the usability of e-Science workflows, domain ontologies, semantic service annotations and reasoning engines have been integrated.

Regarding the sharing of the neuroimaging role taxonomy with other user communities, two approaches could be considered as a continuation of this work: (i) the creation of an OPM profile dedicated to the neuroimaging domain, or (ii) the articulation of the OPM ontology with the DOLCE foundational ontology.

OPM profiles constitute a good opportunity to share knowledge associated to the role of neuroimaging data. OPM profiles consist in a domain specialization of generic OPM graph, thus addressing domain-specific concerns. An OPM neuroimaging profile could be constituted with the two subsets of the OntoNeuroLOG ontology supporting this work, the *Dataset* ontology to extend OPM *Artifacts*, and the *Dataset-processing* ontology to extend OPM *Processes*. The Role taxonomy proposed in this chapter could be

integrated almost directly as sub-classes of the OPM *Role* class[3]. OPM recently evolved through a W3C standardization process which led to the PROV-* standards. Extensibility in PROV-* standards is not envisaged through profiles but through several extensibility points for PROV-DM (PROV data model), or specialization of PROV-O (PROV OWL ontology) classes or properties.

The second approach, more conceptual, would consist in proposing an OPM ontology whose main classes are grounded to foundational ontologies such as DOLCE or BFO (Basic Formal Ontology). It would allow to smartly articulate OPM and domain ontologies based on foundational ontologies such as BIOTOP (Top-Domain ontology for the life sciences) or OBI (Ontology of Biomedical Investigation) life science ontologies, and thus exploit these ontologies at workflow runtime. Indeed considering our approach from an ontology design perspective, a significant effort is still needed for a complete integration in the OntoNeuroLOG framework, since *Role concepts*, designated through the *refers-to* property, should conform to the DOLCE foundational ontology and its related core ontologies. This ontology integration task could also cover the semantic overlap between OPM *Artifacts* and OntoNeuroLOG *Datasets*. However, in the context of this work, this preliminary integration of OPM and OntoNeuroLOG still allows (i) retrieving service descriptions, or provenance statements through SPARQL queries and (ii) producing new meaningful statements through domain-specific inference rules.

We adopted in this chapter an integrative approach to address knowledge capture and extension in the context of e-Science workflows. This approach (i) promotes a clear delineation between *Role* and *Natural* concepts in domain ontologies to disambiguate the semantic annotation of service parameters (RQ_4), thus providing more accurate semantic service descriptions, and (ii) propose a methodology to instrument domain ontologies with inference rules that leverage both the description of *Roles* and domain-agnostic provenance information to finally extend semantic repositories with meaningful domain-specific annotations at runtime (RQ_5). Two main perspectives could be studied as a continuation of this work:

Modeling. As previously discussed, the modeling of domain-specific roles should be studied with particular attention, especially when domain ontologies are grounded to foundational ontologies such as DOLCE. Similar studies could also be envisaged when bridging provenance ontologies, such as OPM or PROV-*, with DOLCE-based domain ontologies.

Inference rules design. The proposed inference rules adapt to several service implementations as soon as they are semantically annotated with the same concepts (or sub-concepts) of the domain ontology. However, they remain highly dependent on the structure of scientific workflows. Workflow evolutions would lead to a necessary

[3] which was not present in the OPM 1.1 specification [Moreau et al., 2011] but was introduced in the OPM OWL ontology [Moreau et al., 2010].

adaptation of the inference rules. Abstract (or conceptual/template) workflow initiatives such as the conceptual workflows introduced in [Cerezo and Montagnat, 2011] could help in the design of inference rules. Indeed, fine-grained workflow structures could be hidden by higher level conceptual workflow elements and inference rules could be attached to these abstract workflow components instead of being attached to fine-grained provenance statements, thus enhancing their genericity. This genericity issue is more precisely discussed in the experiment presented in section 8.4.

Initially applied to computational neurosciences, this work goes beyond this scope, as the same principles are planned to be applied in the context of the VIP project. It is envisaged to validate the applicability and usability of the delineation of *Role* and *Natural* concepts in domain ontologies to (i) ease the design of simulation workflows (*e.g.* simulated cardiac images through ultrasound modality) and (ii) extend semantic repositories with new meaningful statements describing either simulated data or the simulated organs and their constituting anatomical entities. The experiment of section 8.4 evaluates our approach through concrete medical image simulation workflows deployed in the VIP platform.

Key Points

- *A clear delineation between Role and Natural concepts in domain ontologies allows for disambiguating semantic annotation of service parameters.*
- *Domain ontologies, instrumented with inference rules, allows for producing new meaningful statements based on workflow execution provenance.*
- *The reusability of inference rules is enhanced by involving domain-specific service annotations, rather than labels in provenance traces.*

Part III

Implementation and Evaluation

CHAPTER 7

Implementation

Contents

7.1 Introduction

This chapter briefly illustrates the software artifacts resulting from the thesis research contributions. It shows how they are related to the concrete research questions introduced in section 1.2.

E-Science environments aim in general in easing, through computing infrastructures, the setup of "*in silico*" experiments. These environments may address some, or part of, data acquisition, data curation, data selection or data processing (through scientific workflows) requirements. Collaborating activities rely on the coherent sharing of data or processing tools through such environments. They generally benefit from domain ontologies, capturing the knowledge of a particular area. The software developed in this work aims at easing (i) data selection, typically in the context of distributed e-Science platforms, (ii) data processing at both workflow design-time and run-time, and (iii) results publication through semantic guidance into massive data productions.

NeuSemStore The first software contribution, NeuSemStore, supports semantic scientific workflows, with a particular focus on life-science applications. More precisely, NeuSemStore proposes a semantic catalog of composable services, an extension of the Moteur workflow environment capturing on-the-fly provenance information, and means

to infer new meaningful statements from technical provenance information, usually difficult to comprehend. NeuSemStore leverages domain ontologies in the *in silico* experiments design process, and more technically, through *Role* annotations, how semantic service annotations can be disambiguated (RQ_4). NeuSemStore improves the understanding of workflow results (RQ_5), through new meaningful statements inferred from provenance information.

KGRAM-DQP The second software contribution, KGRAM-DQP, is an extension of the KGRAM Semantic Web Framework, supporting federated querying and reasoning over multiple distributed and heterogeneous data sources. More precisely, KGRAM-DQP allows collecting knowledge fragmented over multiple data sources (RQ_1), while still coping with the autonomy property of clinical/medical data providers (data is sensitive and may not be relocated externally). In addition, KGRAM-DQP implements the optimization strategies developed in chapter 4, in order to reduce the evaluation cost of distributed query processing (RQ_3).

The remainder of this chapter describes the NeuSemStore and the KGRAM-DQP software. It highlights their main features, drafts their architecture (through UML models), shows their current limitations and finally discuss their integration to enrich existing research software platforms (Moteur and KGRAM).

7.2 Supporting semantic scientific workflows: NeuSemStore

NeuSemStore is a semantic data store supporting neuroimaging workflows. It is aimed at persisting, retrieving and reasoning on semantic annotations. Semantic data is partitioned between descriptive annotation (service annotation) and invocation annotation (provenance information collected at workflow run-time).

This framework, currently composed of more than 16000 source lines of Java code, was initiated during the NeuroLOG project (ANR-06-TLOG-024) and it was extended during the VIP project (ANR-09-COSI-03). In the context of NeuroLOG the objectives of the framework were twofolds: (i) enhancing the design of neuroimaging workflows by providing means to validate the composition of services, and (ii) generating metadata describing the processed data in order to reinsert workflow results into federated databases. In the context of the VIP platform, NeuSemStore aimed at (i) storing and retrieving semantic annotations describing the simulation models, and (ii) automating the semantic annotation of simulated data resulting from simulation workflows.

To ease its integration within two distinct platforms (NeuroLOG and VIP), NeuSemStore has been designed with an important focus on code reusability and modularity.

7.2.1 Features summary

Annotating semantic services. Through a dedicated graphical user interface, illustrated in Figure 7.1, service providers are able to describe the functionality of the service by "drag-and-drop"-ing ontology concepts displayed in the right side of the window into the dedicated Functions panel located in the left side. A similar operation is necessary to annotate the nature and eventually the role of input and output parameters. Semantic descriptions can then be saved as a collection of RDF statements, or directly published into the service catalog.

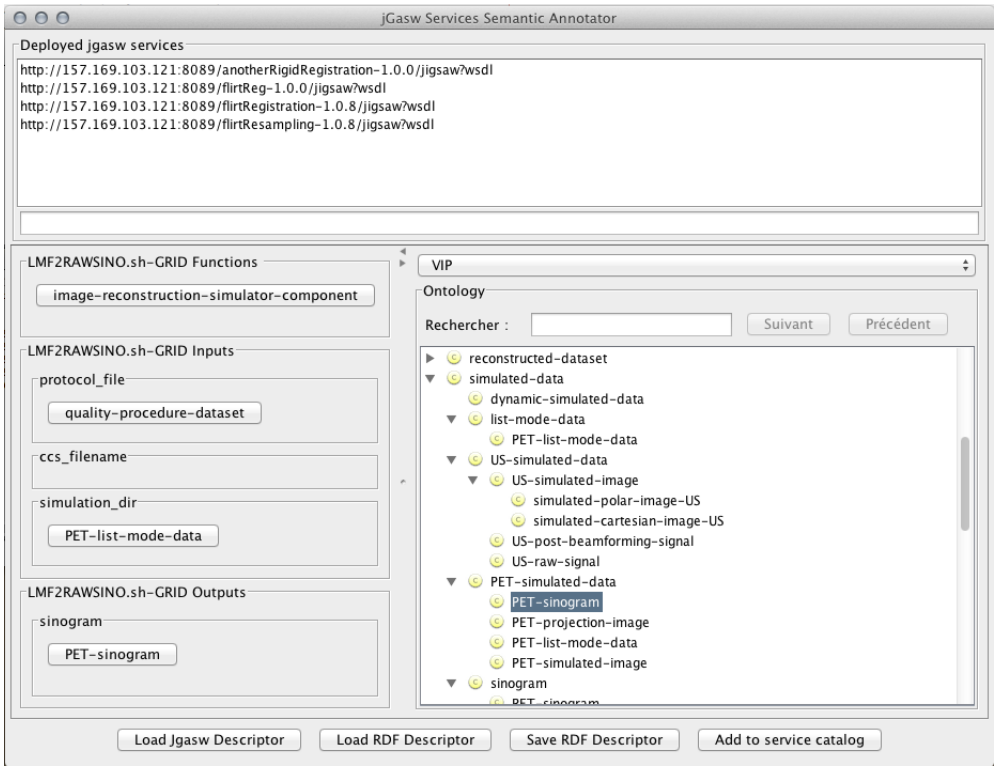

Figure 7.1: A graphical user interface aiming at semantically annotating a grid-instrumented jGasw service with VIP ontology concepts.

Cataloging semantic services. To enhance the overall coherency of workflows at design time, the semantic service catalog can be queried to retrieve services realizing a particular kind of treatment (through an associated concept of the domain ontology), or to retrieve services able to consume a particular kind of data at a given step of the workflow construction.

Recording workflow provenance. From the end-user perspective, recording workflow provenance consists in deploying the *Moteur-provenance-plugin* into the Moteur workflow environment. The plugin is then dynamically loaded when the workflow starts.

Provenance statements are stored on-the-fly, after each successful process invocation. In the current version of NeuSemStore, and to cope with the VIP platform requirements, provenance statements are following the OPM model. They are finally persisted into a JENA-TDB triple store.

Navigating workflow provenance. To help scientific workflow designers in testing their workflows, we propose a simple graphical user interface illustrated in Figure 7.2 which provides hierarchical views focusing on (i) process provenance (left side), and (ii) data provenance (right side).

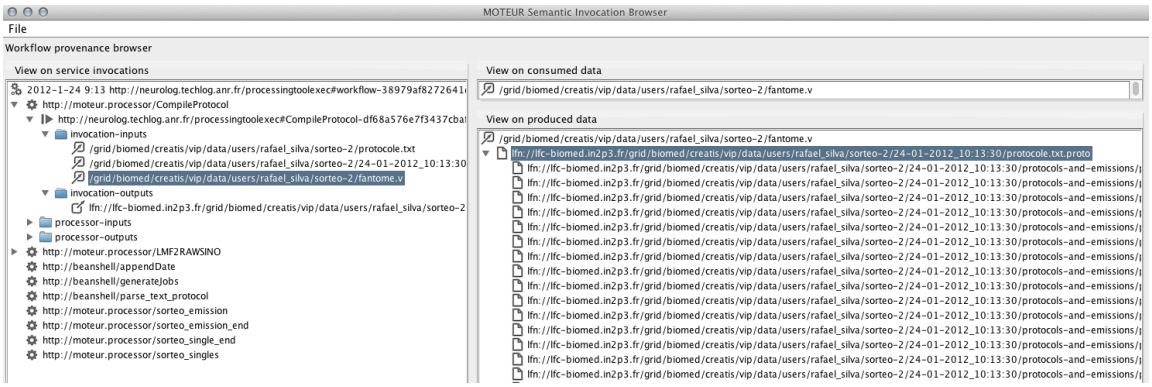

Figure 7.2: A graphical user interface aiming at navigating within process-oriented provenance and data-oriented-provenance.

End-users may first select the workflow invocation to be browsed. Then all services involved might be listed. When a service is selected, it is possible to list all its invocations, and all data that have been consumed as input or produced as output. When an input/output data is selected, it appears on the right side (data-oriented provenance). From this part of the window, end-users may iteratively develop causal dependencies to list either originating data, or derived data.

7.2.2 Architecture

The UML class diagram drafted in Figure 7.3 illustrates the main classes involved in the NeuSemStore framework.

To ease the extensibility of the software, interfaces have been proposed when possible. It allows to easily switch between several implementations when assembling NeuSemStore software components.

Knowledge graphs persistency. Triple persistency is handle through the *RDFManager* interface which provides a template for existing impletations, namely *RDFManagerSDBImpl* and *RDFManagerTDBImpl*. While *RDFManagerSDBImpl* relies on JENA-SDB and an existing SQL backend to perform reliable persistency, *RDFManagerTDBImpl* relies on

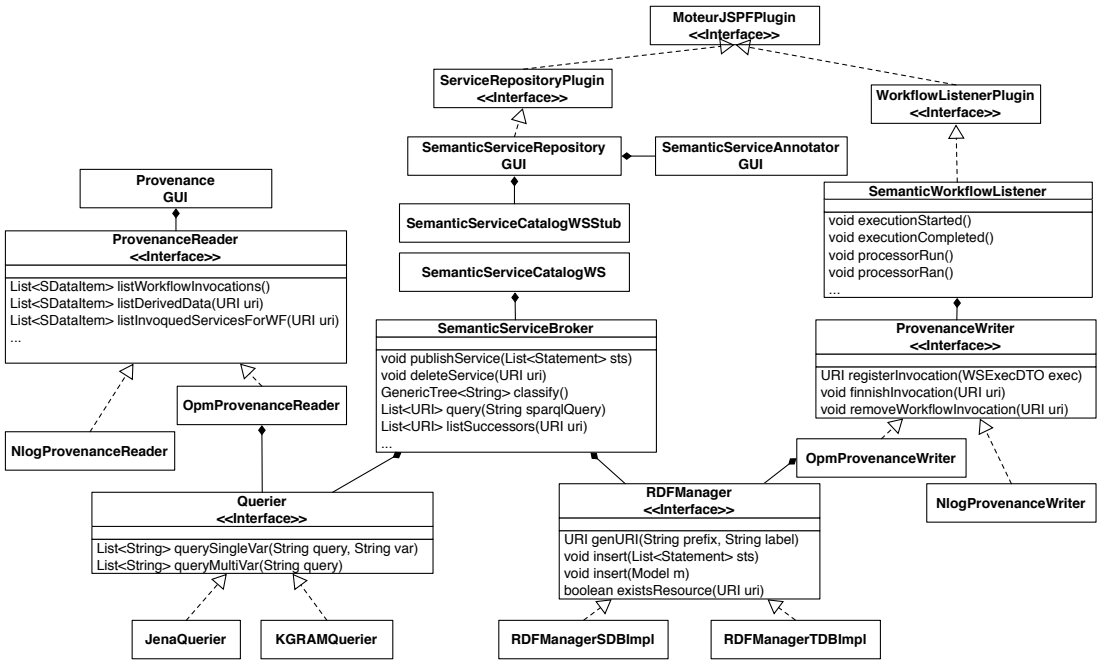

Figure 7.3: A UML static class diagram illustrating the main Java classes composing the NeuSemStore framework.

JENA-TDB to perform high-performance triples reading and writing through an optimized binary file system. *RDFManagers* are responsible for standard CRUD[1] operations on semantic annotations. They are also used to guarantee a coherent identification policy through the generation of URIs from a generated UUID[2], a given prefix, and a given label (the *genURI()* method).

MOTEUR Plugins. Semantic annotations are generated from the semantized Moteur workflow environment, namely Moteur-S. Moteur has been slightly extended with the JSPF plugin framework. Two kind of plugins were envisaged : *ServiceRepositoryPlugins* and *WorkflowListenerPlugins*.

ServiceRepositoryPlugins are dedicated to plug several kinds of Moteur processor catalogs. We could thus envisage a specific catalog for beanshell[3] processors, or another one for grid-instrumented services (jGasw or Gasw services) or yet another one for external web services.

WorkflowListenerPlugins are intercepting workflow and processor invocations to enable pre- or post-processing. In the context of NeuSemStore, the *SemanticWorkflowListener* plugin is responsible, after each successful processor invocation, for recording all input and output data to generate on-the-fly provenance information.

[1]Create, Read, Update, Delete operations.
[2]a Universal Unique Identifier.
[3]a lightweight Java scripting language.

Workflow provenance. The *SemanticWorkflowListener* plugin is thus linked with a *ProvenanceWriter* implementation responsible for the persistence of provenance information through a dedicated provenance model (OPM or OntoNeuroLOG ontologies). Provenance information may be navigated through a dedicated graphical user interface (the *ProvenanceGUI* class) which is linked to a *ProvenanceReader* implementation. The *ProvenanceReader* allows for navigating into workflow provenance graphs through a set of *list*()* methods. Since provenance information is persisted as knowledge graphs, it is queried through a *Querier* class topping either the ARQ Jena engine (*JenaQuerier* class), or the KGRAM engine (*KGRAMQuerier* class).

Semantic Service Catalog. To exploit workflow provenance with domain-specific knowledge, semantic service description must be accessible. The *SemanticServiceCatalogWS* exposes service descriptions – which result from manual annotation (*SemanticServiceAnnotatorGUI* class) – through a web service. The main cataloging features are located into the *SemanticServiceBroker* which enables (i) managing service descriptions (*publishService()* and *deleteService()* methods), (ii) searching services through SPARQL queries or eventually searching for pluggable services (*listSuccessors()* method), and (iii) classifying service descriptions by sorting them into a functionality taxonomy. Note that services are considered as pluggable if they can be connected through semantically compatible data links. The *listSuccessors* implements both *exact match*[4] and *plugin match*[5] strategies as introduced in [Mehandjiev et al., 2010].

The NeuSemStore software is organized in modules as follows:

- *NeuSemStore-core*: a module dedicated to RDF triples persistency. It is based on either JENA-SDB, to benefit from a scalable SQL backend, or on JENA-TDB to propose an optimized, performance-oriented backend. In addition, it relies heavily on the Corese/KGRAM semantic engine to perform in-memory semantic querying and reasoning (forward chaining inference rules and RDFS entailments) ;
- *NeuSemStore-broker*: a module providing a catalog of semantically annotated services;
- *NeuSemStore-provenance*: a module handling the creation of provenance annotations, and their querying through predefined SPARQL queries. Two ontologies are currently supported, the OntoNeuroLOG ontology and Open Provenance Model (OPM) ;
- *NeuSemStore-gui*: a set of graphical user interfaces to annotate services and interact with the service catalog. Another graphical user interface is dedicated to the navigation through provenance annotations ;

[4] when the output O_i of service S_1 is described with exactly the same concept than the input I_j of service S_2 consuming O_i.

[5] when the output O_i of service S_1 is described with a sub-concept of the concept describing the input I_j of service S_2. The *plugin match* is also known as narrower match.

- *NeuSemStore-catalog-moteur-plugin*: a module providing a semantic repository of composable services for the MOTEUR workflow engine ;

- *NeuSemStore-provenance-moteur-plugin*: a module enabling the recording of provenance informations during workflow executions when plugged to the MOTEUR workflow engine ;

- *NeuSemStore-simulation-objects*: a module dedicated to the management of domain specific annotations, such as *simulation objects* in the context of the VIP project.

The NeuSemStore framework has been built upon open source libraries such as Apache JENA for RDF data persistency, KGRAM for abstract graphs querying and reasoning through Semantic Web standards and novel extensions, JAX-WS for the standard Java Web Service stack, Apache Commons and Log4J for helper classes, or JSPF for a simple java plugin framework.

7.2.3 Discussion

Through NeuSemStore, we propose a set of integrated software components supporting Semantic Scientific Workflows. However, exploiting semantic scientific workflows to automate the production of new meaningful statements still remains a complex process : (i) services composed through workflows must be beforehand semantically annotated, (ii) workflow engines must be instrumented to produce and store fine-grained domain-agnostic provenance, (iii) inference rules must be designed, based on provenance, and domain ontologies, to infer new domain-specific statements, and finally (iv) inference rules must be enacted and new resulting inferred statements must be retrieved and persisted back.

The scalability of our approach has been experimented, in the context of the VIP platform, through the setup of two distinct triple stores, managed through NeuSemStore. The first one is dedicated to short-term, fine-grained OPM provenance. The second one is a long-term repository aimed at storing the concise new inferred domain-specific statements constituting the meaningful experiment summaries, and representing a real value-added for end-users.

The NeuSemStore architecture is versatile enough to enable the management of provenance and service annotations through several ontologies. With a limited development effort, we provided two implementations for managing provenance. While the *OpmProvenanceWriter* is responsible for storing provenance trough the OPM standard, the *NlogProvenanceWriter* has been experimented to produce provenance statements through the OntoNeuroLOG ontology. We could similarly envisage to evolve from the OPM provenance model to the the Prov-DM model that is under standardization by the W3C. In a same way, it could be envisaged, with a limited development effort, to produce semantic service descriptions through the OWL-S ontology. This versatility results from the NeuSemStore software architecture design. It was consolidated by its integration

into two e-Science platforms – NeuroLOG dedicated to multi-centric neuroscience studies, and VIP dedicated to multi-modal and multi-organ medical image simulation – that enforced reusability throughout the software development process.

7.3 Semantic federation engine: KGRAM -DQP

KGRAM, also known as Corese/KGRAM and introduced in section 4.2.1, is a Semantic Web Framework, , implementing the W3C standards: RDF , RDFS , SPARQL 1.1 Query & Update and SPARQL Rules for RDF . KGRAM is distributed as an open source software with a CeCCILL-C[6] Licence.

7.3.1 Features summary

The distributed query processing strategies developed in chapter 4 have been implemented as two extensions for the KGRAM Semantic Web Framework. They provide means (i) for data providers, to expose semantic data through a remote semantic query interface, and (ii) for data queriers, to federate multiple distributed data sources through distributed query processing.

KGRAM Endpoint A server capability is the first feature needed to achieve a federation over multiple KGRAM *Producers*. In this first prototype, we developed a standard web service aimed at exposing a KGRAM *Producer* to the Internet. These *Producer* servers are hosted into a standard Apache Tomcat container.

KGRAM Federator To allow KGRAM for querying and reasoning, not only on local graphs, but also on graphs spread over the Internet, we slightly extended the KGRAM framework. We first implemented a new *Producer* aimed at enumerating remote graph edges and nodes. This producer integrates the static and dynamic optimizations developed in chapter 4. In addition, we implemented a new *MetaProducer* aimed at exploiting the service parallelism offered by distributed endpoints, also benefitting from the optimizations strategies developed in chapter 4.

Even if it has only been experimented in the context of distributed KGRAM endpoints, the proposed federation implementation is not limited to KGRAM-interfaced stores. It can exploit any standard SPARQL endpoint. Indeed, since all subqueries transferred over the network are expressed through standard SPARQL queries, we can federate not only KGRAM endpoints but any standard SPARQL endpoints.

7.3.2 Architecture

The UML class diagram drafted in the following Figure 7.4 illustrates the main classes involved in the KGRAM framework extensions, dedicated to distributed query processing.

[6] http://www.cecill.info/licences/Licence_CeCILL-C_V1-en.html

While black UML classes represent existing Java classes composing the initial KGRAM framework, green UML classes represent new Java classes extending KGRAM to address distributed and heterogeneous multi-source federated querying.

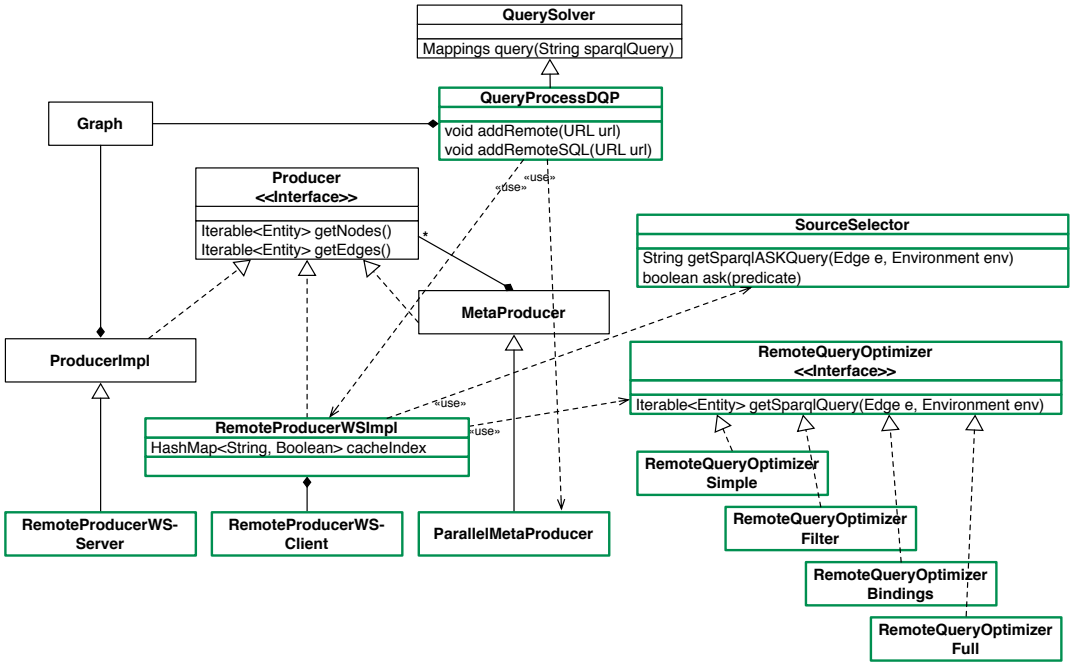

Figure 7.4: An UML static class diagram illustrating the main Java classes composing the KGRAM distributed query processing extension. Green classes represent the KGRAM extensions dedicated to the distributed query processing.

Data provider perspective. From the data provider perspective, the publication of data consists in (i) loading the data to be published into a *Graph* object and (ii) exposing its content through a remote query interface. We developed a KGRAM web service (*RemoteProducerWS-Server*) based on the JAX-WS standard web service stack. The proposed KGRAM endpoint can (i) be populated with RDF content, (ii) be remotely queried through SPARQL 1.1 queries embedded within SOAP messages. It returns RDF results also embedded within SOAP messages. The proposed KGRAM endpoint extends the *ProducerImpl* class and allows navigating into the hosted *Graph*.

Data querier perspective. From the data querier perspective, querying activities consist in (i) configuring KGRAM with a set of URLs identifying and localizing the distributed data sources, and (ii) submitting to KGRAM a standard SPARQL 1.1 query, without specifying any distribution directive (such as the *service* SPARQL extension proposed by the W3C).

The configuration of the KGRAM engine is achieved through the *QueryProcessDQP* class. This class extends the *QuerySolver* class responsible for the evaluation of SPARQL

1.1 queries on abstract graph structures. Target distributed data sources are collected through the *addRemote()* method and the distributed evaluation is launched through the *query()* method inherited from the *QuerySolver* class.

Throughout its configuration, the KGRAM engine instantiates a specialized *Producer* which will not handle a local knowledge graph but act as a client for a remote knowledge graph exposed through the *RemoteProducerWS-Server*. The *RemoteProducerWSImpl* indeed implements the *Producer* interface and embeds the web service client stubs (*RemoteProducerWS-Client*).

All data sources are handled in KGRAM through *MetaProducers* which iterate over a collection of *Producers*. We implemented a *ParallelMetaProducer* aimed at exploiting service parallelism, and based on the algorithms proposed in chapter 4. Object-oriented techniques such as polymorphism were used to easily integrate several versions of the *ParallelMetaProducer* (using the "parallel-pipeline" strategy introduced in section 4.2.4.2 for instance).

Optimizations. Query rewriting optimizations are located into the classes implementing the *RemoteQueryOptimizer* Java interface. During experiments, query optimizers can easily be switched directly from the *RemoteProducerWSImpl* class. The filter rewriting strategy introduced in section 4.2.3.1 is implemented in the *RemoteQueryOptimizerFilter* class, and the binding rewriting strategy introduced in section 4.2.3.2 is implemented in the *RemoteQueryOptimizerBinding* class. Finally, the *RemoteQueryOptimizerFull* class combines these two optimizations.

The *ASK-based* source selection algorithm introduced in section 4.2.3.3 is implemented in the *SourceSelector* class. This class is responsible for the generation of SPARQL ASK queries based on edge requests, and their evaluation on a given remote producer. The ASK results finally populate a cache associated to remote producers (the *HashMap<String,boolean> cacheIndex* attribute). This memory cache allows remote producers checking if a predicate is hosted on the associated endpoint, and thus prevents for unnecessary remote invocation.

7.3.3 Integration of KGRAM-DQP within the existing framework

The KGRAM framework is organized through the following Maven modules:

- *kgram*: the core abstract graph entities, and an associated abstract query language (an abstract SPARQL interpreter) ;
- *sparql*: a concrete SPARQL parser ;
- *kgenv*: a concrete SPARQL compiler ;
- *kgtool*: an utility classes aimed at, among other things, loading or serializing knowledge graphs through RDF or XML Results sandards ;
- *engine*: an inference rule engine ;

- *kgengine*: the Corese API, maintained in KGRAM to guarantee backward-compatibility ;
- *kggui*: the graphical user interface in which users may load knowledge graphs or inference rules, edit/execute SPARQL queries.

Based on these pre-existing modules, we propose three new modules dedicated to federated querying:

- *kg-server*: a standard web service, encapsulating a KGRAM instance, allowing for knowledge graph publication through a SPARQL remote querying interface ;
- *kg-server-stubs*: the automatically generated web service client classes, used for communicating with the KGRAM server exposed as a web service ;
- *kg-dqp*: the KGRAM extensions providing optimized federated querying and reasoning over multi source knowledge graphs.

7.3.4 Discussion

This first version of KGRAM-DQP exposes knowledge graphs through a web service endpoint. An HTTP version is under development and few modifications have been required to handle HTTP-based communications, instead of SOAP-based communications. However, since the federated querying is handled through standard SPARQL queries, any SPARQL endpoint may be federated through KGRAM-DQP.

The current limitations of this first version are :

- named graphs are not yet supported in federated querying ;
- the cost of SPARQL negations should be studied in a distributed setup.

KGRAM can still benefit from more optimization opportunities to enhance the end-user experience when querying massive distributed datasets, specifically when addressing edge grouping through the dynamic generation of SPARQL *Service* clauses (see section 4.2.3.4). However we have seen that edge grouping might, in some cases, lead to less effective invocations. These optimizations require more investigations in the field of query planning (intra and inter group order).

Finally, KGRAM-DQP could also be used to study how new SPARQL 1.1 constructs such as UPDATE could be handled in a distributed context.

7.4 Conclusion

We propose in this chapter a technical description of the software implementing the main thesis contributions.

NeuSemStore provides support towards Semantic Scientific Workflows and finally allows for producing valuable knowledge through the usage of e-Science platforms. More

precisely, its *Annotator* module directly addresses the semantic annotation of services involved in scientific workflows (RQ_4) by proposing a graphical user interface allowing service providers to annotate the service functionality and its parameters through Natural and Role concepts. Its *Broker* and *Service-catalog-plugin* modules allows selecting and composing semantically, appropriate services into the Moteur workflow workbench at design-time. Finally, its *Provenance* and *Provenance-plugin* modules are responsible for provenance management at workflow run-time, and for ultimately inferring new meaningful statements which constitute the proposed meaningful experiment summaries (RQ_5).

KGRAM-DQP provides support for sharing knowledge distributed over multiple, possibly heterogeneous, data sources. More precisely, its *ParallelMetaProducer* and its *RemoteProducers* enable federated semantic querying over multiple distributed data sources (RQ_1). KGRAM-DQP integrates a set of static and dynamic optimizations (RQ_3) reducing the cost of federated semantic querying.

Regarding the dissemination of these software components, while KGRAM-DQP is still under integration into the Corese/KGRAM Semantic Web factory, the NeuSemStore software has been used in production into the NeuroLOG and the VIP platforms, involving significant user communities. The provenance capabilities of NeuSemStore have also been experimented at the Amsterdam Academic Medical Center, to compare provenance traces captured through *post-mortem* analysis of Moteur logs [Benabdelkader et al., 2011].

Key Points

- *NeuSemStore extends, through two plugins, the Moteur workflow engine with (i) a semantic catalog of composable services, and (ii) the capture of OPM provenance at workflow run-time.*
- *NeuSemStore is versatile and allows to adapt to ontologies evolutions (provenance and service ontologies).*
- *KGRAM-DQP provides transparent semantic federated querying. KGRAM-DQP is data-source content-agnostic, and thus does not require explicit distribution directives into* SPARQL *queries.*
- *KGRAM-DQP is versatile and allows, through abstract knowledge graphs, to adapt to distributed and heterogeneous knowledge bases.*

CHAPTER 8

Experimental evaluation

Contents

8.1 Introduction

In this chapter, we propose a set of experiments exploring several scientific challenges related to the Distributed Systems, Knowledge Engineering, or e-Science areas.

The distributed provenance experiment reported in section 8.4 aims at enhancing the exploitation of community knowledge capitalized in a domain ontology, through the execution of e-Science experiments (scientific workflows). The linked data experiment proposed in section 8.3 aims at showing that federated approaches for semantic data querying promote knowledge sharing and semantic interoperability. The heterogeneous federation experiment reported in section 8.3 shows that the proposed KGRAM federated querying engine transparently adapts to both graph-based and relational databases, and addresses structural interoperability issues observed in the NeuroLOG neuro-imaging platform. Finally, large scale experiments reported in section 8.2 evaluate the performance and the scalability of the KGRAM federated querying engine through DBpedia Open Linked Data querying and the FedBench benchmark dedicated to the evaluation of federated approaches. These experiments are supported by the experimental Grid'5000 infrastructure.

Table 8.1 illustrates the proposed experiments and how they relate to performance, scalability, interoperability, usability, or re-usability.

Experiment	*Performance*	*Scalability*	*Interoperability*	*Usability*	*Re-Usability*
Large scale experiments (sec. 8.2)	++	++	−−	−−	−−
Heterogeneous federation (sec. 8.3)	+	−−	++	+	−−
Linked neuro-data (sec. 8.3)	−−	−−	+	−−	++
Distributed provenance (sec. 8.4)	−−	+	−−	++	+

Table 8.1: Four experiments exploring several challenges related to Distributed systems, Knowledge engineering, or e-Science areas.

8.2 Large scale experiments

For practical reasons, and due to memory/processor consumption, it is not always feasible to centralize massive datasets and perform queries against a unique repository. Federated approaches, defended all along this work, aim at coping with multiple distributed data sources by pushing remote subqueries and merging the partial results to finally build a virtual data integration. However, although coping with multiple distributed data sources presents valuable interests such as no central point of failure or no need to periodically populate a centralized repository, transparent federated querying requires costly distributed joins both in terms of volume of network communication, and in terms of overall execution time. The following two experiments aim at (i) demonstrating the interest of federated semantic querying, in particular when addressing massive datasets that could not be handle through a classical centralized triple store, (ii) studying the efficiency of the distributed optimized querying developed in chapter 4, and (iii) benchmarking our approach against state-of-the art federated engines.

The experiment reported in section 8.2.1 shows that in the case of an expensive query (leading to numerous distributed joins) fragmenting the initial dataset over multiple distributed data sources can lead to an important gain (a faster evaluation, up to 7 times), but starting from 8 distributed fragments, the network overhead starts to penalize the distributed evaluation.

The benchmarking experiment reported in section 8.2.2 compares the state-of-the-art transparent federating approaches through a set of predefined Life Science SPARQL queries (see section 10.1) and a set of "Linked Data" datasets covering mainly genetical and chemical data but also general DBpedia data. The experiment, deployed on real distributed infrastructure, shows longer evaluation time compared to the FedX engine, but still performs better than its other competitors. It also provide a higher expressivity (SPARQL 1.1 path expressions and RDFS entailment regime) compared to all other approaches.

To summarize, the objectives of these two large scale experiments are the following:

- *highlighting the interest of federated querying when addressing massive semantic data querying ;*
- *studying the scalability of the proposed algorithm ;*
- *presenting reasonable performances compared to state-of-the-art approaches through the FedBench benchmark.*

8.2.1 Querying distributed DBpedia datasets

8.2.1.1 Material and methods

DBpedia person dataset. DBpedia is one of the major initiatives to publish Linked Open Data. It consists in extracting data from Wikipedia, and publishing it through the Semantic Web standards (RDFS ontologies, and RDF triples). The *Persondata* dataset represents persons extracted from Wikipedia and their characteristics such as their birth date, birth place, and given name. The *Persondata* dataset is structured through the FOAF [1] "Friend-Of-A-Friend" ontology. The *Persondata* DBpedia subset is a good candidate to evaluate the performance and the scalability of our approach since it is composed of 1.7 million of RDF statements following a simple FOAF schema. In this experiment, we compare the querying of the whole dataset through several levels of fragmentation.

Data fragmentation. The 1.7M statements of the *Persondata* DBpedia subset have been fragmented in up to 16 fragments of the same size. Precisely, *Persondata* have been fragmented in 2 fragments of 850K statements each, 4 fragments of 425K statements each, and so on, up to 16 fragments of 106K statements each. The fragmentation was manually and randomly realized, without any consideration regarding the kind of FOAF statement. This distribution setup corresponds to the initial hypothesis made on collaborative e-science platforms, such as the NeuroLOG platform, where each collaborating partner may share the same kind of data.

Selective and expensive queries. The following queries $Q_{8.1}$ and $Q_{8.2}$ have already been introduced in chapter 4. $Q_{8.1}$ leads to only 8 results while $Q_{8.2}$ leads to 1184 results.

Distributed query engine. The distributed query engine used in this experiment is the KGRAM federated engine presented in section 7.3 and implementing the federated querying strategies developed in chapter 4.

When performing a SPARQL query with KGRAM, the engine first populates a cache memory used with indexes needed to accelerate queries involving similar edge requests. For a same query, this internal optimization results in a longer evaluation in the first run and much faster evaluations for the following runs. This experiment focuses on the first

[1] http://xmlns.com/foaf/spec

Listing 8.1: Selective SPARQL query distributed over the full DBpedia-person dataset

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dbpedia: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?x ?name ?date WHERE {
        ?x foaf:name ?name .
        ?x dbpedia:birthDate ?date .
        FILTER (CONTAINS (?name, 'Bobby␣A'))
}
```

Listing 8.2: Expensive SPARQL query with an *optional* statement

```
PREFIX foaf: <http://xmlns.com/foaf/0.1/>
PREFIX dbpedia: <http://dbpedia.org/ontology/>
SELECT DISTINCT ?x ?name ?date ?place WHERE {
        ?x foaf:name ?name .
        ?x dbpedia:birthDate ?date .
        OPTIONAL {?x dbpedia:birthPlace ?place}
        FILTER (CONTAINS (?name, 'Bobby␣A'))
}
```

evaluation only since this experiment aims at showing the benefits of distributing the evaluation load on massive data (objective O_1). Each distributed KGRAM engine (*i.e.* acting as a SPARQL endpoint) is consequently reinitialized before each distributed query evaluation.

Grid'5000 infrastructure. This experiment benefits from the Grid'5000 infrastructure dedicated to the study of large-scale parallel and distributed systems. The grid resources reservation is similar to the grid experiment setup introduced in section 4.3.

8.2.1.2 Results and discussion

Performance. The following experiment aims at measuring the efficiency of the proposed distributed querying algorithms through the deployment of 1 to 16 distributed data stores hosting the full DBpedia-person dataset. We will see that depending on the cost of the distributed joins involved in a distributed query, and on the level of data fragmentation (number of federated endpoints) the overall performance of the distributed query processor can be clearly impacted.

Figure 8.1 shows the benefits of fragmenting linked data through distributed federated endpoints. The experiment has been conducted through 17 nodes of the "Suno" cluster of Grid'5000 (1 node hosting the federator and the 16 remaining nodes hosting the federated endpoints). We observe that for the selective query (Q8.1), the DQP time keeps decreasing when the fragmentation of the knowledge base increases, whereas for the less selective query (Q8.2), the DQP time starts to increase starting from 8 federated

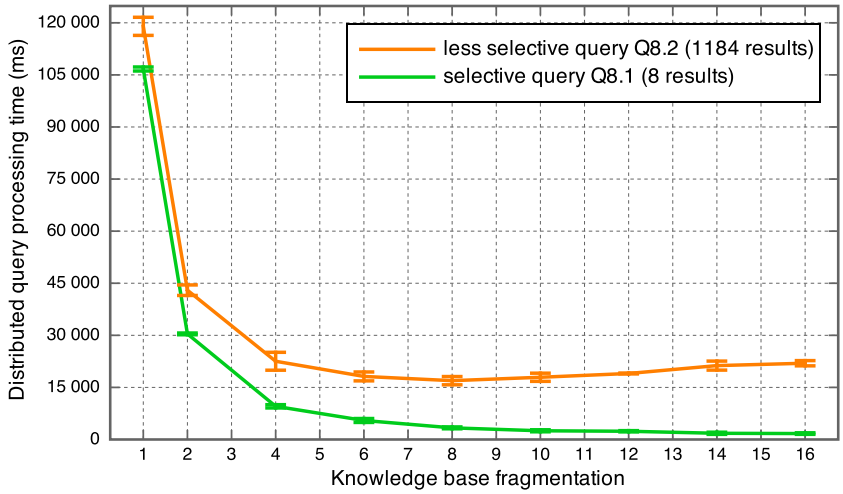

Figure 8.1: Decreasing distributed query processing (DQP) time for a large scale knowledge base (1.7M triples) fragmented through up to 16 data stores, under *full* query rewriting strategy.

endpoints. As expected, the number of transferred triples is constant for all distribution setups (20 for Q8.1 and 3242 for Q8.2). However, since the fragmentation factor increases, the number of remote invocations also increases proportionally. We measured from 13 remote invocation (single federated endpoint) to 208 (16 federated endpoints) for Q8.1 and from 2411 to 38576 remote invocations for Q8.2 in the same distribution scenario. This high number of remote invocations illustrates the network overhead which explains that the DQP time stops decreasing for more than 8 federated endpoints.

Speedup and efficiency. In addition we calculate the efficiency (E_n) of the distributed query processing to characterize the behavior of the system when the number of federated endpoints increases. The efficiency is calculated as follows,

$$E_n = \frac{T_1}{n * T_n}$$

where T_1 represents the query time using a single federated endpoint, n represents the number of federated endpoints, and T_n the global query time for a knowledge base fragmented over n federated endpoints. The efficiency is represented by a number usually between 0 and 1, where 1 denotes an ideal use of the resources available (T_n is exactly n times smaller than T_1).

Figure 8.2 illustrates the measured efficiencies when querying Q8.1 and Q8.2 on all distribution setups (from 1 to 16 federated endpoints). Distributed query processing is performed under the *full* query rewriting optimization and the *parallel-pipeline* strategy. It has to be noted that the T_1 measured does not correspond to a fully centralized setup, that would avoid any network communication, but rather corresponds to a single distributed endpoint scenario. These good results ($E_n > 1$) are due to a particularly naive querying

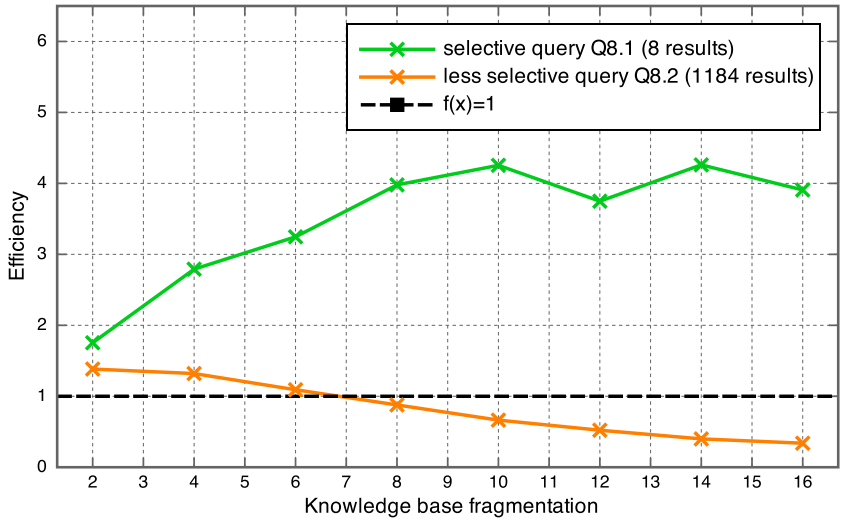

Figure 8.2: Compared to a single distributed endpoint, both distributed queries are "supra-effective". In the case of the less selective query Q8.2, the distributed querying becomes "ineffective" starting from 8 federated endpoints.

in the context of a single distributed endpoint (T_1). Indeed, the algorithm pushes remote queries for each edge request composing the full SPARQL query. In the context of a single federated endpoint, it is not necessary to do so. Sending the full SPARQL query to the unique federated endpoint would be much more effective, but as soon as we consider more than one federated endpoint, this strategy cannot be envisaged.

We finally observe that for the expensive query Q8.2, it becomes "ineffective" to fragment the DBpedia-person dataset over more than 8 federated endpoints. This criterion corresponds to the increasing DQP time observed in Figure 8.1. This can be explained by the network overhead paid for performing up to 38576 remote invocations.

Although the efficiency seems to be poor starting from 8 federated endpoints, Figure 8.3 shows that the speedup of the algorithm remains better than a logarithmic speedup generally observed for "hard-to-parallelize" algorithms. This graph is obtained from the same experiment, and is calculated as follows,

$$S_n = \frac{T_1}{T_n}$$

where T_1 represents the query time using a single distributed endpoint, n represents the number of federated endpoints, and T_n the global query time for a knowledge base fragmented over n federated endpoints.

Figure 8.3 also compares the speedup that would be obtained if T_1 was calculated with a single remote endpoint queried with the full SPARQL query (105s compared to 118s through the federation algorithm), thus avoiding to perform all useless unitary remote invocations. Although the speedup is a bit lower, it still has the same characteristics : better than a logarithmic speedup, and supra-linear up to almost 6 distributed endpoints. It

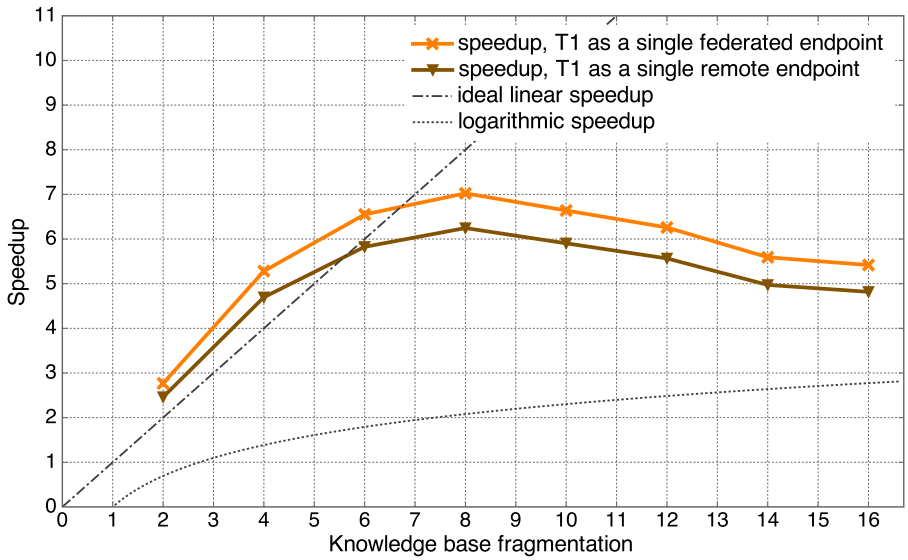

Figure 8.3: Starting from 8 federated endpoints, the distribution algorithm becomes less efficient due to communication overhead.

has to be noted that T_1 obtained in a standalone setup (avoiding any network communication) is far better but is not representative to compare the distributed evaluations over all fragmentation scenarios.

To summarize, this large-scale "DBpedia" experiment brings the following results :

- *Performance: for costly query, the distributed query processing time reduced by a factor 7 when the full DBpedia-person dataset is distributed over 8 data sources ;*
- *Efficiency: the proposed distributed query processing strategies are supra-effective for less than 7 distributed data sources in the context of the studied costly* SPARQL *query.*

8.2.2 The "FedBench" federation benchmark

8.2.2.1 Material and methods

The FedBench benchmark. FedBench [Schmidt et al., 2011] is the first benchmark suite dedicated to the analysis and the evaluation of different federated query processing approaches. Other benchmarks such as LUBM [Guo et al., 2005], SP^{2}Bench [Schmidt et al., 2008], or the Berlin SPARQL benchmark [Bizer and Schultz, 2009] only address the comparison of SPARQL engines under a centralized deployment. FedBench explores several dimensions of federated approaches and focus on both data level and query level. Depending on the federation setup, the query engine may process either local data or remote data physically distributed over the Web, or a combination of both local and remote data. Moreover data access interfaces (1) may vary from native repositories to standard SPARQL endpoints or simple HTTP requests and (2) possibly expose data statistics useful for data source selection and more generally to perform efficient query processing. With respect to the query dimensions, FedBench also addresses the variability of query languages (simple conjunctive queries through BGP processing, or more complex SPARQL 1.1 constructs), and the expected completeness of results. Indeed, depending on the usage scenario, only the first k results might be of interest due to responsiveness constraints, or on the contrary end-users are expecting a complete result set, even if the computation is heavy.

The FedBench datasets. FedBench proposes a set of queries dedicated to federated querying over 3 datasets. The first one addresses a general collection of Linked Open Data (DBpedia, GeoNames, Jamendo, LinkedMDB, New York Times and Semantic Web Dog Food). The second one, SP^{2}Bench, consists in generating synthetic bibliographic data and fragmenting it following the various types of data to finally assemble a distributed setup. The third one is more interesting in the context of this work since it focuses on Life Science Linked Data and the corresponding queries. The Life Science Data Collection includes the KEGG (Kyoto Encyclopedia of Genes and Genomes) Drug dataset, the ChEBI (Chemical Entities of Biological Interest) dictionary of molecular entities, the DrugBank bioinformatics and cheminformatics dataset describing drugs and drug targets through a pharmacological perspective, and finally a DBpedia subset. Datasets are interlinked through either the `keggCompoundId` property (for bridging DrugBank to KEGG) or the general `owl:sameAs` property. Moreover some bridges are established between different data collections by exploiting literal values such as entity names.

To evaluate KGRAM, we propose in the reminder of this section two experiments based on this large-scale FedBench benchmark. Experiment 1 (reported in section 8.2.2.2) positions KGRAM with respect to its competitors based on the experiments conducted in [Schwarte et al., 2011]. Experiment 2 (reported in section 8.2.2.3) refines this evalua-

tion by comparing the most efficient engine, FedX, to KGRAM, on the same controlled distributed computing infrastructure (Grid'5000).

8.2.2.2 Experiment 1: positioning KGRAM with state-of-the-art approaches through the FedBench benchmark.

Distribution setup

The following distribution setup illustrated in table 8.2, aims at evaluating the performance of the KGRAM distributed query processing engine through the evaluation of the FedBench Life Science queries. The Life Science Data collection totalizes more than 49 millions triples, which have been fragmented over 6 distributed data sources.

Data source	*Linked Data collection*	*Size (triples)*
#1	ChEBI	4.7M
#2	DBpedia sub-set #1	13.1M
#3	DBpedia sub-set #2	18.2M
#4	DBpedia sub-set #3	11.8M
#5	DrugBank	0.5M
#6	KEGG Drug	1M

Table 8.2: FedBench Life Science data collection (49M triples) fragmented over 6 data sources.

Seminal evaluation experiments with the FedBench benchmark [Schwarte et al., 2011] have been conducted through a *virtual federation*. More precisely, the physical distribution of data sources has been simulated trough a single server (HP Proliant DL360 G6 with 2GHz 4Core CPU, 32GB 1333MHz RAM, and a 160 GB SCSI hard drive), running several SPARQL endpoints, without any network communications.

In the current evaluation of the KGRAM federation engine, we consider both *virtual* and *physical federation*. Experiments were conducted through the Grid'5000 infrastructure by reserving nodes of the "Suno" cluster (2 Intel CPUs (2.26GHz) with 4 cores per CPU, 32GB of RAM, and 519GB of disk storage):

Virtual federation : A single computing node was allocated for the experiment. An apache Tomcat application container was configured to host 6 KGRAM web service endpoints, one for each data source previously considered. 20GB of RAM were allocated to the apache Tomcat Java virtual machine, and 1GB to the KGRAM Metaproducer (federation engine). Endpoints are collocated on a same host and no network communication is required.

Physical federation : 6 nodes with exactly the same characteristics were allocated for this experiment. On each node, an apache Tomcat was deployed to host a single KGRAM endpoint. We also allocated 20GB of RAM for each of the 6 apache Tomcat JVMs, and 1GB of RAM for the KGRAM Metaproducer, hosted by a dedicated computing node. This distribution setup has been chosen to reflect a real-life distributed scenario, involving network communications between the endpoints.

Results and discussion

Table 8.3 reports the benchmark results of the KGRAM engine when considering both the virtual and physical federations. Results were produced through 4 runs of the 7 Life Science queries. The first run of KGRAM performs a costly initialization of the endpoints which biases request processing measurement times. It is therefore discarded. We then consider the 3 following runs, which reflect a real-life scenario where KGRAM endpoints would be running in production. Table 8.3 finally reports the mean evaluation time over 3 consecutive runs, and the associated standard deviation. It shows that runs on the physical federation are usually longer than on the virtual federation (all queries but LS6), especially when the query processing time is longer (queries LS3, LS5 and LS7), and that the standard deviation is also increasing. These results reflect the impact of network communication on the query processing time.

	LS1	LS2	LS3	LS4	LS5	LS6	LS7
Virtual federation							
Eval. Time (s)	0.62	0.51	50.9	0.034	17.79	0.66	9.18
Std. Dev. (s)	0.00	0.00	0.90	0.00	0.13	0.24	0.24
Physical federation							
Eval. Time (s)	0.632	0.52	55	0.038	18.44	0.54	9.81
Std. Dev. (s)	0.01	0.00	1.94	0.00	0.83	0.01	0.56
Impact of the physical federation							
	+1.93 %	+1.96 %	+8.05 %	+11.76 %	+3.65 %	-18.18 %	+6.86 %

Table 8.3: FedBench results for the KGRAM federation engine considering both virtual and physical federations.

Figure 8.4 compares KGRAM results with the actual state-of-the-art approaches (reported in [Schwarte et al., 2011]). A cross indicates an evaluation error (DARQ with LS2 and AliBaba with LS6 and LS7), and the sad smiley indicates a timeout[2] (AliBaba with LS3 and DARQ with LS7). It has to be noted that for our *physical federation* deployment, response times have been measured for KGRAM on a real distributed infrastructure (Grid'5000), whereas response times measured in [Schwarte et al., 2011] for FedX/-DARQ/AliBaba were obtained only in the context of their *Virtual federation* (a single HP Proliant server as described previously).

The benchmark results first show that compared to AliBaba and DARQ, KGRAM, which implements the full SPARQL 1.1 language, benefits from a better expressivity and is able to answer all the Life Science queries. Moreover the overall response time measured are similar to the one obtained by its competitors. However, we observe that the FedX engine keeps a leg up over their competitors on all queries, and in particular when processing expensive queries such as LS3, LS5 or LS7.

[2] more than 600s

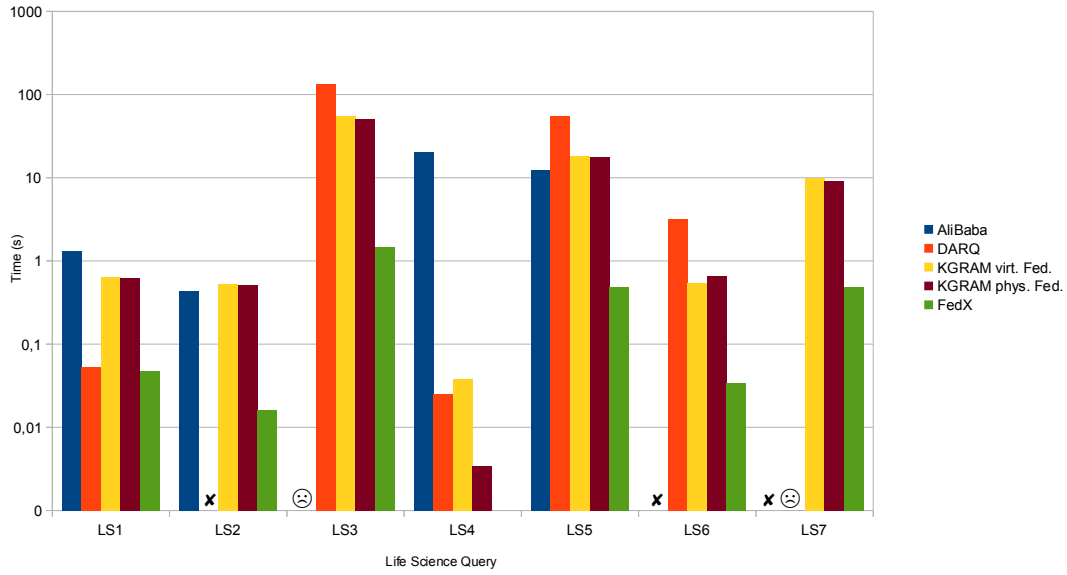

Figure 8.4: Compared to state-of-the-art federation approaches, the KGRAM federation engine is able to process all FedBench Life Science queries. It delivers results in a reasonable amount of time, sometimes faster than AliBaba or DARQ.

8.2.2.3 Experiment 2: comparing KGRAM and FedX engine in a controlled distributed computing infrastructure.

Experiment 1 has been proposed to give an idea on how KGRAM performs with respect to state-of-the-art approaches. We evaluated KGRAM in a supposed less favorable environment, since KGRAM was deployed on a real distributed computing infrastructure (Grid'5000), and other engines have been evaluated on a simulated distributed environment (*virtual federation*).

To more precisely evaluate the relative performance of KGRAM and FedX, we propose in this second experiment, (i) to deploy them in exactly the same distributed computing infrastructure and (ii) to measure the distributed query processing time for the FedBench Life-Science datasets and queries.

Distribution setup

This second experiment has been performed few months after experiment 1. During this period both the FedBench datasets and queries have slightly evolved. The updated Life Science Data collections have been fragmented over 5 distributed data sources which totalizes more than 52 millions triples.

To evaluate FedX (version 2.0 build 73, packaged with FedBench framework version 3), we reserved 6 nodes of the "Suno" Grid'5000 cluster (each node has 2 Intel CPUs (2.26GHz) with 4 cores per CPU, 32GB of RAM, and 519GB of disk storage). One node

Data source	*Linked Data collection*	*Size (triples)*
#1	ChEBI	7.3M
#2	DBpedia sub-set #1	25.3M
#3	DBpedia sub-set #2	18.3M
#4	DrugBank	0.7M
#5	KEGG Drug	1M

Table 8.4: Updated FedBench Life Science data collections (52M triples) fragmented over 5 data sources.

was reserved for the execution of the FedX federation engine, the other 5 nodes were reserved to expose the 5 data sources through a Fuseki SPARQL endpoint (version 0.2.5). It has to be noted that due to some incompatibilities, it has not been possible to deploy FedX with OpenRDF-Sesame SPARQL endpoints (versions 2.6.10 and 2.7.0-beta1) as described in [Schwarte et al., 2011].

To evaluate KGRAM, we deployed a similar environment as previously described in experiment 1 through the *physical federation*. We reserved 6 nodes of the "Suno" Grid'5000 cluster, one of which was dedicated to the KGRAM federation engine, and the 5 remaining nodes exposing the 5 data sources through KGRAM endpoints.

Results and discussion

Due to the update of both the FedBench datasets and queries, it has not been possible, with the two engines, to obtain results for queries LS3 and LS5.

Table 8.5 reports the evaluation of the FedBench Life-Science queries 1, 2, 4, 6, and 7, with FedX. We first observe that the behavior of FedX is different in a real distributed infrastructure compared to results observed in a *virtual federation* setup [Schwarte et al., 2011]. Especially in the case of LS1 and LS6, the measured evaluation times are longer in this physical federation. In addition, we observe a noticeable variability in the measured times: LS6 ended with a timeout (set to 3 minutes) for 4 of the 10 runs, and we measured for LS2 and LS7 an important standard deviation (respectively 1575 ms and 1671 ms) compared to the mean evaluation time (respectively 1562 ms and 909 ms).

Table 8.6 reports the consecutive 10 evaluations of the 7 FedBench Life-Science queries with KGRAM. We observe a noticeable stability in the measured evaluation times. The observed standard deviation for LS1, LS2, LS4 and LS6 varies from 8 ms to 35 ms when the mean evaluation time varies from 322 ms to 1318 ms. The longest evaluation (11 s) is observed for query LS7.

Figure 8.5 compares FedX and KGRAM mean evaluation times for LS1, LS2, LS4, LS6 and LS7. The error bars show $\pm$ 1 stdev. We observe comparable times for LS1 which was not the case in experiment 1. FedX is the fastest for LS4 and LS7. This might be due to its advanced query plan optimizations (e.g. triple pattern grouping). However, KGRAM performs better than FedX for LS2 and LS6. We also observe, for FedX, a huge variability in the measured times for LS2 and LS7 which makes any interpretation difficult. Given

Run	LS1	LS2	LS4	LS6	LS7
#1	710	3062	22	timeout	5666
#2	482	3055	19	timeout	426
#3	468	3058	16	31201	389
#4	462	3054	16	26639	395
#5	462	68	14	30696	395
#6	465	3057	15	30209	369
#7	460	68	15	timeout	371
#8	459	69	15	28525	357
#9	461	69	16	timeout	357
#10	460	67	15	28490	367
Mean Eval. Time (ms)	488.9	1562.7	16.3	29293.33	909.2
Std. Dev. (ms)	77.98	1575.34	2.40	1716.17	1671.50

Table 8.5: 10 consecutive evaluations of the FedBench Life-Science queries with the FedX engine show an important variability for LS2 and LS6.

Run	LS1	LS2	LS4	LS6	LS7
#1	661	429	355	1407	11031
#2	654	412	317	1310	10858
#3	642	404	319	1331	11030
#4	653	403	319	1313	12018
#5	613	412	319	1325	10735
#6	613	409	319	1309	10772
#7	613	407	322	1302	10880
#8	614	405	319	1280	12076
#10	605	407	320	1325	12366
Mean Eval. Time (ms)	627.4	408.8	322.9	1318.8	11259.7
Std. Dev. (ms)	22.27	8.05	11.34	35.08	630.06

Table 8.6: Stable 10 consecutive evaluations of the FedBench Life-Science queries with the KGRAM engine.

that KGRAM shows little variability on the computing infrastructure used for this experiment, infrastructure variability cannot explain FedX results variability. FedX implements non trivial query plan optimizations through a multi-threaded code. Problems with either the plans optimizer or the parallel implementation most probably accounts for the variability observed.

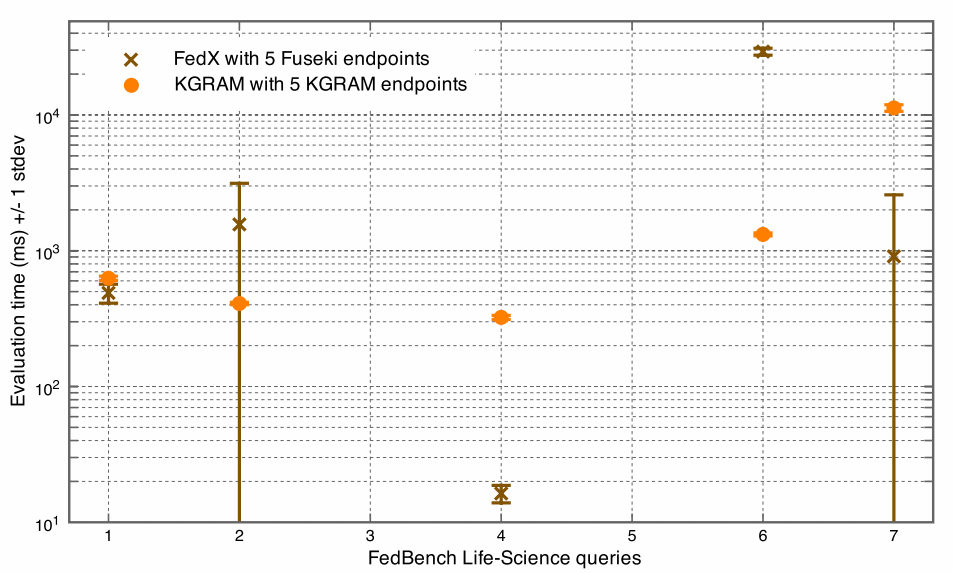

Figure 8.5: Comparing FedX and KGRAM mean evaluation time through the FedBench Life-Science queries.

To conclude this second experiment with the FedBench benchmark, we can say that (i) providing efficient optimized distributed SPARQL query evaluation is non trivial, and that (ii) KGRAM already performs in reasonable amount of time with a noticeable stability. KGRAM is a promising approach for federated semantic querying since it opens opportunities for new optimizations.

To summarize, these two large-scale experiments show that:

- *Performance: through a real distributed setup of the FedBench benchmark, KGRAM-DQP offers results in line with its competitors (and is most of the time better than AliBaba or DARQ) ;*
- *Expressivity: KGRAM-DQP allows for the distribution of most of the SPARQL 1.1 features, including SPARQL path expressions, and allows in addition for distributed inferences.*

8.3 Federating distributed and heterogeneous neuroscience data sources with KGRAM

We show in this experiment how the distributed querying strategies proposed in section 4 allows for federating multiple distributed data sources, in the context of the NeuroLOG platform, and opens new opportunities through bridging external data sources published as Linked Data.

Open environments associated to the web of Linked Data generally imply a high diversity in data formats, data models or schemas, thus raising heterogeneity issues. These issues are lowered through the continuously growing adoption of the RDF Semantic Web standard for the representation of knowledge and the publication/standardization of controlled vocabularies (SKOS[3], RDFS[4] or OWL[5]). In this experiment, we demonstrate how KGRAM can achieve on-the-fly data transformation to address heterogeneity issues. More precisely, at runtime and through predefined mappings, we propose to (i) transform a graph-based semantic query into the target data source query languages, and (ii) return back the results in the form of triples combined to build a knowledge graph result.

Finally, we show in this experiment how the KGRAM federation engine can be used to bridge the NeuroLOG and the NeuroLex neuroimaging knowledge bases and more generally, how the generic distributed graph querying proposed in chapter 4 can perform Open Linked Data querying and reasoning. Our motivation is as follows. We want to show that in spite of a consequent modeling effort addressing the neuroimaging area and how neuroimaging entities are modeled with respect to the DOLCE foundational ontology, the NeuroLOG federation can still benefit from ontologies addressing the same domain but with different objectives (the NeuroLex [6] neuroscience lexicon). More precisely, we show in this experiment how distributed NeuroLOG datasets can be searched by reasoning on the NeuroLex ontology concepts.

To summarize, this experiment addresses the following objectives:

- *Querying multiple distributed data sources through Semantic Web graph querying ;*
- *Mediating structural heterogeneity (relational and semantic data sources) through abstract graph representation and querying ;*
- *Bridging the NeuroLOG and the NeuroLex neuroimaging knowledge bases through Linked Data querying to benefits from several modeling efforts (knowledge sharing).*

[3] http://www.w3.org/TR/skos-reference
[4] http://www.w3.org/TR/rdf-schema
[5] http://www.w3.org/TR/owl2-overview
[6] http://neurolex.org/wiki/Main_Page

8.3.1 Material and methods

The NeuroLOG federation. The distributed system considered in this evaluation is the NeuroLOG platform [Montagnat et al., 2008b]. NeuroLOG is a middleware federating data and computational resources from collaborating neuroscience centers. The prime objective of NeuroLOG is to adapt non-invasively to the legacy environments deployed in each participating center, so that each site remains autonomous in the management of its internal resources and tools, while benefiting from the multi-centric medical studies support from the middleware. The NeuroLOG platform is a distributed system federating 5 neuroscience centers spread over France. Each center exposes a sensitive raw medical data source accessible from the whole federation. On each site, the data source is composed of raw medical image files, and description metadata linked to these files (information on image data acquisition, data content, neuropsychological tests associated to images, etc) [Michel et al., 2010]. The source metadata is often represented and managed in relational databases for historical reasons. The amount of data federated through the platform and its sensitive nature disqualify centralized data warehousing approaches.

Blue components in Figure 8.6 sketch the architecture of the NeuroLOG middleware. On each neuroscience site is deployed an independently managed legacy relational database that is complemented by a NeuroLOG middleware database. The multi-centric studies conducted by neuroscientists may be perceived under several facets [Gibaud et al., 2011a], involving both the native relational data representation and a semantic data representation enabling richer queries. The *DataFederator* commercial tool [SAP] is used to dynamically federate relational data sources into a unified view. It can perform SQL queries that are distributed over all platform data sources. It includes both a mediation layer that aligns heterogeneous relational databases schemas, and rewrite SQL queries applying to the federated view to match the various source schemas. The data mediation semantic alignment is based on a domain ontology, called ONTONEUROLOG that was developed in the context of this project. A federated relational schema is derived from OntoNeuroLOG, serving as the federated view schema. In addition to this relational representation, a semantic representation of the same data sources was created to enable richer querying features delivered by Semantic Web query engines. A centralized approach was adopted, where all relational data sources are mapped to RDF triples (using the MetaMORPHOSES tool [Svihla and Jelínek, 2007]) and aggregated in a unique semantic repository. The NeuroLOG platform thus exposes a dual view of the federation metadata, enabling both dynamic SQL querying and static SPARQL querying. Although very flexible, this system is also confusing for end users due to the dual view of all data entities, and the semantic repository is subject to limitations of a static, centralized system.

To overcome these issues, the NeuroLOG platform was extended in this experiment with the KGRAM query engine introduced in section 4.2.1. Figure 8.7 sketches the upgraded NeuroLOG middleware architecture with the KGRAM federated engine. A KGRAM remote producer was deployed on top of each site legacy database. This endpoint exposes the site data content in RDF through its *Producer*. Depending on the site

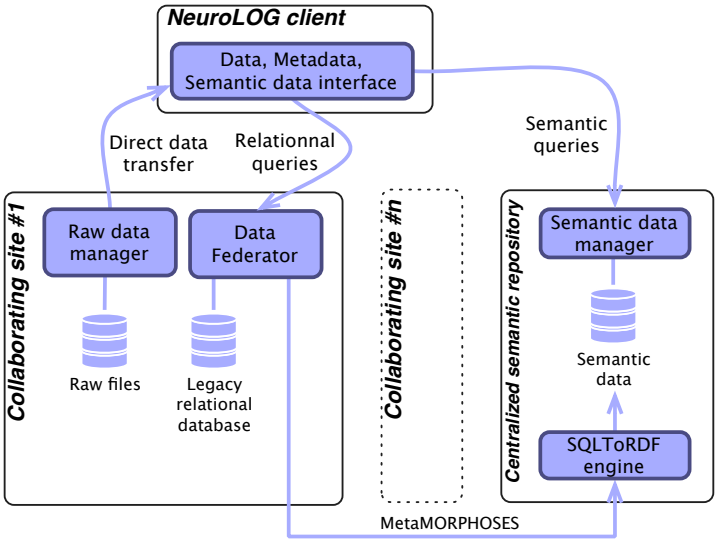

Figure 8.6: Data management layer of the initial NeuroLOG platform.

deployment option, it either interfaces directly to the site native relational database (option ① in Figure 8.7), or accesses an RDF repository representation of this legacy database (option ②). It was performed through an ad-hoc converter able to transform inbound triple pattern requests from the remote producer into corresponding SQL queries and to map the SQL results to semantic entities. The site RDF repository was initially created with MetaMORPHOSES and is accessed through KGRAM 's native RDF remote producer. Consequently, NeuroLOG 's centralized RDF repository (from Figure 8.6) is not needed anymore. This setting enables the unified querying of the RDF repositories and the platform legacy relational databases through the SPARQL language. It solves the problems associated to using a central repository by dynamically enabling direct access to the federation data sources, and distributing the query load over the federation data servers. It proposes a single view over all data.

The NeuroLex knowledge base. NeuroLex [Imam et al., 2011], supported by The Neuroscience Information Framework (NIF[7]) and the International Neuroinformatics Coordinating Facility (INCF[8]) is a dynamic Neuroscience Lexicon which describes 22,273 neuroscience terms (292 neurons and 940 brain parts, 151 spinal cord parts and 10 other parts of the nervous system). It aims at delivering a standard lexicon for neuroscience entities, covering their meaning and their classification, to address data integration issues generally faced in the neuroscience area.

In this Linked Data experiment, neuroimaging data from the NeuroLOG federation is linked with neuroscience "open" knowledge capitalized through the NeuroLex initiative,

[7] http://www.neuinfo.org
[8] http://www.incf.org

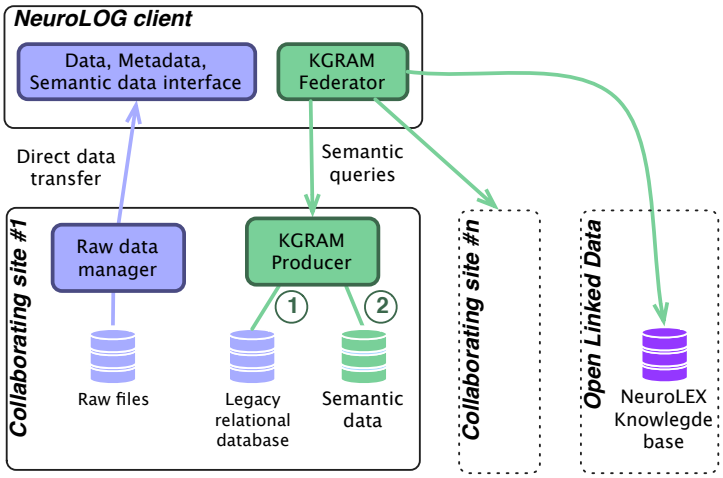

Figure 8.7: Distributed semantic and relational metadata querying through the KGRAM federated engine.

allowing neuroscientists involved in the NeuroLOG federation to benefit from the NeuroLex lexicon and its semantic wiki interface. Thanks to KGRAM versatility, the NeuroLOG platform is easily extended with a new data source that exposes the NeuroLex ontology. To bridge the two knowledge bases, an ad-hoc semantic alignment in which all NeuroLOG datasets are annotated with the NeuroLex *Label* property corresponding to their medical image modality has been implemented.

Query environments. In this experimental study, we compare three query environments using the NeuroLOG platform deployed in production. The first query environment (*relational federation*) only exposes heterogeneous relational databases, virtually integrated through the *DataFederator* commercial middleware. It corresponds to the seminal NeuroLOG platform deployment. Two other environments (*semantic federations*) expose heterogeneous data sources virtually integrated through the KGRAM framework either by dynamic access to the legacy databases (option ①) or by static access to the site RDF repository created from the legacy database (option ②). In the following experiments, we consider an environment named RDF *semantic federation* were all sites are configured with option ②. It corresponds to a modification of the NeuroLOG platform where the central semantic repository is spread over all participating site. Finally, The SQL+RDF *semantic federation* environment is completely heterogeneous, combining one relational data source and other semantic data sources.

Typical queries. The query illustrated in Listing 8.3 below reflects a real clinical user concern and is illustrative of the context of linked neuro-data distributed over collaborating neuroscience research centers. Indeed, this query aims at searching for datasets (acquired with Gadolinium contrast agent) associated to patients (join performed line 4) in the context of multi-centric studies addressing the Multiple Sclerosis pathology.

Listing 8.3: SPARQL query (Q_1) aiming at retrieving patient study and dataset information in the context of the Multiple Sclerosis disease.

```
SELECT distinct ?patient ?study ?dataset ?dsName WHERE {
        ?dataset linguistic−expression:has−for−name ?dsName .
        ?patient examination−subject:has−for−subject−identifier ?clinID .
        ?patient iec:is−referred−to−by ?dataset .
        ?study study:involves−as−patient ?patient .
        FILTER (CONTAINS(?clinID, 'MS') && CONTAINS(?dsName, 'GADO')) }
```

Since two of the NeuroLOG partners are used to collaborate in the context of the Multiple Sclerosis disease, they potentially host either target patient or dataset entities which justify to transparently distribute the query over the federation.

During the evaluation of this typical query, all types of medical images will be searched in order to match a "GADO" tag generally meaning that the Gadolinium contrast agent was used during image acquisition. But the evaluation of this query can be considerably enhanced by exploiting clinical knowledge represented in an ontology. Indeed Gadolinium is used in the context of magnetic resonance (MR) acquisitions (T1 or T2 weighted MRIs for instance) but generally not in the context of any other modality (Ultrasound for instance). By exploiting this domain knowledge with KGRAM , the query time can be seriously reduced since all non-MR datasets (such as Ultrasound datasets) are excluded. To achieve this, the triple pattern (?dataset rdf:type dataset:MRDataset) should be added to the query, which leads, under RDFS entailment, to less intermediate results to be transferred, and an overall faster distributed query evaluation.

The following query from Listing 8.4 achieves the same clinical objective but makes use of SPARQL Property Path expressions. This syntax aims at representing paths between two resources by only specifying, in the form of patterns, the sequence of mandatory, optional, reverse, or multiple repetition of properties linking the resources together. It brings a high expressivity to SPARQL and it is particularly adapted in the context of graph-based querying.

Listing 8.4: SPARQL 1.1 property path expressions to simplify the previous query (list. 8.3).

```
SELECT distinct ?patient ?study ?dsName WHERE {
        ?patient iec:is−referred−to−by/linguistic−exp:has−for−name ?dsName .
        ?patient examination−subject:has−for−subject−identifier ?clinID .
        ?study study:involves−as−patient ?patient .
        FILTER (CONTAINS(?clinID, 'MS') && CONTAINS(?dsName, 'GADO'))
}
```

Queries involving path expressions cannot be easily expressed in SQL. It is thus difficult to implement it with traditional relational databases. As a full SPARQL 1.1 interpreter, compliant with property path expressions, KGRAM allows performing such graph-based information retrieval against SQL data sources, which would not have been possible with

traditional SQL query engines.

Listing 8.5: SPARQL query exploiting the NeuroLex taxonomy for medical image modalities to search imaging data provided through the NeuroLOG platform.

```
1 SELECT ?patient ?dataset ?dsName WHERE {
2         ?t property:Label \"MRI protocol\"^^xsd:string .
3         ?s rdfs:subClassOf* ?t .
4         ?s property:Label ?label .
5         ?dataset property:Label ?label .
6         ?dataset linguistic-expression:has-for-name ?dsName .
7         ?patient iec:is-referred-to-by ?dataset .
8 }
```

Figure 8.5 illustrates a SPARQL query which firstly lists all NeuroLEX subclasses of *MRI protocol* and their associated *label*, and secondly exploits these labels to search for relevant medical images, and their corresponding patients provided by the distributed NeuroLOG data providers. We could easily imagine that the variety of neuroscience ontologies do not share the same modeling objectives. Some may focus on particular pathologies, others may focus on neuro-data processing. In that context, and beyond the neuroscience area, the KGRAM framework provides a transparent mean to query distributed and heterogeneous data sources while potentially benefiting from several ontologies.

Linked data reasoning and querying framework Distributed querying and reasoning performed in this experiment are supported by the distributed querying strategies proposed in chapter 4 and by the KGRAM extension proposed in section 7.3.

KGRAM comes with a default implementation of its *Producer* interface for RDF sources. To seamlessly cope with querying data sources exposed as RDF or as traditional relational data, we provide KGRAM with a mediation capability aiming at querying legacy relational databases with SPARQL queries. We developed an implementation of KGRAM *Producer* for handling relational data as follows. From a triple pattern forming the initial SPARQL query, this producer is able to generate on-the-fly a corresponding SQL query. The result of such a query is a set of tuples which are mapped to the variables of the original SPARQL query to build result graph triples.

The query illustrated in Listing 8.6 consists in constructing relations between ?patient and ?dataset variables through the iec:is-referred-to-by property. But this property is not explicit in the relational database. We consecuently perform an SQL query over the Dataset table (lines 5 and 6) to find the identifier of datasets associated to the patient IRISA-SS-7. The two selected fields are then mapped to the ?x and ?y SPARQL variables (line 7). Finally, ?patient and ?dataset resource URIs are reconstructed with a prefix and the value obtained through ?x and ?y from the relational database (line 8 and 9).

To benefit from intermediate results gathered from contributing remote data sources, we also implemented a *bind join* optimization strategy dedicated to relational data

Listing 8.6: Nested SQL query into an extended SPARQL query.

```
CONSTRUCT {
        ?patient iec:is−referred−to−by ?dataset
} WHERE {
        {SELECT sql(<jdbc:mysql://db>, <jdbc.Driver>,'user', 'password',
                        "SELECT Dataset.subject_id, Dataset.dataset_id FROM Dataset
                        WHERE Dataset.subject_id LIKE 'IRISA−SS−7' ") as (?x, ?y)
                        uri(concat("http://prefix#subject−",?x)) as ?patient
                        uri(concat("http://prefix#dataset−",?y)) as ?dataset WHERE {}}
}
```

sources. Indeed through a template SQL query, we dynamically generate value constraint such as the LIKE clause, thus limiting the amount of transferred results, directly from the data source.

8.3.2 Results and discussion

Distributed querying over structurally heterogeneous knowledge bases. This experiment – and its distributed setup illustrated in Figure 8.7 – shows how the distributed querying strategies developed in chapter 4 allow querying the knowledge bases of 4 distributed collaborating neuroscience centers through SPARQL 1.1. It additionally shows the feasibility of dynamically mediating heterogeneous knowledge bases, namely RDF and SQL repositories, thus providing virtual data integration overcoming common issues encountered in data warehousing approaches such as periodic synchronization or fault tolerance. Finally, contrary to the SPARQL 1.1 federation extension which needs explicit distribution directives (*Service* clauses), the proposed federation mechanism is transparent with respect to the initial SPARQL queries since no hypothesis are made at query design-time on the content of the distributed data sources.

Performance. Table 8.7 compares the distributed query processing times of the *relational federation* (using *DataFederator*) and both *semantic federations* (RDF and RDF +SQL) using KGRAM . Query *Q1* corresponds to Listing 8.3 and query *Q2* searches for datasets acquired through the T2-weighted MRI modality. *Q2* leads to only 5 results and is thus, a very selective query. To be robust against variability observed in real distributed systems, results are averaged over three query runs. The average query execution time $\pm$ one standard deviation is displayed in table 8.7. It shows that for *Q1*, leading to 336 remote invocations, the query times are better with the optimized SQL federation engine than with the semantic federation, but it remains in the same order of magnitude. For very selective queries such as Q_2, we observe comparable query times for all environments.

Expressive queries and reasoning capabilities. When using a knowledge based system, such as the proposed KGRAM federated engine, to query heterogeneous data sources, query designers possibly benefit from inferences based on domain knowledge:

	Semantic federation		*Relational federation*
	RDF	RDF+SQL	SQL
Q_1	6.13 s $\pm$ 0.05	11.76 s $\pm$ 0.05	3.03 s $\pm$ 0.25
Q_2	0.60 s $\pm$ 0.03	1.53 s $\pm$ 0.14	1.52 s $\pm$ 0.62

Table 8.7: Comparison of the KGRAM distributed query processing with the *DataFederator* commercial tool achieving relational federated querying.

KGRAM implements the RDFS entailment regime (subsumption between classes or properties), and other inferences based on algebraic properties like the transitivity or symmetry of properties and inverse properties. It thus provides a richer query interface to legacy databases participating into the federation, compared to traditional SQL query engines.

In addition, in the context of collaborative platforms exposing legacy relational databases, we argue that the design of queries is more intuitive through knowledge-based languages such as SPARQL than through traditional relational languages such as SQL. Indeed, the navigation through links between entities is explicit in SPARQL but it is implicit in SQL and generally requires intermediate joins. Listing 8.7 illustrates the SQL query corresponding to the SPARQL query of Listing 8.3.

Listing 8.7: An SQL translation of the previous sample SPARQL query (fig. 8.3).

```
1 SELECT Subject.subject_id, Subject.subject_common_identifier, Dataset.name
2 FROM Study, Subject, Dataset, Rel_Subject_Study WHERE
3         Rel_Subject_Study.Subject_subject_id = Subject.subject_id AND
4         Rel_Subject_Study.Study_study_id = Study.study_id AND
5         Dataset.Subject_subject_id = Subject.subject_id AND
6         Subject.subject_common_identifier LIKE '%MS%' AND
7         Dataset.name LIKE '%GADO%'
```

Whereas joins are naturally expressed in the SPARQL query (line 4 of Listing 8.3), it is not the case in SQL since a join table may be needed (Rel_Subject_Study table, line 3) and must be explicit (line 3, 4 and 5). This definitely complicates the query design, generally considered as a complex, error-prone, and time consuming activity.

Knowledge sharing By allowing the NeuroLOG federation to benefit from other modeling initiatives such as the NeuroLex neuroscience lexicon, we show that, after a semantic alignment, it is possible to exploit NeuroLOG datasets by reasoning on NeuroLex concepts. Even if this experiment from a clinical application point of view, does not show a real added valued – mainly because a similar query than Listing 8.5 would have been possible by exploiting exclusively the OntoNeuroLOG ontology – it opens new perspectives in the fields of semantic interoperability and translational research.

Indeed, if we consider the situation of a new partner site reaching the NeuroLOG platform. This partner may already have achieved a significant modeling effort resulting in a specific ontology, namely Onto_{new}, addressing for instance the conceptualization of a specific pathology, with the related data processing tools and datasets. With the

proposed distributed querying framework, it would be possible to seamlessly query the NeuroLOG federation and the new partner knowledge base, before a full semantic integration of Onto$_{new}$ into the OntoNeuroLOG modeling framework. However, the proposed approach would require a lightweight integration. This integration step would consists in bridging the data of the new site with concepts of the OntoNeuroLOG ontology to search for the new data source through the OntoNeuroLOG concepts, or bridging the data of the NeuroLOG federation with the concepts of Onto$_{new}$, to immediately benefits from Onto$_{new}$ at the scale of the NeuroLOG federation.

Linked Data querying also opens new opportunities in the field of translational research to the NeuroLOG federation. We could easily imagine a scenario where some medical images provided by a clinical partner are accompanied with patient genetical information. The NeuroLOG platform could thus be extended with a connection to a SPARQL endpoint on top of the the GenBank[9] which provides annotated DNA sequences. New SPARQL queries could finally be designed to search for neuroimaging resources by providing genetical criterions.

To summarize, this experiment brings the following results:

- *Efficient distributed query processing through the* SPARQL *1.1 Semantic Web language ;*
- *Transparent distributed query processing, since no hypothesis are made at query design time on the content of data sources, and consequently no directives are needed to send subqueries to a particular data source ;*
- *Dynamic mediation over structurally heterogeneous graph-based and relational knowledge bases ;*
- *Expressive distributed querying supporting both property path expressions and* RDFS *entailment regime ;*
- *Knowledge sharing through Linked Data querying opportunities for the NeuroLOG platform (eased integration of new partners).*

[9] http://www.ncbi.nlm.nih.gov/genbank

8.4 A real-life medical imaging simulation workflow: semantic mash-up experiment to infer meaningful experiment summaries

The VIP project[10] is a french initiative dedicated to multi-modal and multi-organ medical image simulation [Forestier et al., 2011a] [Glatard et al., 2013]. Integrating several medical image simulators and anatomical models into a unique platform raises challenging interoperability issues. A semantic approach has been adopted to tackle interoperability and to enhance the reusability of simulator components, anatomical models and the resulting simulated data. To assist users in the setup of new simulation experiments (simulation workflows) or in the parametrization of existing simulators, the system relies on knowledge bases describing the simulation models, simulation components, and simulated data involved.

The enactment of simulation workflows produces large amounts of heterogeneous data. Some of it is intermediate data, necessary to achieve a precise step of the simulation workflow while the other part, the resulting simulated data, represents a real interest for end-users. But the size and the diversity of the produced data makes it difficult to comprehend from the end-user perspective. Moreover, in the case of a failure during an experiment, or when abnormal results are observed, it can really be challenging to identify the causes of failures or abnormalities.

In the following experiment, we exploit (i) bridges, developed in chapter 6, between scientific workflow environments and community knowledge, and (ii) distributed query processing strategies proposed in chapter 4, to cope with issues commonly faced by the clinical communities when performing complex medical imaging experiments through scientific workflows enacted over large-scale distributed infrastructures.

More precisely, we propose in this experiment to track workflow provenance for one of the four medical image simulation workflows of the VIP platform. This aims at enhancing the overall quality of simulation workflows executed through the VIP infrastructure. Quality covers here both a technical concern – allowing for workflow designers and experiment operators, to more easily determine the cause of failure or abnormalities – and a reliance concern making scientists more confident in the data produced through their experiments, since the reproducibility of simulation experiments is made easier and data lineage can be controlled.

We also show in this experiment that provenance information, recorded from workflow runs, can be considered as a raw technical material to infer new domain-specific knowledge based on the VIP ontology. These inferences can consequently automate the semantic annotation of data produced through workflow runs. The resulting inferred meaningful statements can then be seen as "semantic experiment summaries" in which a minimal set of statements link together simulation experiment results to experiment pa-

[10] funded by the French National Agency for Research under grant ANR-09-COSI-03

rameters (or data processing services) through a dedicated vocabulary (the VIP ontology) originating from the end-users community (the VIP project partners).

We implemented this experiment in a distributed setup which differs slightly from the original VIP platform setup. Indeed, whereas semantic data is centralized under the VIP portal, we propose here to fragment the provenance data over two provenance data sources. This choice is motivated by the fact that in general, e-science workflow environments perform remote invocations for each of their constituting data processors, and data processor invocations are relocated on a dedicated computing unit or infrastructure. We could thus easily consider that execution traces are generated locally to the computing unit, and the overall provenance data is constituted from these fragmented and distributed execution traces. In addition we consider in a third data source a semantic service registry that exposes semantic service descriptions involved in inference rules.

This experiment addresses the following three objectives :

- *Tracking of data provenance to enhance quality, considering both technical quality (engineers) and data quality (scientist end-users) ;*
- *Inferring new domain-specific knowledge through workflow runs, and thus automating the semantic annotation of workflow results finally provide meaningful experiment summaries ;*
- *Addressing provenance in scientific workflows as distributed linked data.*

8.4.1 Materials and methods

Sorteo simulation workflow. Sorteo [McLennan et al., 2009] is a Monte Carlo-based medical image simulator dedicated to the production of synthetic Positron Emission Tomography (PET) data.

The SORTEO simulation workflow is presented in Figure 8.8. Blue boxes represent both compute intensive activities whose executions are relocated on dedicated computing infrastructure, the EGI grid, and lightweight activities executed locally, by the workflow engine. Green ellipses represent input or output data. Intermediate data transiting over the workflow are not represented in the diagram, but are present through data flows (black arrows in the diagram).

The main inputs are the *protocol*, storing all simulation parameters, and the *phantom* representing the object model to be virtually imaged. The SORTEO simulation workflow produces a single output, a *sinogram*[11], representing the simulated PET data.

The core of the simulation consists in two steps:

(i) the parallel computation of "singles" through the *sorteoSingles* activity ;

[11] A sinogram represents the information of a single slice under all angles of projection, and a projection represents the information from all slices, but for a fixed angle.

(ii) the parallel computation of the "emissions" through the *sorteoEmission* activity.

The remaining activities can be considered either as pre-, or post-processing steps, needed to assemble simulation parameters, or to convert data throughout the simulation workflow.

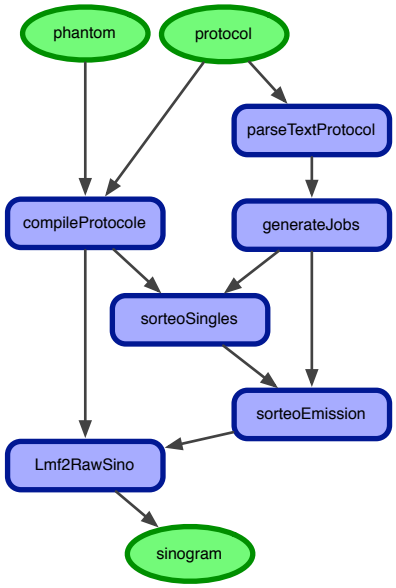

Figure 8.8: Simplified graphical representation of the SORTEO PET medical image simulation workflow.

Instrumented workflow enactor. In this experiment, we rely on the MOTEUR-S workflow engine as briefly presented in section 7.2.2. MOTEUR-S stands for a "semantized" extension of the original MOTEUR workflow engine through two plugins: a semantic provenance plugin and a semantic service annotator and catalog.

The semantic provenance plugin is responsible for on-the-fly tracking of provenance information. For each invocation of a single processing step, OPM statements describing the *Process*, the input and output *Artifacts*, and their *UsedBy* or *WasGeneratedBy* causal dependencies are generated and persisted through the NEUSEMSTORE semantic repository introduced in section 7.2.

The semantic service annotator and catalog plugin is responsible for the annotation and the cataloguing of services involved in Moteur workflows. The service catalog helps in inferring new meaningful experiment summaries since semantic service annotations are necessary to perform inferences over workflow runs. Services are semantically described through the following annotations clearly making the distinction between the nature and the role of parameters as proposed in chapter 6.

- *Functionality:* describes the class of action realized by a service invocation.
- *Parameter nature:* describes the intrinsic nature of the input or output service parameters.
- *Parameter role:* describes how a parameter is related to the service. This role aims at disambiguating the service annotation because in some cases, several parameters may share the same nature but are interpreted differently from the service perspective.

Once annotated, the service descriptions are uploaded to the service catalog. The service catalog is part of the NeuSemStore framework, and allows graphically navigating into classified services, sorted by functionality. More selective service searches might be performed through dedicated SPARQL queries.

Provenance semantic annotations. The following listings illustrate in TURTLE[12] syntax, the main provenance statements describing the invocation of the last processing step of the workflow.

Listing 8.8: OPM statements describing the Lmf2RawSino process invocation.

```
<http://neurolog.techlog.anr.fr/processingtoolexec#LMF2RAWSINO−a2a7c751949f40bbb06b0136033ff919>
    a <http://purl.org/net/opmv/ns#Process> ;
    rdfs:comment "LMF2RAWSINO"^^<http://www.w3.org/2001/XMLSchema#string> ;
    <http://openprovenance.org/model/opmo#account>
        <http://neurolog.techlog.anr.fr/processingtoolexec#workflow−38979af8272641d0b4564bed91df48c5>
```

Listing 8.8 illustrates the registration of the invocation of the Lmf2RawSino process. An instance of the *Process* class is created with the following URI, http://neurolog.techlog.anr.fr/processingtoolexec#LMF2RAWSINO-a2a7c751949f40bbb06b0136033ff919, constructed from a prefix, the name of the workflow processor and a uniform unique identifier (UUID). Additionally, this process invocation is attached to an OPM *Account*, which represents the overall workflow invocation. Note that all OPM *Artifacts* and *Processes* registered through a single workflow invocation are also attached to an OPM *Account*.

Listing 8.9: OPM statements describing the WasGeneratedBy dependency between the output sinogram and the Lmf2RawSino process.

```
<http://openprovenance.org/ontology#WasGeneratedBy−703fe5fdc1784923bcf90d2f5db12bac>
    a <http://openprovenance.org/model/opmo#WasGeneratedBy> ;
    <http://openprovenance.org/model/opmo#account>
        <http://neurolog.techlog.anr.fr/processingtoolexec#workflow−38979af8272641d0b4564bed91df48c5> ;
    <http://openprovenance.org/model/opmo#cause>
        <http://neurolog.techlog.anr.fr/processingtoolexec#LMF2RAWSINO−a2a7c751949f40bbb06b0136033ff919> ;
    <http://openprovenance.org/model/opmo#effect>
        <http://openprovenance.org/ontology#Artifact−62eb46e9c8f74c43bd04bd370480a871> ;
    <http://openprovenance.org/model/opmo#role>
        <http://openprovenance.org/ontology#Role−c41b0ed5768d427f84e8a0cab0707833> ;
    <http://openprovenance.org/model/opmo#time>
        <http://openprovenance.org/ontology#OTime−6dc8f61c7ea24c6fa96e7a3bd6193cc5> .
```

Listing 8.9 illustrates the causal "data production" dependency registered between the previous Lmf2RawSino process invocation and the output sinogram. This dependency is represented by an instance of the *WasGeneratedBy* OPM class and is identified similarly to processes. This instance is linked to both the process invocation through the *cause* OPM property, and the *Artifact* describing the output sinogram through the *effect* OPM property. In addition, the process input or output ports are described through the *role* OPM property linking together the data dependency and an instance of the OPM *Role* class which corresponds to the label of the process input or output port. Finally, the data

[12] TURTLE provides a textual syntax for RDF and enhances the readability of RDF documents, http://www.w3.org/TeamSubmission/turtle

production is timestamped through the OPM *time* property towards an instance of the OPM *OTime* class.

Listing 8.10: OPM statements describing the sinogram produced as an output of the Lmf2RawSino process.

```
<http://openprovenance.org/ontology#Artifact−62eb46e9c8f74c43bd04bd370480a871>
    a <http://purl.org/net/opmv/ns#Artifact> ;
    <http://openprovenance.org/model/opmo#account>
        <http://neurolog.techlog.anr.fr/processingtoolexec#workflow−38979af8272641d0b4564bed91df48c5> ;
    <http://openprovenance.org/model/opmo#avalue>
        <http://openprovenance.org/ontology#AValue−e856c3dbad6a48ddad6707d863e86ae3> .

<http://openprovenance.org/ontology#AValue−e856c3dbad6a48ddad6707d863e86ae3>
    a <http://openprovenance.org/model/opmo#AValue> ;
    <http://openprovenance.org/model/opmo#account>
        <http://neurolog.techlog.anr.fr/processingtoolexec#workflow−38979af8272641d0b4564bed91df48c5> ;
    <http://openprovenance.org/model/opmo#content>
        "lfn://lfc−biomed.in2p3.fr/grid/biomed/creatis/vip/data/users/rafael_silva/sorteo−2/
        24−01−2012_10:13:30/dataLMF.ccs.sino"^^<http://www.w3.org/2001/XMLSchema#anyURI> .
```

Finally listing 8.10 describes the OPM *Artifact* corresponding to the output sinogram of the *Sorteo* PET simulation workflow. An *Artifact* instance is created. It has already been attached to the *WasGeneratedBy* causal dependency through the *effect* property of the previous listing. An *Artifact* is an abstract entity and OPM allows for associating their concrete values. The *Artifact* is thus linked to an instance of the *AValue* OPM class through the *avalue* property. Finally a content is associated to the value through the OPM *content* property. This content finally gives the logical file name (LFN) of the sinogram, an URI locating the data on the EGI grid infrastructure. Data might be later on downloaded through a dedicated data transfer interface[13].

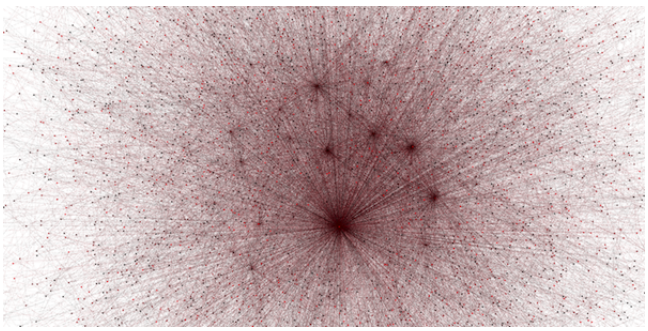

Figure 8.9: A screenshot of the full OPM graph tracked through the invocation of the Sorteo workflow.

Figure 8.9 and Figure 8.10 provide graphical representations[14] for the provenance graph of a single invocation of the Sorteo workflow. While figure 8.9 represents the

[13] by using the GLite user interface for instance.

[14] the graphical representations have been generated through the Gephi [Bastian et al., 2009] visualization tool (https://gephi.org) and the semantic web import plugin (https://gephi.org/plugins/semanticwebimport).

full graph composed of 4523 nodes and 15154 edges, figure 8.10 represents a simplified graph in which some nodes have been removed such as the unique instance of the OPM *Account* allowing to retrieve all instances generated in the context of a single workflow execution. from this simplified graph we can distinguish two main nodes http://moteur.processor/sorteo_singles and http://moteur.processor/sorteo_emissions which correspond to services with a large number of invocations.

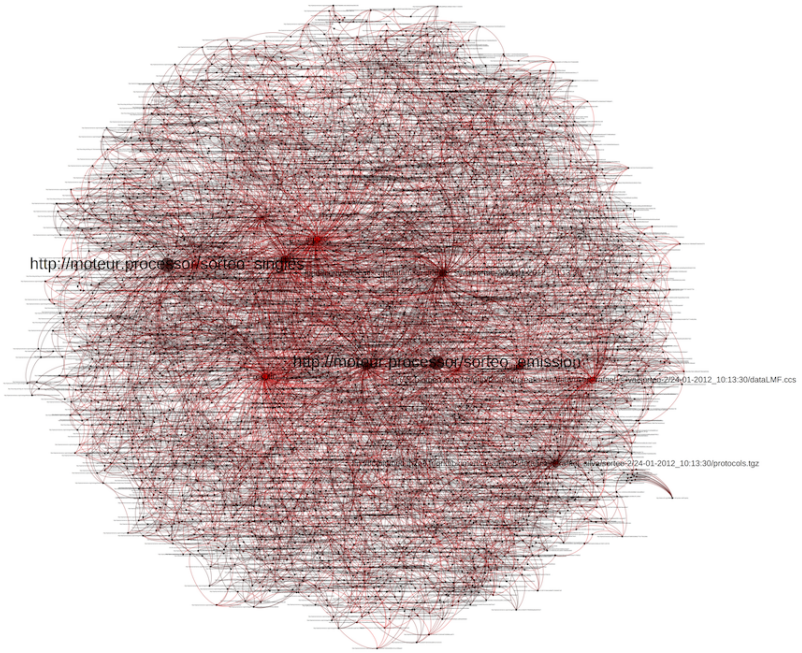

Figure 8.10: A filtered OPM provenance graph with removed *rdf:type* properties for the main OPM classes such as *Artifact*, *Used*, *WasGeneratedBy*, etc.

Due to its fine granularity and its size, the OPM model leads to complex graphs involving large amounts of generic and technical elements. Interpreting these OPM graphs is difficult. To address this issue, we segmented the produced semantic annotations through two distinct semantic repositories. First, a short-term repository, aiming at temporarily storing OPM statements, as the necessary input data to infer new meaningful statements. Second, a long-term repository, aiming at permanently storing the new statements resulting from inferences involving domain-specific entities provided by the VIP ontology.

Distributed provenance data. We propose in this experiment a slightly different setup, compared to the original VIP platform, in which we consider distributed data sources. We will compare in the results of this experiment the performance of the inference engine in both a centralized and a distributed setup.

The first semantic repository corresponds to the service registry, hosting the semantic service descriptions of the processors involved in the Sorteo simulation workflow. The

two other repositories store each half of the OPM provenance statements (7.5K statements each). Provenance statements have been split manually without any consideration on their nature. Each provenance repository is homogeneous in the sense that the same kind of OPM instances or properties may be present in the two repositories.

Each of the three semantic repositories are exposed through a SPARQL Endpoint deployed in an Apache Tomcat 6 server hosted in three nodes of the Suno cluster of the Grid'5000 infrastructure. An additional node is reserved to perform distributed querying.

Linked data reasoning and querying framework. The distributed querying of this experiment is performed through the KGRAM extension proposed in section 7.3 and benefits from the optimization strategies proposed in chapter 4. Thanks to the genericity of KGRAM, our implementation of a distributed *Producer* leverages the pre-existing query and forward-chaining inference engine, to finally perform distributed linked data reasoning and querying in a reasonable amount of time, over both provenance information and service descriptions.

Inferring meaningful statements from distributed provenance. To achieve objective, we propose the inference rule illustrated in Listing 8.11 through a CONSTRUCT SPARQL query. Its WHERE clause corresponds to the antecedent of the rule (an *"If"* condition) and its CONSTRUCT clause corresponds to the consequent of the rule (a *"Then"* consequence) where the new meaningful statements are created, involving classes and properties defined in the VIP ontology.

- *Lines 29 to 32* identify a process invocation, its corresponding service description through an *?agent* instance, and the achieved class of action through the *iec:refers-to* property. In this rule, the class of action is an image reconstruction.

- *Lines 33 to 34* identify the output *Artifact* (*?out*) through an instance of the *WasGeneratedBy* causal dependency (*?wgb*). This dependency is linked to the *Process* (*?x*) through an *opmo:cause* property, and to the output *Artifact* through an *opmo:effect* property. The value and content are associated to the *Artifact* through the *opmo:avalue* and the *opmo:content* properties (lines 66 to 67).

- *Lines 36 to 39* identify a process invocation realizing a parameters generation action, similarly to lines 9 to 13.

- *Lines 41 to 48* identify, the *Artifacts* (and their associated values) used as input of a process invocation realizing a parameters generation action. Additionally, the *?role* characterizing how the *Artifact* has been used by the process is identified. It allows to technically identify the parameters in the semantic service description associated to the process (line 47 and 48).

Listing 8.11: Inference rule based on a Sparql CONSTRUCT query to associate the input phantom to the produced output sinogram resulting from an invocation of the Sorteo simulation workflow.

```
PREFIX rdf: <http://www.w3.org/1999/02/22−rdf−syntax−ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf−schema#>
PREFIX opmo: <http://openprovenance.org/model/opmo#>
PREFIX opmv: <http://purl.org/net/opmv/ns#>
PREFIX ws: <http://www.irisa.fr/visages/team/farooq/ontologies/web−service−owl−lite.owl#>
PREFIX iec: <http://www.irisa.fr/visages/team/farooq/ontologies/iec−owl−lite.owl#>

PREFIX vip−model: <http://vip.cosinus.anr.fr/vip−model.owl#>
PREFIX vip−simulation: <http://vip.cosinus.anr.fr/vip−simulation.owl#>
PREFIX vip−simulated−data: <http://vip.cosinus.anr.fr/vip−simulated−data.owl#>

CONSTRUCT {
        ?inPhantom rdf:type vip−model:medical−image−simulation−object−model
        ?inPhantom vip−model:is−stored−in−file ?cInPhantom

        ?inProtocole rdf:type vip−simulation:simulation−parameter−set
        ?inProtocole vip−model:is−stored−in−file ?cInProtocole

        ?out vip−model:derives−from−model ?inPhantom
        ?out vip−simulation:derives−from−parameter−set ?inProtocole
        ?out rdf:type vip−simulated−data:PET−sinogram
        ?out vip−model:is−stored−in−file ?cOut
        ?out vip−simulation:is−a−result−of−at ?wf

        ?wf rdf:type vip−simulation:PET−simulation
        ?wf vip−simulation:uses−as−model−in−simulation ?inPhantom
        ?wf vip−simulation:uses−as−parameter−in−simulation ?inProtocole
} WHERE {
        ?agent (iec:refers−to/rdf:type) <http://vip.cosinus.anr.vip.fr/vip−simulation.owl#image−reconstruction−simulator−component> .
        ?wcb opmo:cause ?agent .
        ?wcb opmo:effect ?x .
        ?x rdf:type opmv:Process .
        ?wgb opmo:cause ?x .
        ?wgb opmo:effect ?out .

        ?agent2 (iec:refers−to/rdf:type) <http://vip.cosinus.anr.vip.fr/vip−simulation.owl#parameters−generation−simulator−component> .
        ?wcb2 opmo:cause ?agent2 .
        ?wcb2 opmo:effect ?y .
        ?y rdf:type opmv:Process .

        ?used1 opmo:cause ?inPhantom .
        ?used1 opmo:effect ?y .

        ?used2 opmo:cause ?inProtocole .
        ?used2 opmo:effect ?y .

        ?used1 opmo:role/rdfs:label ?techRolePhantom .
        ?used2 opmo:role/rdfs:label ?techRoleProtocole .

        ?agent2 ws:has−input ?inPortPhantom .
        ?inPortPhantom (iec:refers−to/rdf:type) <http://vip.cosinus.anr.fr/vip−model.owl#geometrical−phantom−object−model> .
        ?inPortPhantom rdfs:comment ?techRolePhantom .

        ?agent2 ws:has−input ?inPortProtocole .
        ?inPortProtocole (iec:refers−to/rdf:type) <http://vip.cosinus.anr.fr/vip−model.owl#quality−procedure−dataset> .
        ?inPortProtocole rdfs:comment ?techRoleProtocole .

        ?x opmo:account ?wf .

        ?inPhantom opmo:avalue ?vInPhantom .
        ?vInPhantom opmo:content ?cInPhantom .

        ?inProtocole opmo:avalue ?vInProtocole .
        ?vInProtocole opmo:content ?cInProtocole .

        ?out opmo:avalue ?vOut .
        ?vOut opmo:content ?cOut .
}
```

- *Lines 50 to 56* finally join the service description of (*?agent2*) to the process invocation (*?y*) through the label associated to the input port (*?role*), this input port referring to a geometrical phantom (lines 51 and 52).

To summarize, the elements involved in this experiment are the (i) the Sorteo simulation workflow, (ii) generic fine-grained RDF statements describing data provenance for a single workflow invocation, (iii) semantic service annotations, describing the main processors of the Sorteo workflow by using the VIP ontology, and (iv) an inference rule using both provenance and service description annotations to produce new meaningful statements thus providing to end-users a "semantic experiment summary" based on classes and properties of the VIP ontology.

8.4.2 Results and discussion

Semantic experiment summaries The main result of this experiment is a set of new meaningful statements inferred from the execution of a medical image simulation experiment. These new statements provide a high-level, and concise "semantic experiment summary". We consider the experiment summary as a high-level description since it only involves domain-specific classes and properties defined in the VIP ontology, compared to the generic and technical entities provided by the OPM provenance ontology. We also consider the experiment summary as concise since only 7 statements might be produced, compared to the 15 thousand statements produced through the Moteur OPM provenance plugin.

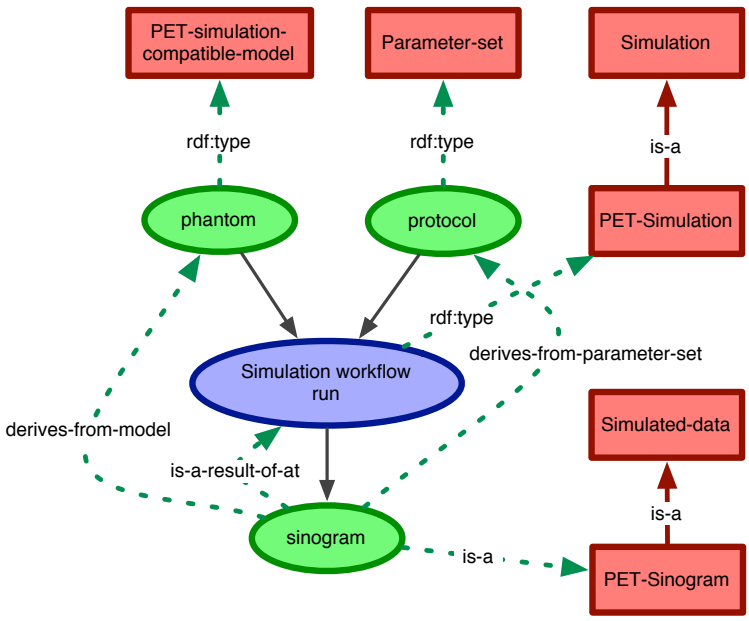

Figure 8.11: New inferred meaningful statements (dashed arrows) constituting the semantic experiment summary.

Figure 8.11 illustrates the experiment summary resulting from the invocation of the Sorteo medical imaging workflow. Green ellipses represent input or output data, the blue ellipse represents the Sorteo workflow shown as a "black box", and red rectangles represent VIP ontology classes. The inference rule presented in Listing 8.11 only automates the semantic annotation of the output sinogram and the corresponding input phantom, but it could easily be extended to also annotate the input protocol and the overall simulation execution as it is shown here in the experiment summary. Dashed green arrows represent the new inferred statements. For instance, the output sinogram is related to its corresponding input phantom through the *vip:has-for-phantom* property (Listing 8.11, line 6). The nature of the sinogram is also determined through the *is-a* property towards the VIP class *PET-Sinogram* (Listing 8.11, line 5).

Usability and quality. The approach considered in this experiment aims at enhancing the usability of data produced through e-Science workflows, and precisely medical imaging workflows in the context of the VIP platform. Both usability and quality are considered here. Workflow designers can exploit raw fine-grained OPM provenance information while designing and debugging workflows. But due to its size and genericity, it is not aimed at being directly exploited by scientists. Through the proposed "semantic experiment summaries" we aim at enhancing the confidence of scientists in the quality of their e-Science experiments by providing concise domain-specific annotations describing the produced data (nature and role) and coarse-grained relations between the data produced and the experiment parameters.

Performance of inferences over distributed data sources. The experiment was run on the Grid'5000 experimental infrastructure to compare a distributed and a centralized setup. Results from table 8.8 were obtained through 3 consecutive runs to avoid the variability observed in a real distributed infrastructure. Results show that although the performance is better when all semantic annotations are loaded into a single KGRAM engine, inferences performed against 3 distributed data sources are performed in less than 0.5 second, which is very acceptable in the context of real medical imaging workflows producing large amounts of provenance statements, possibly distributed.

	Distributed inferencing	*Centralized inferencing*
mean (ms) $\pm$ *std. deviation*	489 ± 2	185.3 ± 6.8

Table 8.8: Performance evaluation of inference

However, the design of inference rules is an important issue since a non-expert user may produce very ineffective inference rules, in spite of the optimization strategies proposed in chapter 4. For instance, if we come back to the inference rule (Listing 8.11) proposed, we could imagine that a query designer proposes a suboptimal rule by reordering the triple patterns. Listing 8.12 illustrates the reordered triple patterns aiming at identifying process invocations which realize an image reconstruction action.

Listing 8.12: Reordered triple patterns corresponding to lines 9 to 13 of the listing 8.11

```
1   ?x rdf:type opmv:Process .
2   ?wcb opmo:effect ?x .
3   ?wcb opmo:cause ?agent .
4   ?agent (iec:refers−to/rdf:type)
5       <http://vip.cosinus.anr.vip.fr/vip−simulation.owl#image−reconstruction−simulator−component> .
```

This order is ineffective since a lot of candidates are retrieved from the first line. Indeed, the KGRAM engine enumerates all process invocation without making any selection. The results space is reduced only at line 4 where candidate process invocations are joined through an agent responsible for a specific class of action (image reconstruction). The triple pattern order is even more important in a distributed scenario because all candidate results are communicated from the distributed data sources. We similarly reordered lines 9 to 13 and lines 19 to 23 and re-run the inference against the three data sources deployed on the Grid'5000 infrastructure. We measured 10.8 s $\pm$ 0.8 which is approximately 20 times longer than with the order proposed in the original inference rule. This can be explained because the service catalog only exposes a single service realizing an image reconstruction action, and the following triple pattern requests are successively joined with few intermediate results, thus leading to a short evaluation time.

Reusability of inference rules. Since the order of the triple patterns forming the inference rules has a huge impact on performance, their design is crucial and thus rules should be as much as possible reused. This is made possible by the loose coupling between inference rules and technical provenance information. Indeed service parameters are not identified by their technical label (which depends on the service implementation), but through their semantic annotations. In this way, as soon as two services, with two different implementations, are described with the same semantic annotations, the same inference rule can be applied for the two service invocations.

By designing inference rules exploiting the right abstraction level in a subsumption hierarchy, we enhance the rule reusability. Figure 8.12 illustrates an updated version of the Sorteo workflow where the last process, *Lmf2RawSino_v3*, in purple, is an updated version of the original *Lmf2RawSino*. We consider that its implementation is completely different, technical parameters may have changed, but the functionality is still the same. Since the semantic description of *Lmf2RawSino_v3* is subsumed[15] by the semantic description of *Lmf2RawSino*, and the inference rule involves semantic description of *Lmf2RawSino*, the same inference rule can be applied to also annotate the results of *Lmf2RawSino_v3* and thus, to similarly produce "semantic experiment summaries" for the updated version of the Sorteo workflow.

To conclude on reusability, a same inference rule can be reused for several service implementations realizing the same objectives. However, specific inference rules may be

[15] we consider that the whole semantic description of a service S_{child} is subsumed by its parent service S_{parent} if the functionality and all parameter annotations of S_{child} are subsumed by S_{parent}.

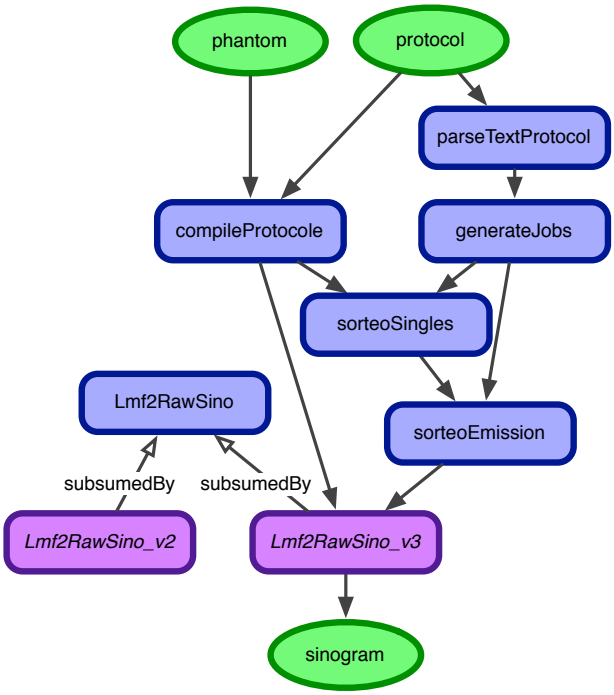

Figure 8.12: Updated Sorteo workflow involving a refined Lmf2RawSino service.

created for each simulation workflow because the expected "semantic experiment summary" are supposed to differ. But template rules could be proposed to rule designers, based on the original rule (Listing 8.11). Indeed, since workflow input/outputs are identified in the same way for all simulation workflows, rule designers could only "tune" the template rule by informing the adequate nature/role of data and the specific functionality of processes based on the VIP ontology.

Scalability. Since technical fine-grained OPM provenance information is useful at workflow design-time and workflow debug-time, it is temporarily stored in a short-term semantic repository. To perform inferences, only the provenance information related to a single invocation, and the service descriptions are needed. We finally store in a long-term repository, the few meaningful statements composing the "experiment summary". In this experiment, only 7 statements form the experiment summary are inferred when more than 15 thousand statements are recorded through the Moteur OPM provenance plugin. As an illustration, Figure 8.13 represents the content of the VIP long-term repository storing the meaningful experiment summaries. 118 medical image simulations are summarized with 2656 RDF triples based on the VIP domain ontology. This graphical representation has to be compared with Figure 8.9 and Figure 8.10 representing the fine-grained technical provenance information for a single simulation.

The scalability is finally guaranteed by (i) the materialization of very few inferred statements, and (ii) the periodic removal of fine-grained technical OPM provenance

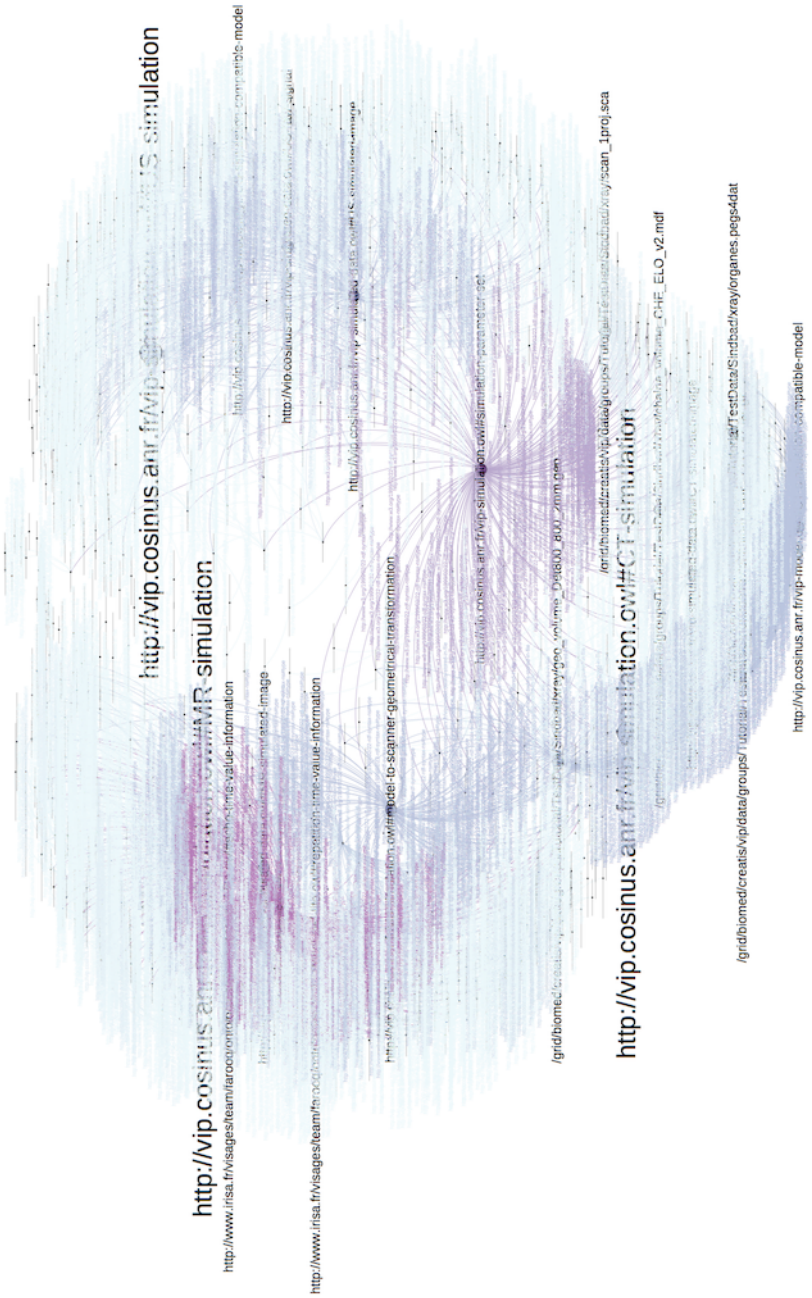

Figure 8.13: Graphical representation of 118 medical image simulations (launched during a single week). Only 2656 triples summarize 118 medical image simulations while more than 15000 provenance triples were stored for a single simulation (see Figure 8.9).

information.

The results of this medical image simulation experiment are summarized trough the following facets:

- *Performance: inference times in a same order of magnitude for both the centralized and the distributed scenarios ;*
- *Modeling: distinction between nature and role of service parameters in semantic service annotations ;*
- *Scalability: materialization of few meaningful statements, and "clean up" of verbose fine-grained technical provenance ;*
- *Versatility: no data-warehousing to infer over distributed provenance and service registries ;*
- *Usability: experiment summaries and reusable inference rules.*

8.5 Conclusion

This chapter proposes three experimental studies to assess the benefits of federated approaches to address distributed knowledge sharing and production, in the context of scientific workflows.

The performance and the scalability of the federated querying strategies (see Chapter 4) are evaluated in Section 8.2. The scalability has been evaluated through a real distributed setup provided by the Grid'5000 experimental and controlled infrastructure. The overall performance has been evaluated through the FedBench benchmark, also on Grid'5000, and showed results in line with the state of the art approaches while still benefitting from the high expressivity of the SPARQL 1.1 language.

The federated querying strategies have been evaluated in Section 8.3, in the context of the NeuroLOG collaborative neuroimaging platform. This application showed the suitability of semantic data federations, compared to traditional relational federations. Moreover the underlying abstract knowledge graphs help in mediating structural heterogeneity, thus enabling hybrid federations, typically involving both RDF triple stores and relational databases, while still providing SPARQL as single querying and reasoning language. Finally, moving the NeuroLOG platform towards semantic data federation opens new sharing perspectives through the Web of Linked Data, and makes a step further towards translational research.

Finally we evaluate, through a medical imaging simulation workflow deployed in production in the VIP platform, the suitability of our approach towards semantic scientific workflows developed in chapter 6 (RQ_4) to systematically produce domain knowledge from their executions (RQ_5). Provenance data is captured from a distributed workflow execution, and semantically queried through the distributed evaluation strategies developed in chapter 4 ($RQ_{1,3}$) to finally infer new meaningful statements constituting valuable semantic experiment summaries (RQ_5). This experiment showed that in the context of this real-life scientific workflow, we are able to efficiently track, in a distributed setup, both fine-grained domain-agnostic provenance and coarse-grain domain-specific provenance (meaningful experiment summaries), thus easing both the quality and the usability of scientific workflows and their massive data production.

Through these three experiments, we encompass the main contributions of the thesis towards distributed knowledge sharing and production in e-Science collaborative platforms (i) by providing, as much as possible, high-performance and scalable federated querying over multi-source distributed knowledge graphs while still offering to query designers the high expressivity of SPARQL 1.1 ; (ii) by opening an existing neuroimaging federations to the Web of Linked Data, thus creating new data sharing perspectives ; (iii) by leveraging domain ontologies at workflow runtime to enhance the quality of distributed, large-scale "in silico" experiments, and to ease the exploitation of results through meaningful experiment summaries.

Key Points

- *The performance and the scalability is evaluated on large open linked datasets, in a real distributed and controlled infrastructure (Grid'5000).*
- *The versatility of our approach enables knowledge sharing while still coping with the autonomy property of collaborative data providers, in the context of the NeuroLOG platform.*
- *The usability of scientific workflows and their data production is enhanced through both fine-grained domain-agnostic provenance and coarse-grained domain-specific provenance, the latter constituting valuable meaningful experiment summaries.*

Part IV

Conclusions

CHAPTER 9

Conclusion and perspectives

Contents

9.1 Contributions summary

In this dissertation, we addressed translational life-science data management and processing issues in the context of massive data, distributed at large-scale (*"Data Deluge"*). Our approach defends, in the context of digital collaborative Life-science platforms, the use of knowledge engineering methods in distributed data management systems. This multi-disciplinary approach aims at improving usage, in terms of search, analysis, processing of relevant multi-source data, and at accelerating data interpretation, hopefully leading to scientific outcomes. Our approach defends the following thesis: (T_1) data sources federation enhance the adoption and the scalability of e-Science collaborative platforms, (T_2) knowledge-enabled e-Science platforms reduces the complexity of experiment setups and ease the exploitation/interpretation of produced data, and (T_3) domain modeling (through Knowledge Engineering and Semantic Web technologies) is crucial for knowledge sharing and capitalization in life-science.

Knowledge-based life-science resources federation

This manuscript defends the idea that providing ontology-driven resources federation (versus centralized warehousing) is crucial to improve the scalability of e-Science platforms, and to ease the setup of coherent multi-centric collaborations (T_1, T_3). We showed in chapter 2 that (i) life-scientists require rich information sharing while still coping with the autonomy of data providers, (ii) that semantic distributed querying techniques are

still under active investigation and often suffer from a lack of expressivity, and (iii) that significant life-science data integration projects have targeted ontology-driven data sharing but did not clearly succeed in exploiting ontologies when performing distributed querying. Based on these observations, we proposed a distributed security model for life-science collaborations in chapter 3 that considers the autonomy of life-science data sources as a prime-order constraint. While still being applied to raw neuroimages, these strategies could be extended to multi-sources semantic data opening interesting perspective in the context of "closed" knowledge graphs. We also tackled in chapter 4 how semantic distributed querying can be (i) versatile, through scattered abstract knowledge graphs, and without prior knowledge on data sources content, (ii) performant, through a set of optimization for transparent federated querying, and (iii) expressive, through a strong compatibility with the SPARQL 1.1 language. Our approach has finally been experimentally evaluated at large-scale on concrete scenarios in chapter 8.

Knowledge-based scientific worflows

This manuscript also defends the idea that providing semantically instrumented scientific workflows helps in conducting large-scale data analysis procedures (T_2, T_3). Indeed, we showed along this dissertation that attaching a coherent meaning to both data processing tools, and processed data allows e-Scientists to more precisely search/combine/share/re-purpose data analysis procedures or processed data. We explored first, in chapter 5, the state-of-the-art approaches for semantically annotating web services, which provide a foundational descriptive layer towards semantic scientific workflows. Then, on the other way around, we explored provenance-based approaches for representing workflows runs, with a data-centric perspective. We showed that few approaches are directed towards better interpreting (by exploiting domain ontologies) data processed or analyzed through workflows. Then, we showed in chapter 6 that (i) the notion of domain-specific "Roles" was crucial to disambiguate the semantic annotation of service parameters, and (ii) that these "Roles", coupled with domain-agnostic provenance information assembled from workflow runs, were needed to infer domain-specific provenance information, more easily interpretable from an e-Scientist perspective. We finally experimented our approach in chapter 8 through a concrete medical imaging workflow, deployed in production in the VIP platform.

Software deployment in production platforms, dissemination

The research contributions proposed through chapters 4 and 6 led to the NeuSemStore and KGRAM-DQP softwares, detailed in chapter 7. KGRAM-DQP will soon be part of the Corese/KGRAM Semantic Web factory. NeuSemStore has already been deployed in both NeuroLOG and VIP platforms. Semantic experiment summaries are inferred in VIP on a daily basis with currently 18 production rules. One week after its deployment in VIP, NeuSemStore inferred and stored 118 simulation experiment summaries represented with 2656 RDF triples (roughly 22 triples per experiment). It has to be compared, for

instance, with a single SORTEO simulation leading to 1429 generic provenance triples. Regarding dissemination, NeuSemStore has also be used in the Amsterdam Academic Medical Center to compare on-the-fly provenance tracking with post-mortem workflow log analysis.

9.2 Future directions

9.2.1 Towards high performance semantic distributed querying

In spite of its limitations with respect to triple pattern grouping, the distributed query processing features we integrated into the KGRAM semantic web framework opens exciting perspectives.

From the performance experiments reported in section 8.2.1, we can see that the querying time decreases importantly when the full DBPedia person dataset is fragmented. Unsurprisingly, when the fragmentation increases, the gain is slowed down until the network overhead starts being penalizing.

With democratized cloud infrastructure in mind, we could conceive a system that would provide, based on self-adaptation methods (control loops) and dedicated cluster infrastructures, an elastic semantic data repository dedicated to the SPARQL querying of massive knowledge graphs. This is particularly relevant since the SPARQL 1.1 language now propose Update queries allowing for triple writing. The system would then monitor (i) the size of the global dataset, and (ii) its response time based on a set of predefined queries. The system would then be able to optimize the ratio between the global size of the dataset and the number of computing nodes hosting the graph by allocating new nodes and re-fragmenting the global dataset over these nodes.

A similar principle could be envisaged in the context of modern multi-core computers. We could indeed imagine better querying performances, since the communication overhead between different cores is incomparably lower than network overhead. A huge dataset, if loadable in the central memory, could be queried by fragments, in parallel, by several CPU cores. Through these highly parallel machines, similar "elasticity" could be developed to tune the performance of the semantic querier on-demand. KGRAM, as semantic web framework implemented in Java, could benefit from the novelties of Java version 7 which eases, by source code annotation, the parallelization of computing tasks on multi-core computers.

9.2.2 Towards highly expressive semantic distributed querying

Due to the availability[1] of significant and largely adopted life-science domain ontologies such as FMA (general anatomy), RadLex (radiology), SNOMED CT (clinical terms), or NIFSTD (neuroscience), e-Scientists have become more familiar with semantic querying.

[1]The BioPortal (http://bioportal.bioontology.org) is catalog of more than 300 biomedical ontologies defining more than 5 Million terms.

In addition, biomedical formal ontologies appeared as a need to provide logically consistent biomedical concepts. OBO foundry [Smith et al., 2007] (grounded the BFO formal ontology), BioTOP [Beisswanger et al., 2008] (providing bridges with BFO or the DOLCE formal ontology), or OntoVIP [Gibaud et al., 2012] (grounded to DOLCE), are initiatives aimed at providing biomedical conceptualization with strong logical consistency. In this context, e-Scientists want to benefit from the underlying modeling efforts and are thus demanding for expressive querying and reasoning capabilities.

Coping with SPARQL complexity

We chose in this thesis to focus on the query expressiveness. We proposed a federation engine which handles most of the SPARQL 1.1 recommendations in a distributed environment, with optimized evaluation. However, due to the complexity of SPARQL [Pérez et al., 2009] [Schmidt et al., 2010], queries more expressive than conjunctive queries (i.e. involving aggregate functions (counting, values comparisons), disjunctions, optionals, or property paths) may lead to performance issues. It would then be necessary to precisely study these works on complexity, to determine which SPARQL language feature lead to hardly tractable querying (such as OPTIONAL statements [Aranda et al., 2011], or NEGATION which would lead to a complete exploration of all available data sources). As a continuation of our work, it would be interesting (i) to focus on how expressive SPARQL queries behave in a distributed setup and (ii) propose new heuristics to enhance their evaluation in a distributed environment.

Taking into account that most of the currents works towards SPARQL federated querying focus on basic graph patterns (conjunctive queries), the extension of existing benchmarks such as FedBench with these expressive queries would certainly have a positive effect, hopefully leading to more research addressing the expressivity of distributed SPARQL querying.

Distributed Semantic Web reasoning

In our work, we proposed distributed query processing over scattered knowledge graphs, as an extension of the KGRAM Semantic Web framework. RDFS entailment regime is provided in the core of the KGRAM federation engine. Thus, reasoning is performed centrally based on distributed and scattered knowledge graphs. Towards efficient reasoning with KGRAM, three interesting perspectives could be envisaged.

First, since KGRAM can be deployed as a SPARQL endpoint, it is possible to perform RDFS entailments remotely (such as subsumption or transitivity) on each fragment of the distributed knowledge graph. This would allow to distribute both the querying and reasoning cost. However it raises the issue of the locality of the ontology (TBox), or the data (ABox). Indeed, in the context of RDF/RDFS, we could imagine two typical use cases. A first use case would consist in some data sources hosting only RDF data while a single source hosts the RDFS schema. In this context, ABox reasoning would only be possible on the source hosting the RDFS schema. Providing distributed ABox reasoning would also

necessitate to coordinate multiple data sources. The second use case would consist in a modular ontology in which some parts are distributed over the data sources. Similarly, providing distributed TBox reasoning would also necessitate coordination between data sources. A first pragmatic approach would consist in an hybrid centralized/distributed strategy based on (i) performing all possible entailments locally to the data source, then (ii) coordinating data sources with transfers of the ABox or the TBox so that missing entailments become possible. In this direction of distributed ABox/TBox reasoning, the "peer to peer" and "distributed databases" research communities led to significant results [Abdallah et al., 2009], [Abiteboul et al., 2012], but still, significant research efforts should be directed towards highly expressive distributed SPARQL reasoning.

Then, KGRAM being also provided with a forward and backward chaining inference engine, an interesting research direction for our first results in handling distributed knowledge bases would consist in studying planning strategies to find the optimal inference rules execution order. These planning strategies could be based on basic statistics [Langegger and Wöß, 2009], [Alexander et al., 2009] or advanced data source indices giving information on frequent graph structures [Basse et al., 2010].

Finally distributed reasoning is challenging in terms of inferred statement management. When considering large amount of distributed semantic data, the transitive closure of the global graph may be huge. It raises thus the still-open issue of distributing the inferred statements over the participating sources. Should the distribution of inferred statements be content-agnostic or content-specific ? In other words, would it be interesting to localize the inferred statements close to the originating data/ontology source ? What is the impact in terms of access control and autonomy when the originating data sources are protected ? Tracking the provenance of inferred statements would certainly be a first step to envisage further studies on these challenging issues.

9.2.3 Towards versatile and reliable knowledge-based data federations

This thesis has been strongly motivated by the autonomy constraints of life-science data providers. Autonomy requirements are partly contradictory with sharing initiatives. However, our contributions established compromises tending to ease the setup of multicentric studies, the sharing and repurposing of data consumed or produced in e-Science *"in silico"* experiments. These first results raise important points that would require substantial research efforts.

Versatile knowledge-based data federations

It is not realistic to envisage life-science data providers, who exploited legacy databases for years, rapidly adopting Semantic Web technologies as their foundational IT framework. Although Semantic Web technologies provide means for high expressivity, meaningful and coherent data integration, eventually logical reasoning, openness to other data sharing initiatives in the context of the Linked Data Cloud, they are still technologically less mature than relational data models, established at an industrial scale.

The proposed distributed querying strategies, based on abstract knowledge graphs open interesting perspectives in terms of versatility. We proposed in our work a simple ad-hoc KGRAM producer (see figure 8.6 of section 8.3) dedicated to the mediation of SQL data. Given a set of predefined mappings between RDF predicates and SQL queries, this KGRAM SQL data producer provides distributed SPARQL querying over both semantic and relational data sources through dynamic query rewriting.

Although this approach is appealing because of its dynamicity, the mediation capability of the proposed SQL KGRAM producer is still limited due to predefined set of mappings, yet hardly extensible. Recently, significant research activities have been conducted to map relational data to RDF data and should be precisely analyzed. These activities led to the R2RML W3C recommendation [Das et al., 2012], which defines a standard language to express customized mappings from relational databases to RDF datasets. Since most of R2RML engines provide a static extraction of SQL data to build an RDF dataset, this work opens an interesting perspective of dynamically mediating SQL data through SPARQL and a KGRAM producer that would be built based on an SQL database endpoint and an R2RML document specifying the mappings. The provision of such structural mediation would be a first step towards longer-term research activities addressing advanced ontology-based querying and reasoning over legacy relational databases.

Foundational (formal) ontologies (Dolce/BFO/OBO/BioTop) are gaining a lot of interest to address semantic heterogeneity often faced when dealing with data integration activities. Semantic heterogeneity is due to several data sources publishing data expressed through multiple, distinct and possibly overlapping ontologies. Even if a lot of efforts are dedicated to the sharing and reusing of domain ontologies, we will probably face the multiplicity and diversity of ontologies for describing a same kind of data. Towards versatile semantic querying, we could also consider KGRAM Producers as semantic aligner components to address semantic heterogeneity. As a continuation of this work, we could envisage to study in a first step, how SPARQL querying would behave when querying for instance two data sources exposing data through two distinct ontologies. In a longer-term perspective, we could study how RDFS entailments could be operationalized in this context.

Reliable knowledge-based data federations

Data source fault-tolerance. Taking into account the autonomy constraint of life-science data providers, for any reason, technical (electrical outages, modification of site network security policies, etc.) or operational (data source managers may decide to leave a data federation), the data sources availability is not guaranteed. Consequently, distributed joins over multi-source data may fail when one of the data sources is unavailable. This raises challenging issues which have not been studied much in the context of Linked Open Data. Recent research works such as RDFSync [Tummarello et al., 2007] or Live Linked Data [Ibáñez et al., 2012], aiming at synchronizing semantic data stores, should be considered to address the reliability of federated data sources. However in a context of hardly relocatable biomedical data, these approaches would still face the

autonomy challenges, in terms of access control. Instrumenting such synchronization techniques with provenance information and cryptography would probably help data providers in keeping the control over the data they operate.

Generalized provenance tracking. Providing fault-tolerant data federation is crucial to limit the impact of unavailable data sources. However, it only covers a narrow spectrum of reliability. Quality and confidence on life-science resources (data or analysis procedures) has become a major criterion of reliability in e-Science. We proposed in chapter 6 a provenance-based approach providing both (i) fine-grained technical information helping "*in silico*" experiment designers in determining the cause of workflow failures, or abnormal/unexpected results, and (ii) coarse-grained semantic annotations providing ontology-based experiment summaries, helping e-Scientists in comprehending the produced/analyzed data. Provenance has been considered in this work as tightly coupled to scientific workflows. The reliability of knowledge-based data federations could considerably be enhanced by extending the usage of provenance in the context of distributed querying.

Provenance information could be assembled and attached to SPARQL results, finally informing end users on the data sources the results originate from. This information could be used to provide explanations of federated querying failures. Explaining semantic querying is currently under investigation in our laboratory [Hasan and Gandon, 2012]. It is particularly relevant in the context of transparent federated querying as proposed in chapter 4 and would open interesting applications in the context of query explanation. Closer to e-Scientists, generalizing provenance information to initial research/clinical databases, or linked open data stores (version, location, curation status, etc.) is particularly relevant since it would enhance the confidence on consumed/produced/analyzed data, finally enhancing the overall reliability of e-Science platforms.

Based on the proposed semantic experiment summaries, an interesting future direction would consist in providing bridges between "experiment provenance" and "data source provenance" so that e-Scientists navigating their semantic experiment summaries could be informed about the data sources that contributed to their processed/analyzed data. This could be a first step towards data rewarding, and motivate data holders for considering sustainable data provision.

9.2.4 Towards reduced information overload in e-Science

E-science platforms bring together a large variety of data, processing tools, assembled into scientific workflows, and knowledge captured through domain ontologies. Because of several intertwined granularities and representation layers, and several end-user activities, it is difficult to provide the relevant information to each user. One of our contribution consists in better exploiting processed data through semantic experiment summaries. Another related research activity is conducted in our laboratory and aims at better exploiting e-science workflows through their associated domain knowledge [Cerezo and Montagnat, 2011].

This work addresses the design of scientific workflows through a goal-based approach. The objective is to generate workflow descriptions from conceptual descriptions based on high-level *fragments*. These *Fragments* are composed by a *Pattern* and a *Blueprint* responsible for injecting the target workflow activities. Conceptual workflows are derived into concrete workflows by weaving blueprints.

This work on conceptual workflows and our contribution towards meaningful experiment summaries are complementary. Indeed, conceptual workflows could help in the design of provenance-based inference rules and enhance their genericity. Conversely, provenance information could be used to suggest annotations at conceptual workflow design-time based on produced and annotated data.

Key Perspectives

- **Efficiency:** *triple pattern grouping and query planning for scattered abstract knowledge graphs ; scalable SPARQL endpoint for massive knowledge graphs ;*
- **Versatility:** *R2RML-based dynamic mediation of SQL databases through abstract knowledge graphs ; dynamic semantic alignment through abstract knowledge graphs ;*
- **Reliability:** *fault-tolerant semantic data sources ; generalized provenance, from processed data to originating data sources ;*
- **Expressiveness:** *extension of the FedBench benchmark with more expressive queries ; distributed Semantic Web reasoning (inference planning, materialization of inferred statements) ; conceptual workflows to ease the design of provenance-based inference rules ;*

9.3 Concluding remarks

We proposed in this thesis some methods and techniques dedicated to the practice of digital sciences in the *"Data Deluge"* and directed towards a rationalized practice of Science.

With regards to a rationalized practice of Science we hope that efficient and semantically accurate distributed systems will foster the coherent publication, on the Web, of valuable data and tools, finally opening new sharing and repurposing opportunities across permeable communities. This still raises the question of the reluctance of data owners to coherently publish their data, and the question of the sustainability of data publication to enhance reproducibility and scientific value. We also hope that from *"Big Data"* (usually exploited through statistical approaches), research on IT systems will soon be able to provide valuable "data diets" which would mean few, meaningful and relevant data at the right time for the right person.

Knowledge associated to searched/combined/analyzed data has been central in this work. We consider knowledge engineering as a discipline which moves machines closer to humans. We hope to contribute in our future works to a virtuous circle in which (i) knowledge would drive collaborative science, (ii) knowledge would be produced and published more rapidely from the usage of e-Science platforms, and (iii) the resulting scientific outcomes would more rapidly be translated to the society (*e.g.* patients in the context of translational medicine). Semantic Web technologies and standards, combining knowledge engineering and distributed systems, are currently a major opportunity to work in this direction of knowledge-based translational sciences.

Part V

Appendix

CHAPTER 10

Appendix

10.1 FedBench Life Science Queries

Listing 10.1: LS1 FedBench query

```
SELECT $drug $melt WHERE {
    { $drug <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/meltingPoint> $melt. }
    UNION
    { $drug <http://dbpedia.org/ontology/Drug/meltingPoint> $melt . }
}
```

Listing 10.2: LS2 FedBench query

```
SELECT ?predicate ?object WHERE {
    { <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugs/DB00201> ?predicate ?object . }
    UNION
    { <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugs/DB00201>
                                        <http://www.w3.org/2002/07/owl#sameAs> ?caff .
      ?caff ?predicate ?object . }
}
```

Listing 10.3: LS3 FedBench query

```
SELECT ?Drug ?IntDrug ?IntEffect WHERE {
    ?Drug <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dbpedia.org/ontology/Drug> .
    ?y <http://www.w3.org/2002/07/owl#sameAs> ?Drug .
    ?Int <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/interactionDrug1> ?y .
    ?Int <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/interactionDrug2> ?IntDrug .
    ?Int <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/text> ?IntEffect .
}
```

Listing 10.4: LS4 FedBench query

```
SELECT ?drugDesc ?cpd ?equation WHERE {
    ?drug <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/drugCategory>
                    <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugcategory/cathartics> .
    ?drug <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/keggCompoundId> ?cpd .
    ?drug <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/description> ?drugDesc .
    ?enzyme <http://bio2rdf.org/ns/kegg#xSubstrate> ?cpd .
    ?enzyme <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://bio2rdf.org/ns/kegg#Enzyme> .
    ?reaction <http://bio2rdf.org/ns/kegg#xEnzyme> ?enzyme .
    ?reaction <http://bio2rdf.org/ns/kegg#equation> ?equation .
}
```

Listing 10.5: LS5 FedBench query

```
SELECT $drug $keggUrl $chebiImage WHERE {
  $drug <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
                        <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/drugs> .
  $drug <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/keggCompoundId> $keggDrug .
  $keggDrug <http://bio2rdf.org/ns/bio2rdf#url> $keggUrl .
  $drug <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/genericName> $drugBankName .
  $chebiDrug <http://purl.org/dc/elements/1.1/title> $drugBankName .
  $chebiDrug <http://bio2rdf.org/ns/bio2rdf#image> $chebiImage .
}
```

Listing 10.6: LS6 FedBench query

```
SELECT ?drug ?title WHERE {
        ?drug <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/drugCategory>
                        <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugcategory/micronutrient> .
        ?drug <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/casRegistryNumber> ?id .
        ?keggDrug <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
                                        <http://bio2rdf.org/ns/kegg#Drug> .
        ?keggDrug <http://bio2rdf.org/ns/bio2rdf#xRef> ?id .
        ?keggDrug <http://purl.org/dc/elements/1.1/title> ?title .
}
```

Listing 10.7: LS7 FedBench query

```
SELECT $drug $transform $mass WHERE {
        { $drug <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/affectedOrganism>
                                                                    'Humans and other mammals'.
          $drug <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/casRegistryNumber> $cas .
          $keggDrug <http://bio2rdf.org/ns/bio2rdf#xRef> $cas .
          $keggDrug <http://bio2rdf.org/ns/bio2rdf#mass> $mass
              FILTER ( $mass > '5' )
        }
        OPTIONAL {$drug <http://www4.wiwiss.fu-berlin.de/drugbank/resource/drugbank/biotransformation>
            $transform .}
}
```

Bibliography

Shibboleth. URL http://shibboleth.internet2.edu/. (Cited on page 59.)

JRA3: Global Security Architecture, rev. 1. EGEE Project deliverable, 2005. (Cited on page 60.)

N. Abdallah, F. Goasdoué, and M.-C. Rousset. Dl-liter in the light of propositional logic for decentralized data management. In *Proceedings of the 21st international jont conference on Artifical intelligence*, IJCAI'09, pages 2010–2015, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc. URL http://dl.acm.org/citation.cfm?id=1661445.1661766. (Cited on page 201.)

S. Abiteboul, É. Antoine, and J. Stoyanovich. Viewing the Web as a Distributed Knowledge Base. In *International Conference on Data Engineering*, Washington DC, États-Unis, 2012. IEEE. URL http://hal.inria.fr/hal-00703210. (Cited on page 201.)

J. Abraham, P. Brazier, A. Chebotko, J. Navarro, and A. Piazza. Distributed storage and querying techniques for a semantic web of scientific workflow provenance. In *Proceedings of the 2010 IEEE International Conference on Services Computing*, SCC '10, pages 178–185, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4126-6. doi: http://dx.doi.org/10.1109/SCC.2010.14. URL http://dx.doi.org/10.1109/SCC.2010.14. (Cited on page 118.)

R. L. Ackoff. From data to wisdom. *Journal of Applied Systems Analysis*, 16, 1989. (Cited on page 19.)

K. Alexander, R. Cyganiak, M. Hausenblas, and J. Zhao. Describing Linked Datasets - On the Design and Usage of voiD, the 'Vocabulary of Interlinked Datasets'. In *WWW 2009 Workshop: Linked Data on the Web (LDOW2009)*, Madrid, Spain, 2009. (Cited on pages 41, 43 and 201.)

R. Alfieri, R. Cecchini, V. Ciaschini, L. dell'Agnello, Á. Frohner, A. Gianoli, K. Lörentey, and F. Spataro. Voms, an authorization system for virtual organizations. In *European Across Grids Conference*, pages 33–40, 2003. (Cited on page 59.)

I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludascher, and S. Mock. Kepler: an extensible system for design and execution of scientific workflows. In *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*, pages 423 – 424, june 2004. doi: 10.1109/SSDM.2004.1311241. (Cited on pages 116 and 117.)

A. Anjum, N. Bessis, R. Hill, R. McClatchey, I. Habib, K. Soomro, P. Bloodsworth, and A. Branson. Provenance management for neuroimaging workflows in neugrid. In *P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2011 International Conference on*, pages 67 –74, oct. 2011. doi: 10.1109/3PGCIC.2011.20. (Cited on page 117.)

A. Ankolekar, D. Martin, D. McGuinness, S. McIlraith, M. Paolucci, and B. Parsia. OWL-S' Relationship to Selected Other Technologies [http://www.w3.org/submission/owl-s-related], November 2004. URL http://www.w3.org/Submission/OWL-S-related/. (Cited on page 108.)

M. Antonioletti, N. P. C. Hong, A. C. Hume, M. Jackson, K. Karasavvas, A. Krause, J. M. Schopf, M. P. Atkinson, B. Dobrzelecki, M. Illingworth, N. McDonnell, M. Parsons, and E. Theocharopoulous. Ogsa-dai 3.0 - the what's and whys. In *UK e-Science All Hands Meeting*, 2007. (Cited on pages 49 and 50.)

C. B. Aranda, M. Arenas, and Ó. Corcho. Semantics and optimization of the sparql 1.1 federation extension. In *ESWC (2)*, pages 1–15, 2011. (Cited on pages 46 and 200.)

A. Arbona. Fura, a self-contained grid middleware. 2009. URL http://fura.sourceforge.net. (Cited on page 50.)

N. Ashish, J. Ambite, M. Muslea, and J. Turner. Neuroscience data integration through mediation: An (f)birn case study. *Frontiers in Neuroinformatics*, 4:12, 2010. URL http://www.frontiersin.org/Journal/Abstract.aspx?s=752&name=neuroinformatics&ART_DOI=10.3389/fninf.2010.00118. (Cited on pages 49 and 53.)

F. Baader and W. Nutt. Basic description logics. In Baader et al. [2003], pages 43–95. ISBN 0-521-78176-0. (Cited on page 31.)

F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*, 2003. Cambridge University Press. ISBN 0-521-78176-0. (Cited on pages 214 and 229.)

R. G. Baraniuk. More Is Less: Signal Processing and the Data Deluge. *Science*, 331(6018): 717–719, Feb. 2011. ISSN 1095-9203. doi: 10.1126/science.1197448. URL http://dx.doi.org/10.1126/science.1197448. (Cited on page 2.)

A. Basse, F. Gandon, I. Mirbel, and M. Lo. Dfs-based frequent graph pattern extraction to characterize the content of rdf triple stores. 2010. URL http://journal.webscience.org/356/1/websci1_submission_57.pdf. (Cited on pages 99, 100 and 201.)

M. Bastian, S. Heymann, and M. Jacomy. Gephi: An open source software for exploring and manipulating networks, 2009. URL http://www.aaai.org/ocs/index.php/ICWSM/09/paper/view/154. (Cited on page 182.)

S. Battle, A. Bernstein, H. Boley, B. Grosof, M. Gruninger, R. Hull, M. Kifer, D. Martin, S. McIlraith, D. McGuinness, J. Su, and S. Tabet. Semantic Web Services Ontology (SWSO) [http://www.w3.org/submission/swsf-swso], September 2005. URL http://www.w3.org/Submission/SWSF-SWSO. (Cited on page 130.)

E. Beisswanger, S. Schulz, H. Stenzhorn, and U. Hahn. Biotop: An upper domain ontology for the life sciences: A description of its current structure, contents and interfaces to obo ontologies. *Appl. Ontol.*, 3(4):205–212, 2008. ISSN 1570-5838. URL http://dl.acm.org/citation.cfm?id=1516154.1516159. (Cited on page 200.)

K. Belhajjame, S. M. Embury, N. W. Paton, R. Stevens, and C. A. Goble. Automatic annotation of web services based on workflow definitions. *TWEB*, 2(2), 2008a. (Cited on page 113.)

K. Belhajjame, K. Wolstencroft, O. Corcho, T. Oinn, F. Tanoh, A. William, and C. A. Goble. Metadata management in the taverna workflow system. In *International Conference on Computational Science (4)*. IEEE Computer Society, 2008b. (Cited on page 113.)

G. Bell, T. Hey, and A. Szalay. Beyond the Data Deluge. *Science*, 323(5919):1297–1298, Mar. 2009. ISSN 1095-9203. doi: 10.1126/science.1170411. URL http://dx.doi.org/10.1126/science.1170411. (Cited on page 2.)

A. Benabdelkader, M. Santcroos, S. Madougou, A. H. C. van Kampen, and S. D. Olabarriaga. A provenance approach to trace scientific experiments on a grid infrastructure. In *eScience*, pages 134–141, 2011. (Cited on page 154.)

S. Benkner, A. Arbona, G. Berti, A. Chiarini, R. Dunlop, G. Engelbrecht, A. Frangi, C. Friedrich, S. Hanser, P. Hasselmeyer, R. Hose, J. Iavindrasana, M. Kö andhler, L. Iacono, G. Lonsdale, R. Meyer, B. Moore, H. Rajasekaran, P. Summers, A. Wö andhrer, and S. Wood. @neurist: Infrastructure for advanced disease management through integration of heterogeneous data, computing, and complex processing services. *Information Technology in Biomedicine, IEEE Transactions on*, 14(6):1365 –1377, nov. 2010. ISSN 1089-7771. doi: 10.1109/TITB.2010.2049268. (Cited on page 50.)

T. Berners-Lee, R. Cailliau, A. Luotonen, H. F. Nielsen, and A. Secret. The world-wide web. *Commun. ACM*, 37(8):76–82, Aug. 1994. ISSN 0001-0782. doi: 10.1145/179606.179671. URL http://doi.acm.org/10.1145/179606.179671. (Cited on page 1.)

T. Berners-Lee, J. Hendler, and L. Ora. *The Semantic Web*. Scientific American, 2001. (Cited on pages 3 and 26.)

J. Bhagat, F. Tanoh, E. Nzuobontane, T. Laurent, J. Orlowski, M. Roos, K. Wolstencroft, S. Aleksejevs, R. Stevens, S. Pettifer, R. Lopez, and C. A. Goble. BioCatalogue: a universal catalogue of web services for the life sciences. *Nucleic acids research*, Article in press, May 2010. doi: 10.1093/nar/gkq394. URL http://dx.doi.org/10.1093/nar/gkq394. (Cited on page 130.)

C. Bizer and A. Schultz. The berlin sparql benchmark. *International Journal On Semantic Web and Information Systems*, 2009. (Cited on pages 39 and 162.)

C. Bizer and A. Seaborne. D2RQ - treating non-RDF databases as virtual RDF graphs. In *ISWC2004 (posters)*, November 2004. URL http://sites.wiwiss.fu-berlin.de/suhl/bizer/pub/Bizer-D2RQ-ISWC2004-Poster.pdf. (Cited on page 52.)

C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *Int. J. Semantic Web Inf. Syst.*, 5(3):1–22, 2009a. (Cited on pages 3, 40 and 72.)

C. Bizer, J. Lehmann, G. Kobilarov, S. Auer, C. Becker, R. Cyganiak, and S. Hellmann. Dbpedia - a crystallization point for the web of data. *J. Web Sem.*, 7(3):154–165, 2009b. (Cited on pages 80 and 91.)

I. Blanquer, V. Hernandez, D. Segrelles, and E. Torres. Enhancing privacy and authorization control scalability in the grid through ontologies. *IEEE Trans Inf Technol Biomed*, 13 (1):16–24, 2009. ISSN 1558-0032 (Electronic). doi: 10.1109/TITB.2008.2003369. (Cited on page 60.)

M. Boeker, H. Stenzhorn, K. Kumpf, P. Bijlenga, S. Schulz, and S. Hanser. The @neurist ontology of intracranial aneurysms: providing terminological services for an integrated it infrastructure. In *AMIA ... Annual Symposium proceedings / AMIA Symposium. AMIA Symposium*, pages 56–60, 2007. (Cited on page 50.)

R. Bolze, F. Cappello, E. Caron, M. Dayde, F. Desprez, E. Jeannot, Y. Jégou, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, P. Primet, B. Quétier, O. Richard, T. El-Ghazali, and I. Touche. Grid'5000: A Large Scale And Highly Reconfigurable Experimental Grid Testbed. *International Journal of High Performance Computing Applications*, 20(4):481–494, 2006. doi: 10.1177/1094342006070078. URL http://hal.inria.fr/hal-00684943. (Cited on page 90.)

A. Boukottaya and C. Vanoirbeek. Schema matching for transforming structured documents. In *Proceedings of the 2005 ACM symposium on Document engineering*, DocEng '05, pages 101–110, New York, NY, USA, 2005. ACM. ISBN 1-59593-240-2. doi: 10.1145/1096601.1096629. URL http://doi.acm.org/10.1145/1096601.1096629. (Cited on page 110.)

S. Bowers. Scientific workflow, provenance, and data modeling challenges and approaches. *Journal on Data Semantics*, 1:19–30, 2012. ISSN 1861-2032. URL http://dx.doi.org/10.1007/s13740-012-0004-y. 10.1007/s13740-012-0004-y. (Cited on page 116.)

V. Breton, K. Dean, T. Solomonides, I. Blanquer, V. Hernandez, E. Medico, N. Maglaveras, S. Benkner, G. Lonsdale, S. Lloyd, K. Hassan, R. McClatchey, S. Miguet, J. Montagnat, X. Pennec, W. D. Neve, C. D. Wagter, G. Heeren, L. Maigne, K. Nozaki, M. Taillet, H. Bilofsky, R. Ziegler, M. Hofmann, C. Jones, M. Cannataro, P. Veltri, G. Aloisio, S. Fiore, M. Mirto, I. Chouvarda, A. Koutkias, A. Malousi, V. Lopez, I. Oliveira, J. Sanchez, F. Martin-Sanchez, G. D. Moor, B. Claerhout, and J. Herveg. The healthgrid white paper. *Studies in health technology and informatics, IOS Press*, 112:249–321, 2005. URL http://www.ncbi.nlm.nih.gov/pubmed/15923733. (Cited on page 55.)

J. Broekstra, A. Kampman, and F. V. Harmelen. Sesame: A generic architecture for storing and querying rdf and rdf schema. pages 54–68. Springer, 2002. (Cited on page 37.)

G. Brumfiel. High-energy physics: Down the petabyte highway. *Nature*, 469(7330):282–283, Jan. 2011. ISSN 0028-0836. doi: 10.1038/469282a. URL http://dx.doi.org/10.1038/469282a. (Cited on page 2.)

A. Bucur, S. Ruping, T. Sengstag, S. Sfakianakis, M. Tsiknakis, and D. Wegener. The acgt project in retrospect: Lessons learned and future outlook. *Procedia Computer Science*, 4(0):1119 – 1128, 2011. ISSN 1877-0509. doi: 10.1016/j.procs.2011.04.119. URL http://www.sciencedirect.com/science/article/pii/S1877050911001773. <ce:title>Proceedings of the International Conference on Computational Science, ICCS 2011</ce:title>. (Cited on page 51.)

C. Buil Aranda and Ó. Corcho García. Federating queries to rdf repositories. 2010. URL http://oa.upm.es/3302/. (Cited on page 45.)

D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The dl-lite family. *J. of Automated Reasoning*, 39(3):385–429, 2007. (Cited on page 36.)

N. Capit, G. Da Costa, Y. Georgiou, G. Huard, C. Martin, G. Mounie, P. Neyron, and O. Richard. A batch scheduler with high level components. In *Cluster Computing and the Grid, 2005. CCGrid 2005. IEEE International Symposium on*, volume 2, pages 776 – 783 Vol. 2, may 2005. doi: 10.1109/CCGRID.2005.1558641. (Cited on page 90.)

J. Cardoso, J. Miller, and S. Emani. Web Services Discovery Utilizing Semantically Annotated WSDL. *Reasoning Web*, pages 240–268, 2008. URL http://dx.doi.org/10.1007/978-3-540-85658-0_7. (Cited on page 111.)

N. Cerezo and J. Montagnat. Scientific Workflow Reuse through Conceptual Workflows. In *6th Workshop on Workflows in Support of Large-Scale Science(WORKS'11)*, , Seattle, WA, USA, Nov. 2011. ACM. URL http://hal.archives-ouvertes.fr/hal-00677831/PDF. (Cited on pages 122, 140 and 203.)

D. W. Chadwick, G. Zhao, S. Otenko, R. Laborde, L. Su, and T.-A. Nguyen. Permis: a modular authorization infrastructure. *Concurrency and Computation: Practice and Experience*, 20(11):1341–1357, 2008. (Cited on page 59.)

A. Chebotko, S. Lu, X. Fei, and F. Fotouhi. Rdfprov: A relational rdf store for querying and managing scientific workflow provenance. *Data Knowl. Eng.*, 69:836–865, August 2010. ISSN 0169-023X. doi: http://dx.doi.org/10.1016/j.datak.2010.03.005. URL http://dx.doi.org/10.1016/j.datak.2010.03.005. (Cited on page 118.)

E. Christensen, F. Curbera, G. Meredith, and S. Weerawarana. Web Services Description Language (WSDL) 1.1 [http://www.w3.org/TR/wsdl], March 2001. URL http://www.w3.org/TR/wsdl. (Cited on page 108.)

H.-P. D. Company. Jena – A Semantic Web Framework for Java[http://jena.sourceforge.net/], 2009. URL http://jena.sourceforge.net/. (Cited on page 35.)

O. Corby and C. Faron-Zucker. The kgram abstract machine for knowledge graph querying. *Web Intelligence and Intelligent Agent Technology, IEEE/WIC/ACM International Conference on*, 1:338–341, 2010. doi: http://doi.ieeecomputersociety.org/10.1109/WI-IAT.2010.144. (Cited on page 74.)

O. Corby, R. Dieng-Kuntz, and C. Faron-Zucker. Querying the semantic web with corese search engine. In *ECAI*, pages 705–709, 2004. URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.80.887&rep=rep1&type=pdf. (Cited on page 111.)

O. Corby, A. Gaignard, C. Faron-Zucker, and J. Montagnat. KGRAM Versatile Inference and Query Engine for the Web of Linked Data. In *IEEE/WIC/ACM International Conference on Web Intelligence(WI'12)*, , Macao, China, 2012. URL http://hal.archives-ouvertes.fr/docs/00/74/67/72/PDF/wi12.pdf. (Cited on pages 7 and 14.)

L. Costabello, S. Villata, and F. Gandon. Context-aware access control for rdf graph stores. In L. D. Raedt, C. Bessière, D. Dubois, P. Doherty, P. Frasconi, F. Heintz, and P. J. F. Lucas, editors, *ECAI*, volume 242 of *Frontiers in Artificial Intelligence and Applications*, pages 282–287. IOS Press, 2012. ISBN 978-1-61499-097-0. (Cited on page 69.)

R. Cyganiak. A relational algebra for SPARQL. 2005. URL http://www.hpl.hp.com/techreports/2005/HPL-2005-170.html. (Cited on page 46.)

S. Das, S. Sundara, and R. Cyganiak. R2RML: RDB to RDF Mapping Language, September 2012. URL http://www.w3.org/TR/r2rml. (Cited on page 202.)

J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, Jan. 2008. ISSN 0001-0782. doi: 10.1145/1327452.1327492. URL http://doi.acm.org/10.1145/1327452.1327492. (Cited on page 2.)

E. Deelman, D. Gannon, M. Shields, and I. Taylor. Workflows and e-science: An overview of workflow system features and capabilities. *Future Generation Computer Systems*, 25(5):528 – 540, 2009. ISSN 0167-739X. doi: 10.1016/j.future.2008.06.012. URL http://www.sciencedirect.com/science/article/pii/S0167739X08000861. (Cited on page 116.)

K. Derouiche and D. A. Nicole. Semantically resolving type mismatches in scientific workflows. In *OTM Workshops (1)*, pages 125–135, 2007. (Cited on page 109.)

R. Dieng-Kuntz and O. Corby. Conceptual graphs for semantic web applications. In *Proceedings of the 13th international conference on Conceptual Structures: common Semantics for Sharing Knowledge*, ICCS'05, pages 19–50, Berlin, Heidelberg, 2005. Springer-Verlag. ISBN 3-540-27783-8, 978-3-540-27783-5. doi: 10.1007/11524564_2. URL http://dx.doi.org/10.1007/11524564_2. (Cited on page 74.)

L. Ding, J. Michaelis, J. McCusker, and D. L. McGuinness. Linked provenance data: A semantic web-based approach to interoperable workflow traces. *Future Generation*

Computer Systems, In Press, Corrected Proof:–, 2010. ISSN 0167-739X. doi: DOI: 10.1016/j.future.2010.10.011. URL http://www.sciencedirect.com/science/article/B6V06-51B1WH8-2/2/fa49c1d11941a1cdac36282ddaefd33c. (Cited on page 120.)

I. Dinov, K. Lozev, P. Petrosyan, Z. Liu, P. Eggert, J. Pierce, A. Zamanyan, S. Chakrapani, J. Van Horn, D. S. Parker, R. Magsipoc, K. Leung, B. Gutman, R. Woods, and A. Toga. Neuroimaging study designs, computational analyses and data provenance using the loni pipeline. *PLoS ONE*, 5(9):e13070, 09 2010. doi: 10.1371/journal.pone.0013070. (Cited on page 117.)

I. D. Dinov, J. D. Van Horn, K. M. Lozev, R. Magsipoc, P. Petrosyan, Z. Liu, A. Mackenzie-Graham, P. Eggert, D. S. Parker, and A. W. Toga. Efficient, distributed and interactive neuroimaging data analysis using the loni pipeline. *Front Neuroinformatics*, 3:22, 2009. doi: 10.3389/neuro.11.022.2009. (Cited on page 116.)

O. Erling and I. Mikhailov. RDF Support in the Virtuoso DBMS. In S. Auer, C. Bizer, C. Müller, and A. V. Zhdanova, editors, *Conference on Social Semantic Web*, volume 113 of *LNI*, pages 59–68. GI, 2007. ISBN 978-3-88579-207-9. (Cited on page 37.)

J. Farrell and H. Lausen. Semantic Annotations for WSDL and XML Schema [http://www.w3.org/tr/sawsdl], August 2007. URL http://www.w3.org/TR/sawsdl. (Cited on page 109.)

D. F. Ferraiolo, R. S. Sandhu, S. I. Gavrila, D. R. Kuhn, and R. Chandramouli. Proposed nist standard for role-based access control. *ACM Trans. Inf. Syst. Secur.*, 4(3):224–274, 2001. (Cited on page 62.)

D. Florescu, A. Levy, I. Manolescu, and D. Suciu. Query optimization in the presence of limited access patterns. *SIGMOD Rec.*, 28:311–322, June 1999. ISSN 0163-5808. doi: http://doi.acm.org/10.1145/304181.304210. URL http://doi.acm.org/10.1145/304181.304210. (Cited on page 42.)

G. Flouris, I. Fundulaki, M. Michou, and G. Antoniou. Controlling access to rdf graphs. In *Proceedings of the Third future internet conference on Future internet*, FIS'10, pages 107–117, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15876-5, 978-3-642-15876-6. URL http://dl.acm.org/citation.cfm?id=1929268.1929280. (Cited on page 69.)

G. Forestier, A. Marion, H. Benoit-Cattin, P. Clarysse, D. Friboulet, T. Glatard, P. Hugonnard, C. Lartizien, H. Liebgott, J. Tabary, and B. Gibaud. Sharing object models for multi-modality medical image simulation: A semantic approach. *Computer-Based Medical Systems, IEEE Symposium on*, 0:1–6, 2011a. doi: http://doi.ieeecomputersociety.org/10.1109/CBMS.2011.5999167. (Cited on pages 7 and 178.)

G. Forestier, A. Marion, H. Benoit-Cattin, P. Clarysse, D. Friboulet, T. Glatard, P. Hugonnard, C. Lartizien, H. Liebgott, J. Tabary, and B. Gibaud. Sharing object models for multi-modality medical image simulation: A semantic approach. In *Computer-Based Medical Systems (CBMS), 2011 24th International Symposium on*, pages 1 –6, june 2011b. doi: 10.1109/CBMS.2011.5999167. (Cited on page 122.)

P. Fox and J. A. Hendler. Semantic eScience: encoding meaning in next-generation digitally enhanced science. In T. Hey, S. Tansley, and K. M. Tolle, editors, *The Fourth Paradigm*, pages 147–152. Microsoft Research, 2009. ISBN 978-0982544204. (Cited on page 3.)

J. Futrelle, J. Gaynor, J. Plutchak, J. D. Myers, R. E. McGrath, P. Bajcsy, J. Kastner, K. Kotwani, J. S. Lee, L. Marini, R. Kooper, T. McLaren, and Y. Liu. Semantic middleware for e-science knowledge spaces. In *MGC '09: Proceedings of the 7th International Workshop on Middleware for Grids, Clouds and e-Science*, pages 1–6, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-847-6. doi: http://doi.acm.org/10.1145/1657120.1657124. (Cited on page 38.)

A. Gaignard and J. Montagnat. A distributed security policy for neuroradiological data sharing. In *HealthGrid'09*, , pages 257–262, Berlin, Germany, June 2009. IOS Press. URL http://nyx.unice.fr/publis/gaignard-montagnat:2009.pdf. (Cited on pages 6 and 13.)

A. Gaignard and J. Montagnat. Intégration des connaissances en neurosciences dans un environnement multi-centrique. In *21es Journées francophones d'Ingénierie des Connaissances(IC 2010)*, , pages 21–30, Nîmes, France, June 2010. URL http://hal.archives-ouvertes.fr/hal-00691827/PDF. (Cited on pages 7 and 14.)

A. Gaignard, J. Montagnat, B. Wali, and B. Gibaud. Characterizing semantic service parameters with Role concepts to infer domain-specific knowledge at runtime. In *International Conference on Knowledge Engineering and Ontology Development, KEOD 2011 AR=11%*, , Paris, France, Oct. 2011. URL http://hal.archives-ouvertes.fr/hal-00677829/PDF. (Cited on pages 7 and 14.)

A. Gaignard, J. Montagnat, C. Faron-Zucker, and O. Corby. Semantic Federation of Distributed Neurodata. In *MICCAI-DCITAI workshop, Data- and Compute-Intensive Clinical and Translational Imaging Applications*, , Nice, France, Oct. 2012a. (Cited on pages 7 and 14.)

A. Gaignard, J. Montagnat, C. Faron-Zucker, and O. Corby. Fédération multi-sources en neurosciences : intégration de données relationnelles et sémantiques. In *23es Journées francophones d'Ingénierie des Connaissances (IC 2012)*, , Paris, France, June 2012b. URL http://hal.archives-ouvertes.fr/hal-00691827/PDF. (Cited on pages 7 and 14.)

B. Gibaud, F. Ahmad, C. Barillot, F. Michel, B. Wali, B. Batrancourt, M. Dojat, P. Girard, A. Gaignard, D. Lingrand, J. Montagnat, J. Rojas Balderrama, G. Malandain, X. Pennec, D. Godard, G. Kassel, and M. Pélégrini-Issac. A federated system for sharing and reuse of images and image processing tools in neuroimaging. In *Computer Assisted Radiology and Surgery(CARS'11)*, , Berlin, Germany, June 2011a. (Cited on page 170.)

B. Gibaud, G. Kassel, M. Dojat, B. Batrancourt, F. Michel, A. Gaignard, and J. Montagnat. NeuroLOG: sharing neuroimaging data using an ontology-based federated approach. *AMIA ... Annual Symposium proceedings [electronic resource] / AMIA Symposium. AMIA Symposium.*, 2011:472–80, 2011b. URL http://hal.archives-ouvertes.fr/hal-00683087. (Cited on page 122.)

B. Gibaud, G. Forestier, H. Benoit-Cattin, F. Cervenansky, P. Clarysse, D. Friboulet, A. Gaignard, P. Hugonnard, C. Lartizien, H. Liebgott, J. Montagnat, J. Tabary, and T. Glatard. Ontovip: An ontology for the annotation of object models used for medical image simulation. In *Healthcare Informatics, Imaging and Systems Biology (HISB), 2012 IEEE Second International Conference on*, page 110, sept. 2012. doi: 10.1109/HISB.2012.35. (Cited on page 200.)

Y. Gil, V. Ratnakar, E. Deelman, G. Mehta, and J. Kim. Wings for pegasus: Creating large-scale scientific applications using semantic representations of computational workflows. In *AAAI*, pages 1767–1774. AAAI Press, 2007. ISBN 978-1-57735-323-2. (Cited on page 116.)

M. Gillam, C. Feied, J. Handler, E. Moody, B. Shneiderman, C. Plaisant, M. Smith, and J. Dickason. The healthcare singularity and the age of semantic medicine. In T. Hey, S. Tansley, and K. M. Tolle, editors, *The Fourth Paradigm*, pages 57–64. Microsoft Research, 2009. ISBN 978-0982544204. (Cited on page 2.)

T. Glatard, J. Montagnat, D. Lingrand, and X. Pennec. Flexible and efficient workflow deployement of data-intensive applications on grids with MOTEUR. *International Journal of High Performance Computing Applications (IJHPCA) IF=1.109 Special issue on Special Issue on Workflows Systems in Grid Environments*, 22(3):347–360, Aug. 2008. URL http://nyx.unice.fr/publis/glatard-montagnat-etal:2008.pdf. (Cited on page 117.)

T. Glatard, C. Lartizien, B. Gibaud, R. Ferreira Da Silva, G. Forestier, F. Cervenansky, M. Alessandrini, H. Benoit-Cattin, O. Bernard, S. Camarasu-Pop, N. Cerezo, P. Clarysse, A. Gaignard, P. Hugonnard, H. Liebgott, S. Marache, A. Marion, J. Montagnat, J. Tabary, and D. Friboulet. A Virtual Imaging Platform for multi-modality medical image simulation. *IEEE Transactions on Medical Imaging (TMI) IF=3.6*, 32(1):110 – 118, Jan. 2013. URL http://www.hal.inserm.fr/docs/00/76/24/97/PDF/06310064.pdf. (Cited on pages 10 and 178.)

C. Goble and R. Stevens. State of the nation in data integration for bioinformatics. *J. of Biomedical Informatics*, 41(5):687–693, Oct. 2008. ISSN 1532-0464. doi: 10.1016/j.jbi.

2008.01.008. URL http://dx.doi.org/10.1016/j.jbi.2008.01.008. (Cited on page 2.)

C. Goble, D. De Roure, and S. Bechhofer. Accelerating scientists' knowledge turns. In J. Filipe and A. L. N. Fred, editors, *KDIR*, page 7. SciTePress, 2011. ISBN 978-989-8425-79-9. (Cited on page 57.)

C. A. Goble and D. D. Roure. The impact of workflow tools on data-centric research. In T. Hey, S. Tansley, and K. M. Tolle, editors, *The Fourth Paradigm*, pages 137–145. Microsoft Research, 2009. ISBN 978-0982544204. (Cited on page 2.)

A. González-Beltrán, B. Tagger, and A. Finkelstein. Ontology-based queries over cancer data. In *SWAT4LS*, 2010. (Cited on page 49.)

A. González-Beltrán, B. Tagger, and A. Finkelstein. Federated ontology-based queries over cancer data. *BMC Bioinformatics*, 13:1–24, 2012. doi: 10.1186/1471-2105-13-S1-S9. URL http://dx.doi.org/10.1186/1471-2105-13-S1-S9. (Cited on pages 49 and 53.)

P. M. K. Gordon and C. W. Sensen. Creating bioinformatics semantic web services from existing web services: A real-world application of sawsdl. In *ICWS*, pages 608–614, 2008. (Cited on page 113.)

O. Görlitz and S. Staab. SPLENDID: SPARQL Endpoint Federation Exploiting VOID Descriptions. In *Proceedings of the 2nd International Workshop on Consuming Linked Data*, Bonn, Germany, 2011. URL http://uni-koblenz.de/~goerlitz/publications/GoerlitzAndStaab_COLD2011.pdf. (Cited on pages 41, 43 and 73.)

K. Grant Clark, L. Feigenbaum, and E. Torres. SPARQL Protocol for RDF [http://www.w3.org/tr/rdf-sparql-protocol/], January 2008. URL http://www.w3.org/TR/rdf-sparql-protocol/. (Cited on page 36.)

B. C. Grau, I. Horrocks, B. Motik, B. Parsia, P. Patel-Schneider, and U. Sattler. Owl 2: The next step for owl. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):309 – 322, 2008. ISSN 1570-8268. doi: DOI:10.1016/j.websem.2008.05.001. URL http://www.sciencedirect.com/science/article/B758F-4TP1FC8-1/2/9d2f647c7ac874b8f8baa9cf92cf73a3. Semantic Web Challenge 2006/2007. (Cited on page 31.)

M. Gruninger, R. Hull, and S. McIlraith. A short overview of flows: A first-order logic ontology of web services. *IEEE Data Engineering Bulletin.*, 31(3):3–7, 2008. URL ieee08.pdf. (Cited on pages 123 and 130.)

Y. Guo, Z. Pan, and J. Heflin. Lubm: A benchmark for owl knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2-3):158–182, October 2005. ISSN 15708268. doi: 10.1016/j.websem.2005.06.005. URL http://dx.doi.org/10.1016/j.websem.2005.06.005. (Cited on pages 39 and 162.)

L. M. Haas, J. C. Freytag, G. M. Lohman, and H. Pirahesh. Extensible query processing in starburst. *SIGMOD Rec.*, 18:377–388, June 1989. ISSN 0163-5808. doi: http://doi.acm.org/10.1145/66926.66962. URL http://doi.acm.org/10.1145/66926.66962. (Cited on page 41.)

L. M. Haas, D. Kossmann, E. L. Wimmers, and J. Yang. Optimizing queries across diverse data sources. In *Proceedings of the 23rd International Conference on Very Large Data Bases*, VLDB '97, pages 276–285, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. ISBN 1-55860-470-7. URL http://dl.acm.org/citation.cfm?id=645923.670995. (Cited on pages 42 and 43.)

P. Haase, T. Mathäß, and M. Ziller. An evaluation of approaches to federated query processing over linked data. In *Proceedings of the 6th International Conference on Semantic Systems*, I-SEMANTICS '10, pages 5:1–5:9, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0014-8. doi: http://doi.acm.org/10.1145/1839707.1839713. URL http://doi.acm.org/10.1145/1839707.1839713. (Cited on pages 20 and 23.)

F. W. Hartel, S. de Coronado, R. Dionne, G. Fragoso, and J. Golbeck. Modeling a description logic vocabulary for cancer research. *J. of Biomedical Informatics*, 38(2): 114–129, Apr. 2005. ISSN 1532-0464. doi: 10.1016/j.jbi.2004.09.001. URL http://dx.doi.org/10.1016/j.jbi.2004.09.001. (Cited on page 49.)

A. Harth, K. Hose, M. Karnstedt, A. Polleres, K.-U. Sattler, and J. Umbrich. Data summaries for on-demand queries over linked data. In *Proceedings of the 19th international conference on World wide web*, WWW '10, pages 411–420, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-799-8. doi: http://doi.acm.org/10.1145/1772690.1772733. URL http://doi.acm.org/10.1145/1772690.1772733. (Cited on page 41.)

O. Hartig, C. Bizer, and J.-C. Freytag. Executing sparql queries over the web of linked data. *informatikhuberlinde*, 5823:293–309, 2009. URL http://www.springerlink.com/index/Q37381173G66W7N2.pdf. (Cited on page 47.)

R. Hasan and F. Gandon. Linking justifications in the collaborative semantic web applications. In *Proceedings of the 21st international conference companion on World Wide Web*, WWW '12 Companion, pages 1083–1090, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1230-1. doi: 10.1145/2187980.2188245. URL http://doi.acm.org/10.1145/2187980.2188245. (Cited on page 203.)

P. Hayes and B. McBride. RDF semantics [http://www.w3.org/tr/rdf-mt], February 2004. URL http://www.w3.org/TR/rdf-mt. (Cited on pages 34 and 120.)

J. Henriksson, M. Pradel, S. Zschaler, and J. Z. Pan. Ontology design and reuse with conceptual roles. In *Proceedings of the 2nd International Conference on Web Reasoning and Rule Systems*, RR '08, pages 104–118, Berlin, Heidelberg, 2008. Springer-Verlag. ISBN 978-3-540-88736-2. doi: http://dx.doi.org/10.1007/978-3-540-88737-9_9. URL http://dx.doi.org/10.1007/978-3-540-88737-9_9. (Cited on page 129.)

T. Hey and A. Trefethen. Cyberinfrastructure for e-Science. *Science*, 308(5723):817–821, May 2005. ISSN 1095-9203. doi: 10.1126/science.1110410. URL http://dx.doi.org/10.1126/science.1110410. (Cited on page 3.)

B. Holbrook and W. Brown. *A history of computing research at Bell Laboratories (1937-75).* 1982. (Cited on page 1.)

I. Horrocks, P. F. Patel-Schneider, and F. V. Harmelen. From shiq and rdf to owl: The making of a web ontology language. *Journal of Web Semantics*, 1:2003, 2003. (Cited on page 31.)

I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. SWRL: A Semantic Web Rule Language Combining OWL and RuleML [http://www.w3.org/submission/swrl/], May 2004. URL http://www.w3.org/Submission/SWRL/. (Cited on page 34.)

R. Housley and T. Polk. *Planning for PKI: Best Practices Guide for Deploying Public Key Infrastructure*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 2001. ISBN 0471397024. URL http://portal.acm.org/citation.cfm?id=558370. (Cited on page 57.)

D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. R. Pocock, P. Li, and T. Oinn. Taverna: a tool for building and running workflows of services. *Nucleic Acids Research*, 34(suppl 2): W729–W732, 2006. doi: 10.1093/nar/gkl320. URL http://nar.oxfordjournals.org/content/34/suppl_2/W729.abstract. (Cited on page 116.)

L. D. Ibáñez, H. Skaf-Molli, P. Molli, and O. Corby. Synchronizing semantic stores with commutative replicated data types. In A. Mille, F. L. Gandon, J. Misselis, M. Rabinovich, and S. Staab, editors, *WWW (Companion Volume)*, pages 1091–1096. ACM, 2012. ISBN 978-1-4503-1230-1. (Cited on page 202.)

F. T. Imam, S. D. Larson, J. S. Grethe, A. Gupta, A. Bandrowski, and M. E. Martone. NIF-STD and NeuroLex: Comprehensive Neuroscience Ontology Development Based on Multiple Biomedical Ontologies and Community Involvement. In *ICBO*, 2011. (Cited on pages 53 and 171.)

F. T. Imam, S. D. Larson, A. Bandrowski, J. S. Grethe, A. Gupta, and M. E. Martone. Maturation of neuroscience information framework: An ontology driven information system for neuroscience. In M. Donnelly and G. Guizzardi, editors, *FOIS*, volume 239 of *Frontiers in Artificial Intelligence and Applications*, pages 15–28. IOS Press, 2012. (Cited on page 53.)

W. Jie, J. Arshad, R. Sinnott, P. Townend, and Z. Lei. A review of grid authentication and authorization technologies and support for federated access control. *ACM Comput. Surv.*, 43(2):12:1–12:26, Feb. 2011. ISSN 0360-0300. doi: 10.1145/1883612.1883619. URL http://doi.acm.org/10.1145/1883612.1883619. (Cited on page 59.)

M. Kifer and H. Boley. Rif overview [http://www.w3.org/tr/rif-overview], 2010. URL http://www.w3.org/TR/rif-overview. (Cited on page 34.)

J. Kim, E. Deelman, Y. Gil, G. Mehta, and V. Ratnakar. Provenance trails in the wings/pegasus system. *Concurrency and Computation: Practice and Experience*, 20(5):587–597, 2008a. (Cited on page 120.)

J.-D. Kim, H. Shin, D. Jeong, and D.-K. Baik. Jena storage plug-in providing an improved query processing performance for semantic grid computing environment. In *Computational Science and Engineering Workshops, 2008. CSEWORKSHOPS '08. 11th IEEE International Conference on*, pages 393 –398, 16-18 2008b. doi: 10.1109/CSEW.2008.71. (Cited on page 36.)

A. Kiryakov, D. Ognyanov, and D. Manov. OWLIM –A Pragmatic Semantic Repository for OWL. *Web Information Systems Engineering –WISE 2005 Workshops*, pages 182–192, 2005. URL http://dx.doi.org/10.1007/11581116_19. (Cited on page 38.)

M. Koehler and S. Benkner. A service oriented approach for distributed data mediation on the grid. In *Grid and Cooperative Computing, 2009. GCC '09. Eighth International Conference on*, pages 401 –408, aug. 2009. doi: 10.1109/GCC.2009.35. (Cited on page 50.)

J. Kopecký, T. Vitvar, C. Bournez, and J. Farrell. Sawsdl: Semantic annotations for wsdl and xml schema. *IEEE Internet Computing*, 11(6):60–67, 2007. (Cited on page 130.)

J. Kopecký, K. Gomadam, and T. Vitvar. hRESTS: An HTML Microformat for Describing RESTful Web Services. *Web Intelligence and Intelligent Agent Technology, IEEE/WIC/ACM International Conference on*, 1:619–625, 2008. doi: http://doi.ieeecomputersociety.org/10.1109/WIIAT.2008.379. (Cited on page 111.)

D. Kossmann. The state of the art in distributed query processing. *ACM Comput. Surv.*, 32:422–469, December 2000. ISSN 0360-0300. doi: http://doi.acm.org/10.1145/371578.371598. URL http://doi.acm.org/10.1145/371578.371598. (Cited on pages 41 and 45.)

D. Kourtesis and I. Paraskakis. Combining SAWSDL, OWL-DL and UDDI for Semantically Enhanced Web Service Discovery. In *5th European Semantic Web Conference (ESWC2008)*, pages 614–628, June 2008. URL http://data.semanticweb.org/conference/eswc/2008/paper/375. (Cited on page 110.)

U. Küster and B. König-ries. Semantic mediation between business partners - a sws-challenge solution using diane service descriptions. *Web Intelligence and Intelligent Agent Technology, International Conference on*, 0:139–143, 2007. doi: http://doi.ieeecomputersociety.org/10.1109/WI-IATW.2007.14. (Cited on page 111.)

U. Küster, B. König-Ries, and A. Krug. OPOSSum - an online portal to collect and share semantic service descriptions. In *Proceedings of the 5th European Semantic Web Conference (ESWC08), Poster Session*, Tenerife, Canary Islands, Spain, June 2008. (Cited on page 112.)

G. Ladwig and T. Tran. Linked data query processing strategies. In *Proceedings of the 9th international semantic web conference on The semantic web - Volume Part I*, ISWC'10, pages 453–469, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-17745-X, 978-3-642-17745-3. URL http://dl.acm.org/citation.cfm?id=1940281.1940311. (Cited on pages 40, 41, 46 and 73.)

G. Ladwig and T. Tran. Sihjoin: querying remote and local linked data. In *Proceedings of the 8th extended semantic web conference on The semantic web: research and applications - Volume Part I*, ESWC'11, pages 139–153, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-21033-4. URL http://dl.acm.org/citation.cfm?id=2008892.2008905. (Cited on page 47.)

B. Lang, I. Foster, F. Siebenlist, R. Ananthakrishnan, and T. Freeman. A multipolicy authorization framework for grid security. In *in Proceedings of the Fifth IEEE Symposium on Network Computing and Application*, pages 269–272, 2006. (Cited on page 50.)

A. Langegger. *A Flexible Architecture for Virtual Information Integration based on Semantic Web Concepts*. PhD thesis, 2010. (Cited on page 23.)

A. Langegger and W. Wöß. Rdfstats - an extensible rdf statistics generator and library. In A. M. Tjoa and R. Wagner, editors, *DEXA Workshops*, pages 79–83. IEEE Computer Society, 2009. ISBN 978-0-7695-3763-4. (Cited on pages 44 and 201.)

A. Langegger, W. Wöß, and M. Blöchl. A semantic web middleware for virtual data integration on the web. In M. Hauswirth, M. Koubarakis, and S. Bechhofer, editors, *Proceedings of the 5th European Semantic Web Conference*, LNCS, Berlin, Heidelberg, June 2008. Springer Verlag. URL http://data.semanticweb.org/conference/eswc/2008/papers/244. (Cited on pages 44 and 73.)

P. Larvet, B. Christophe, and A. Pastor. Semantization of legacy web services: From wsdl to sawsdl. In *ICIW '08: Proceedings of the 2008 Third International Conference on Internet and Web Applications and Services*, pages 130–135, Washington, DC, USA, 2008. IEEE Computer Society. ISBN 978-0-7695-3163-2. doi: http://dx.doi.org/10.1109/ICIW.2008.119. (Cited on page 113.)

F. Lecue and A. Léger. A formal model for semantic web service composition. In *5th International Semantic Web Conference (ISWC2006)*, 2006. URL http://data.semanticweb.org/conference/iswc/2006/paper-35. (Cited on page 110.)

F. Lécue, S. Salibi, P. Bron, and A. Moreau. Semantic and syntactic data flow in web service composition. *Web Services, IEEE International Conference on*, 0:211–218, 2008. doi: http://doi.ieeecomputersociety.org/10.1109/ICWS.2008.96. (Cited on page 110.)

M. Lenzerini. Data integration: a theoretical perspective. In *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 233–246, New York, NY, USA, 2002. ACM. ISBN 1581135076. doi:

http://dx.doi.org/10.1145/543613.543644. URL http://dx.doi.org/10.1145/543613.543644. (Cited on page 22.)

M. Lo and F. Gandon. Integrating dynamic resources in corporate semantic web: an approach to enterprise application integration using semantic web services. Technical Report 5663, INRIA, August 2005. (Cited on page 111.)

P. Lord, P. Alper, C. Wroe, and C. Goble. Feta: A light-weight architecture for user oriented semantic service discovery. In *European Semantic Web Conference*, pages 17–31. Springer Berlin / Heidelberg, 2005. doi: 10.1007/11431053_2. (Cited on page 112.)

S. Lynden, A. Mukherjee, A. C. Hume, A. A. Fernandes, N. W. Paton, R. Sakellariou, and P. Watson. The design and implementation of ogsa-dqp: A service-based distributed query processor. *Future Generation Computer Systems*, 25(3):224 – 236, 2009. ISSN 0167-739X. doi: 10.1016/j.future.2008.08.003. URL http://www.sciencedirect.com/science/article/pii/S0167739X08001222. (Cited on pages 45, 49 and 50.)

S. Madougou, M. Santcroos, A. Benabdelkader, B. D. C. Van Schaik, S. Shayan, V. Korkhov, A. H. C. Van Kampen, and S. D. Olabarriaga. Provenance for distributed biomedical workflow execution. In *HealthGrid'12*, pages 91–100, 2012. (Cited on page 117.)

D. Martin, M. Burstein, J. Hobbs, O. Lassila, D. McDermott, S. McIlraith, S. Narayanan, M. Paolucci, B. Parsia, T. Payne, E. Sirin, N. Srinivasan, and K. Sycara. OWL-S: Semantic Markup for Web Services [http://www.w3.org/submission/owl-s], Novembre 2004. URL http://www.w3.org/Submission/OWL-S. (Cited on page 108.)

D. Martin, M. Burstein, D. Mcdermott, S. Mcilraith, M. Paolucci, K. Sycara, D. L. Mcguinness, E. Sirin, and N. Srinivasan. Bringing semantics to web services with owl-s. *World Wide Web*, 10(3):243–277, September 2007a. doi: 10.1007/s11280-007-0033-x. URL http://dx.doi.org/10.1007/s11280-007-0033-x. (Cited on pages 108 and 130.)

D. Martin, M. Paolucci, and M. Wagner. Bringing semantic annotations to web services: Owl-s from the sawsdl perspective. In *ISWC/ASWC*, pages 340–352, 2007b. (Cited on page 109.)

L. Martín, A. Anguita, G. de la Calle, M. García-Remesal, J. Crespo, M. Tsiknakis, and V. Maojo. Semantic data integration in the European ACGT project. In *AMIA... Annual Symposium proceedings/AMIA Symposium. AMIA Symposium*, page 1042, 2007. (Cited on page 52.)

L. Martín, A. Anguita, V. Maojo, E. Bonsma, A. I. D. Bucur, J. Vrijnsen, M. Brochhausen, C. Cocos, H. Stenzhorn, M. Tsiknakis, M. Doerr, and H. Kondylakis. Ontology based integration of distributed and heterogeneous data sources in acgt. In L. Azevedo and A. R. Londral, editors, *HEALTHINF (1)*, pages 301–306. INSTICC - Institute for Systems

and Technologies of Information, Control and Communication, 2008. ISBN 978-989-8111-16-6. (Cited on page 52.)

C. Masolo, S. Borgo, A. Gangemi, N. Guarino, and A. Oltramari. WonderWeb Deliverable D18. The WonderWeb Library of Foundational Ontologies and the DOLCE ontology, December 2003. URL http://wonderweb.semanticweb.org/deliverables/documents/D18.pdf. (Cited on pages 122 and 131.)

B. McBride. Jena: a semantic web toolkit. *Internet Computing, IEEE*, 6(6):55 – 59, nov/dec 2002. ISSN 1089-7801. doi: 10.1109/MIC.2002.1067737. (Cited on page 35.)

D. L. Mcguinness and F. van Harmelen. OWL Web Ontology Language Overview [http://www.w3.org/tr/owl-features], February 2004. URL http://www.w3.org/TR/owl-features/. (Cited on page 108.)

A. McLennan, A. Reilhac, and M. Brady. Sorteo: Monte carlo-based simulator with list-mode capabilities. In *Engineering in Medicine and Biology Society, 2009. EMBC 2009. Annual International Conference of the IEEE*, pages 3751 –3754, sept. 2009. doi: 10.1109/IEMBS.2009.5334536. (Cited on page 179.)

N. Mehandjiev, F. Lecue, U. Wajid, and A. Namoun. Assisted service composition for end users. In *Proceedings of the 2010 Eighth IEEE European Conference on Web Services*, ECOWS '10, pages 131–138, Washington, DC, USA, 2010. IEEE Computer Society. ISBN 978-0-7695-4310-9. doi: http://dx.doi.org/10.1109/ECOWS.2010.30. URL http://dx.doi.org/10.1109/ECOWS.2010.30. (Cited on page 148.)

F. Michel, A. Gaignard, F. Ahmad, C. Barillot, B. Batrancourt, M. Dojat, B. Gibaud, P. Girard, D. Godard, G. Kassel, D. Lingrand, G. Malandain, J. Montagnat, M. Pélégrini-Issac, X. Pennec, J. Rojas Balderrama, and B. Wali. Grid-wide neuroimaging data federation in the context of the NeuroLOG project. In *HealthGrid'10 AR=45%*, , pages 112–123, Paris, France, June 2010. IOS Press. URL http://nyx.unice.fr/publis/michel-gaignard-etal:2010.pdf. (Cited on pages 8 and 170.)

P. Missier and C. Goble. Workflows to open provenance graphs, round-trip. *Future Generation Computer Systems*, In Press, Corrected Proof:–, 2010. ISSN 0167-739X. doi: DOI: 10.1016/j.future.2010.10.012. URL http://www.sciencedirect.com/science/article/B6V06-51BNWMW-1/2/5751ee15fd4787993ec6911f9e4480c3. (Cited on page 119.)

P. Missier, S. Sahoo, J. Zhao, C. Goble, and A. Sheth. Janus: from workflows to semantic provenance and linked open data. In *IPAW-10*, 2010. URL http://www.mygrid.org.uk/files/presentations/SP-IPAW10.pdf. (Cited on page 121.)

J. Montagnat, A. Frohner, D. Jouvenot, C. Pera, P. Kunszt, B. Koblitz, N. Santos, C. Loomis, R. Texier, D. Lingrand, P. Guio, R. Brito Da Rocha, A. Sobreira de Almeida, and Z. Farkas. A Secure Grid Medical Data Manager Interfaced to the gLite Middleware. *Journal of Grid Computing (JGC)*, 6(1):45–59, Mar. 2008a. URL http://rainbow.

polytech.unice.fr/publis/montagnat-frohner-etal:2007.pdf. (Cited on page 60.)

J. Montagnat, A. Gaignard, D. Lingrand, J. Rojas Balderrama, P. Collet, and P. Lahire. NeuroLOG: a community-driven middleware design. In *HealthGrid*, pages 49–58, Chicago, June 2008b. IOS Press. URL http://rainbow.polytech.unice.fr/publis/montagnat-gaignard-etal:2008.pdf. (Cited on pages 6, 8 and 170.)

J. Montagnat, B. Isnard, T. Glatard, K. Maheshwari, and M. B. Fornarino. A data-driven workflow language for grids based on array programming principles. In *Proceedings of the 4th Workshop on Workflows in Support of Large-Scale Science*, WORKS '09, pages 7:1–7:10, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-717-2. doi: 10.1145/1645164.1645171. URL http://doi.acm.org/10.1145/1645164.1645171. (Cited on page 117.)

G. E. Moore. Cramming more components onto integrated circuits. *Electronics*, 38(8), April 1965. (Cited on page 1.)

L. Moreau, B. Clifford, J. Freire, Y. Gil, P. Groth, J. Futrelle, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, Y. Simmhan, E. Stephan, and J. V. den Bussche. The open provenance model — core specification (v1.1). *Future Generation Computer Systems*, December 2009. URL http://eprints.ecs.soton.ac.uk/18332/. (Cited on page 38.)

L. Moreau, L. Ding, J. Futrelle, D. Garijo, P. Groth, M. Jewell, S. Miles, P. Missier, J. Pan, and J. Zhao. Open Provenance Model (OPM) OWL Specification, October 2010. URL http://openprovenance.org/model/opmo. (Cited on page 139.)

L. Moreau, B. Clifford, J. Freire, J. Futrelle, Y. Gil, P. Groth, N. Kwasnikowska, S. Miles, P. Missier, J. Myers, B. Plale, Y. Simmhan, E. Stephan, and J. V. denBussche. The Open Provenance Model Core Specification (v1.1). *To appear in Future Generation Computer Systems*, 2011. URL papers/moreau-etal-fgcs11.pdf. (Cited on pages 118, 135, 136 and 139.)

B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz. OWL 2 Web Ontology Language Profiles, October 2009. URL http://www.w3.org/TR/owl2-profiles. (Cited on page 36.)

D. Nardi and R. J. Brachman. An introduction to description logics. In Baader et al. [2003], pages 1–40. ISBN 0-521-78176-0. (Cited on page 31.)

OASIS. Introduction to UDDI: Important Features and Functional Concepts. Technical report, OASIS, 2004. (Cited on page 110.)

E. Orri and M. Ivan. SPARQL and Scalable Inference on Demand [http://virtuoso.openlinksw.com/dataspace/dav/wiki/main/vosscalableinference], 2009. URL http://virtuoso.openlinksw.com/dataspace/dav/wiki/Main/VOSScalableInference. (Cited on page 37.)

J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of sparql. In *International Semantic Web Conference*, pages 30–43, 2006. (Cited on page 43.)

J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of sparql. *ACM Trans. Database Syst.*, 34:16:1–16:45, September 2009. ISSN 0362-5915. doi: http://doi.acm.org/10.1145/1567274.1567278. URL http://doi.acm.org/10.1145/1567274.1567278. (Cited on pages 43, 46 and 200.)

C. Petrie, T. Margaria, H. Lausen, and M. Zaremba. Introduction to the first year of the semantic web services challenge. *Semantic Web Services Challenge*, pages 1–11, novembre 2008. URL http://dx.doi.org/10.1007/978-0-387-72496-6_1. (Cited on page 107.)

S. Polovina. An introduction to conceptual graphs. In *ICCS '07: Proceedings of the 15th international conference on Conceptual Structures*, pages 1–14, Berlin, Heidelberg, 2007. Springer-Verlag. ISBN 978-3-540-73680-6. doi: http://dx.doi.org/10.1007/978-3-540-73681-3_1. (Cited on page 32.)

E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF [http://www.w3.org/TR/rdf-sparql-query], January 2008. URL http://www.w3.org/TR/rdf-sparql-query. (Cited on pages 30 and 36.)

E. Prud'hommeaux, C. Buil Aranda, A. Seaborne, A. Polleres, L. Feigenbaum, and G. Todd Williams. SPARQL 1.1 Federated Query [http://www.w3.org/TR/sparql11-federated-query], November 2011. URL http://www.w3.org/TR/sparql11-federated-query. (Cited on page 72.)

C. Qu, F. Zimmermann, K. Kumpf, R. Kamuzinzi, V. Ledent, and R. Herzog. Semantics-Enabled Service Discovery Framework in the SIMDAT Pharma Grid. *IEEE Transactions on Information Technology in Biomedicine*, 12(2):182–190, 2008. (Cited on page 112.)

B. Quilitz and U. Leser. Querying distributed rdf data sources with sparql. *The Semantic Web Research and Applications*, 5021:524–538, 2008. URL http://www.springerlink.com/index/hm1v15q75371640p.pdf. (Cited on pages 41, 42 and 73.)

H. Rahmouni, T. Solomonides, M. Mont, and S. Shiu. Privacy compliance and enforcement on European healthgrids: an approach through ontology. *Physical and Engineering Sciences*, 368(1926):4057–4072, Sept. 2010. doi: 10.1098/rsta.2010.0169. URL http://dx.doi.org/10.1098/rsta.2010.0169. (Cited on page 56.)

H. Rajasekaran, L. L. Iacono, P. Hasselmeyer, J. Fingberg, P. E. Summers, S. Benkner, G. Engelbrecht, A. Arbona, A. Chiarini, C. M. Friedrich, M. Hofmann-Apitius, K. Kumpf, B. Moore, P. Bijlenga, J. Iavindrasana, H. Müller, R. D. Hose, R. Dunlop, and A. F. Frangi. @neurist - towards a system architecture for advanced disease management through integration of heterogeneous data, computing, and complex processing services. In *CBMS*, pages 361–366, 2008. (Cited on page 68.)

K. Rohloff, M. Dean, I. Emmons, D. Ryder, and J. Sumner. An evaluation of triple-store technologies for large data stores. *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops*, pages 1105–1114, 2007. URL http://dx.doi.org/10.1007/978-3-540-76890-6_38. (Cited on page 39.)

J. Rojas Balderrama, J. Montagnat, and D. Lingrand. jGASW: A Service-Oriented Framework Supporting High Throughput Computing and Non-functional Concerns. In *IEEE International Conference on Web Services(ICWS 2010)*, , Miami, FL, USA, July 2010. IEEE Computer Society. ISBN 978-0-7695-4128-0. URL http://nyx.unice.fr/publis/rojasbalderrama-montagnat-etal:2010.pdf. (Cited on page 67.)

D. Roman, J. de Bruijn, A. Mocan, H. Lausen, J. Domingue, C. Bussler, and D. Fensel. Www: Wsmo, wsml, and wsmx in a nutshell. pages 516–522. 2006. doi: 10.1007/11836025_49. URL http://dx.doi.org/10.1007/11836025_49. (Cited on pages 108 and 130.)

O. Sacco, A. Passant, and S. Decker. An access control framework for the web of data. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2011 IEEE 10th International Conference on*, pages 456 –463, nov. 2011. doi: 10.1109/TrustCom.2011.59. (Cited on page 69.)

S. S. Sahoo and A. Sheth. Provenir ontology: Towards a framework for escience provenance management. In *Microsoft eScience Workshop*, Oct 15-17 2009. (Cited on pages 120 and 121.)

S. S. Sahoo, A. Sheth, and C. Henson. Semantic provenance for escience: Managing the deluge of scientific data. *IEEE Internet Computing*, 12(4):46–54, July 2008. ISSN 1089-7801. doi: 10.1109/MIC.2008.86. URL http://dx.doi.org/10.1109/MIC.2008.86. (Cited on page 121.)

S. S. Sahoo, O. Bodenreider, P. Hitzler, A. Sheth, and K. Thirunarayan. Provenance context entity (pace): scalable provenance tracking for scientific rdf data. In *Proceedings of the 22nd international conference on Scientific and statistical database management*, SSDBM'10, pages 461–470, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-13817-9, 978-3-642-13817-1. URL http://dl.acm.org/citation.cfm?id=1876037.1876075. (Cited on page 120.)

J. Saltz, S. Oster, S. Hastings, S. Langella, T. Kurc, W. Sanchez, M. Kher, A. Manisundaram, K. Shanbhag, and P. Covitz. caGRID: Design and implementation of the core architecture of the cancer biomedical informatics grid. *Bioinformatics*, 22(15):1910–1916, 2006. URL http://bioinformatics.oxfordjournals.org/cgi/content/abstract/22/15/1910. (Cited on page 48.)

R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *IEEE Computer*, 29(2):38–47, 1996. (Cited on page 58.)

SAP. Data federator. URL http://www.sap.com/solutions/technology/enterprise-information-management/data-integrator/index.epx. (Cited on page 170.)

D. Scardaci and G. Scuderi. Managing confidential data in the glite middleware. In *WETICE*, pages 298–299, 2007. (Cited on page 60.)

M. Schmidt, T. Hornung, G. Lausen, and C. Pinkel. Sp2bench: A sparql performance benchmark. *CoRR*, abs/0806.4627, 2008. (Cited on page 162.)

M. Schmidt, M. Meier, and G. Lausen. Foundations of sparql query optimization. In *Proceedings of the 13th International Conference on Database Theory*, ICDT '10, pages 4–33, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-947-3. doi: http://doi.acm.org/10.1145/1804669.1804675. URL http://doi.acm.org/10.1145/1804669.1804675. (Cited on page 200.)

M. Schmidt, O. Görlitz, P. Haase, G. Ladwig, A. Schwarte, and T. Tran. Fedbench: A benchmark suite for federated semantic data query processing. In *10th International Semantic Web Conference*, Bonn, Germany, 2011. (Cited on page 162.)

A. Schwarte, P. Haase, K. Hose, R. Schenkel, and M. Schmidt. Fedx: optimization techniques for federated query processing on linked data. In *Proceedings of the 10th international conference on The semantic web - Volume Part I*, ISWC'11, pages 601–616, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-25072-9. URL http://dl.acm.org/citation.cfm?id=2063016.2063055. (Cited on pages 43, 45, 73, 162, 163, 164 and 166.)

A. Seaborne, G. Manjunath, C. Bizer, J. Breslin, S. Das, I. Davis, S. Harris, K. Idehen, O. Corby, K. Kjernsmo, and B. Nowack. SPARQL Update [http://www.w3.org/Submission/SPARQL-Update], July 2008. URL http://www.w3.org/Submission/SPARQL-Update/. (Cited on page 36.)

L. Seitz, J.-M. Pierson, and L. Brunie. Key management for encrypted data storage in distributed systems. In *IEEE Security in Storage Workshop*, pages 20–30, 2003. (Cited on page 60.)

P. G. Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. Access path selection in a relational database management system. In *Proceedings of the 1979 ACM SIGMOD international conference on Management of data*, SIGMOD '79, pages 23–34, New York, NY, USA, 1979. ACM. ISBN 0-89791-001-X. doi: http://doi.acm.org/10.1145/582095.582099. URL http://doi.acm.org/10.1145/582095.582099. (Cited on page 41.)

S. T. Selvi, R. Balachandar, K. Vijayakumar, N. Mohanram, M. Vandana, and R. Raman. Semantic discovery of grid services using functionality based matchmaking algorithm. *Web Intelligence, IEEE / WIC / ACM International Conference on*, 0:170–173, 2006. doi: http://doi.ieeecomputersociety.org/10.1109/WI.2006.154. (Cited on page 110.)

A. P. Sheth, K. Gomadam, and A. Ranabahu. Semantics enhanced Services: METEOR-S, SAWSDL and SA-REST. *IEEE Data Eng. Bull.*, 31(3):8–12, 2008. (Cited on page 111.)

R. O. Sinnott, D. W. Chadwick, T. Doherty, D. Martin, A. Stell, G. Stewart, L. Su, and J. P. Watt. Advanced security for virtual organizations: The pros and cons of centralized vs decentralized security models. In *CCGRID*, pages 106–113, 2008. (Cited on pages 59 and 68.)

B. Smith, W. Ceusters, B. Klagges, J. Kohler, A. Kumar, J. Lomax, C. Mungall, F. Neuhaus, A. Rector, and C. Rosse. Relations in biomedical ontologies. *Genome Biol*, 6(5):R46, 2005. ISSN 1465-6906. doi: 10.1186/gb-2005-6-5-r46. (Cited on page 120.)

B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L. J. Goldberg, K. Eilbeck, A. Ireland, C. J. Mungall, N. Leontis, P. Rocca-Serra, A. Ruttenberg, S.-A. Sansone, R. H. Scheuermann, N. Shah, P. L. Whetzel, and S. Lewis. The obo foundry: coordinated evolution of ontologies to support biomedical data integration. *Nat Biotech*, 25(11):1251–1255, nov 2007. ISSN 1087-0156. URL http://dx.doi.org/10.1038/nbt1346. (Cited on page 200.)

J. F. Sowa. *Conceptual structures: information processing in mind and machine*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984. ISBN 0-201-14472-7. (Cited on pages 32, 74, 129 and 130.)

J. Sroka, J. Hidders, P. Missier, and C. Goble. A formal semantics for the taverna 2 workflow model. *Journal of Computer and System Sciences*, 76(6):490 – 508, 2010. ISSN 0022-0000. doi: 10.1016/j.jcss.2009.11.009. URL http://www.sciencedirect.com/science/article/pii/S0022000009001251. (Cited on page 117.)

A. Stell, R. Sinnott, O. Ajayi, and J. Jiang. Security oriented e-infrastructures supporting neurological research and clinical trials. In *Proceedings of the The Second International Conference on Availability, Reliability and Security*, pages 629–636, Washington, DC, USA, 2007. IEEE Computer Society. ISBN 0-7695-2775-2. doi: 10.1109/ARES.2007.137. URL http://dl.acm.org/citation.cfm?id=1249254.1250576. (Cited on page 60.)

M. Stocker and M. Smith. Owlgres: A scalable owl reasoner. In *OWLED*, 2008. (Cited on page 36.)

M. Stocker, A. Seaborne, A. Bernstein, C. Kiefer, and D. Reynolds. Sparql basic graph pattern optimization using selectivity estimation. In *Proceedings of the 17th international conference on World Wide Web*, WWW '08, pages 595–604, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-085-2. doi: http://doi.acm.org/10.1145/1367497.1367578. URL http://doi.acm.org/10.1145/1367497.1367578. (Cited on page 45.)

V. Stodden. The Scientific Method in Practice: Reproducibility in the Computational Sciences. Technical Report Working Paper 4773-10, MIT Sloan School of Management, Feb. 2010. URL http://ssrn.com/abstract=1550193. (Cited on page 57.)

R. Studer, V. R. Benjamins, and D. Fensel. Knowledge engineering: principles and methods. *Data Knowl. Eng.*, 25(1-2):161–197, Mar. 1998. ISSN 0169-023X. doi: 10.1016/S0169-023X(97)00056-6. URL http://dx.doi.org/10.1016/S0169-023X(97)00056-6. (Cited on page 3.)

M. Svihla and I. Jelínek. Benchmarking rdf production tools. In R. Wagner, N. Revell, and G. Pernul, editors, *DEXA*, volume 4653 of *Lecture Notes in Computer Science*, pages 700–709. Springer, 2007. ISBN 978-3-540-74467-2. (Cited on page 170.)

L. Temal, M. Dojat, G. Kassel, and B. Gibaud. Towards an ontology for sharing medical images and regions of interest in neuroimaging. *J. of Biomedical Informatics*, 41(5):766–778, 2008. ISSN 1532-0464. doi: http://dx.doi.org/10.1016/j.jbi.2008.03.002. (Cited on page 131.)

G. Tummarello, C. Morbidoni, R. Bachmann-Gmür, and O. Erling. Rdfsync: Efficient remote synchronization of rdf models. In *6th Int. Semantic Web Conf.*, volume 4825 of *LNCS*, pages 537–551, Busan, Korea, November 2007. Springer. ISBN 978-3-540-76297-3. (Cited on page 202.)

T. Vitvar, J. Kopecky, J. Viskova, and D. Fensel. WSMO-Lite Annotations for Web Services. In *5th European Semantic Web Conference (ESWC2008)*, pages 674–689, June 2008. URL http://data.semanticweb.org/conference/eswc/2008/paper/281. (Cited on page 109.)

W3C OWL Working Group. OWL 2 Web Ontology Language Document Overview [http://www.w3.org/TR/owl2-overview/], December 2012. URL http://www.w3.org/TR/owl2-overview/. (Cited on pages 29 and 32.)

D. Withers, E. Kawas, L. McCarthy, B. Vandervalk, and M. Wilkinson. Semantically-guided workflow construction in taverna: the sadi and biomoby plug-ins. In *Proceedings of the 4th international conference on Leveraging applications of formal methods, verification, and validation - Volume Part I*, ISoLA'10, pages 301–312, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-16557-5, 978-3-642-16557-3. URL http://portal.acm.org/citation.cfm?id=1939281.1939311. (Cited on page 131.)

K. Wolstencroft, P. Alper, D. Hull, C. Wroe, P. W. Lord, R. D. Stevens, and C. A. Goble. The mygrid ontology: bioinformatics service discovery. *IJBRA*, 3(3):303–325, 2007. (Cited on page 112.)

C. Wroe, C. A. Goble, A. Goderis, P. W. Lord, S. Miles, J. Papay, P. Alper, and L. Moreau. Recycling workflows and services through discovery and reuse. *Concurrency and Computation: Practice and Experience*, 19(2):181–194, 2007. (Cited on page 112.)

Distributed knowledge sharing and production through collaborative e-Science platforms

Abstract: This thesis addresses the issues of coherent distributed knowledge production and sharing in the Life-science area. In spite of the continuously increasing computing and storage capabilities of computing infrastructures, the management of massive scientific data through centralized approaches became inappropriate, for several reasons: (i) they do not guarantee the autonomy property of data providers, constrained, for either ethical or legal concerns, to keep the control over the data they host, (ii) they do not scale and adapt to the massive scientific data produced through e-Science platforms.

In the context of the NeuroLOG and VIP Life-science collaborative platforms, we address on one hand, distribution and heterogeneity issues underlying, possibly sensitive, resource sharing ; and on the other hand, automated knowledge production through the usage of these e-Science platforms, to ease the exploitation of the massively produced scientific data. We rely on an ontological approach for knowledge modeling and propose, based on Semantic Web technologies, to (i) extend these platforms with efficient, static and dynamic, transparent federated semantic querying strategies, and (ii) to extend their data processing environment, from both provenance information captured at run-time and domain-specific inference rules, to automate the semantic annotation of "in silico" experiment results.

The results of this thesis have been evaluated on the Grid'5000 distributed and controlled infrastructure. They contribute to addressing three of the main challenging issues faced in the area of computational science platforms through (i) a model for secured collaborations and a distributed access control strategy allowing for the setup of multi-centric studies while still considering competitive activities, (ii) semantic experiment summaries, meaningful from the end-user perspective, aimed at easing the navigation into massive scientific data resulting from large-scale experimental campaigns, and (iii) efficient distributed querying and reasoning strategies, relying on Semantic Web standards, aimed at sharing capitalized knowledge and providing connectivity towards the Web of Linked Data.

Keywords: Scientific workflows, Semantic web services, Provenance, Semantic web, Web of Linked Data, Federated knowledge bases, Distributed data integration, e-Science, e-Health.

Partage et production de connaissances distribuées dans des plateformes scientifiques collaboratives

Résumé: Cette thèse s'intéresse à la production et au partage cohérent de connaissances distribuées dans le domaine des sciences de la vie. Malgré l'augmentation constante des capacités de stockage et de calcul des infrastructures informatiques, les approches centralisées pour la gestion de grandes masses de données scientifiques multi-sources deviennent inadaptées pour plusieurs raisons: (i) elles ne garantissent pas l'autonomie des fournisseurs de données qui doivent conserver un certain contrôle sur les données hébergées pour des raisons éthiques et/ou juridiques, (ii) elles ne permettent pas d'envisager le passage à l'échelle des plateformes en sciences computationnelles qui sont la source de productions massives de données scientifiques.

Nous nous intéressons, dans le contexte des plateformes collaboratives en sciences de la vie NeuroLOG et VIP, d'une part, aux problématiques de distribution et d'hétérogénéité sous-jacentes au partage de ressources, potentiellement sensibles ; et d'autre part, à la production automatique de connaissances au cours de l'usage de ces plateformes, afin de faciliter l'exploitation de la masse de données produites. Nous nous appuyons sur une approche ontologique pour la modélisation des connaissances et proposons à partir des technologies du web sémantique (i) d'étendre ces plateformes avec des stratégies efficaces, statiques et dynamiques, d'interrogations sémantiques fédérées et (ii) d'étendre leur environnent de traitement de données pour automatiser l'annotation sémantique des résultats d'expérience "in silico", à partir de la capture d'informations de provenance à l'exécution et de règles d'inférence spécifiques au domaine.

Les résultats de cette thèse, évalués sur l'infrastructure distribuée et contrôlée Grid'5000, apportent des éléments de réponse à trois enjeux majeurs des plateformes collaboratives en sciences computationnelles : (i) un modèle de collaborations sécurisées et une stratégie de contrôle d'accès distribué pour permettre la mise en place d'études multi-centriques dans un environnement compétitif, (ii) des résumés sémantiques d'expérience qui font sens pour l'utilisateur pour faciliter la navigation dans la masse de données produites lors de campagnes expérimentales, et (iii) des stratégies efficaces d'interrogation et de raisonnement fédérés, via les standards du Web Sémantique, pour partager les connaissances capitalisées dans ces plateformes et les ouvrir potentiellement sur le Web de données.

Mots-clés: Flots de services et de données scientifiques, Services web sémantiques, Provenance, Web de données, Web sémantique, Fédération de bases de connaissances, Intégration de données distribuées, e-Sciences, e-Santé.