HAL open science

Protecting user data in distributed systems

Patricia Serrano-Alvarado

To cite this version:

Patricia Serrano-Alvarado. Protecting user data in distributed systems. Cryptography and Security [cs.CR]. Université de Nantes (UN), 2020. tel-02942581

HAL Id: tel-02942581

https://theses.hal.science/tel-02942581v1

Submitted on 18 Sep 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution - ShareAlike 4.0 International License

Habilitation à Diriger des Recherches

Patricia Serrano Alvarado

Université de Nantes
Comue Université Bretagne Loire
École Doctorale N°601
Mathématiques et Sciences et Technologies de l'Information et de la Communication
Spécialité : Informatique

Protecting user data in distributed systems

Habilitation présentée et soutenue à Nantes, le 16 juin 2020
Unité de recherche : Laboratoire des Sciences du Numérique de Nantes

Composition du Jury

Présidente :	**Mme Pascale Kuntz Cosperec**	Professeure, Université de Nantes
Rapporteurs :	**Mr. Luc Bouganim**	Directeur de Recherches, Inria Saclay-Île de France
	Mr. Sébastien Gambs	Professeur, Université du Québec à Montréal
	Mr. Daniel Le Metayer	Directeur de Recherches, Inria Alpes
Examinatrices :	**Mme Catherine Faron Zucker**	Maîtresse de Conférences HDR, Université Côte d'Azur
	Mme Marie-Christine Rousset	Professeure, Université Grenoble Alpes

To Biani and Emilio

Acknowledgements

I express my gratitude to the jury members who honored me to review this manuscript. Thank you for your time and valuable comments.

This document covers most of my research contributions since I was hired as Associate Professor at the University of Nantes. I warmly thank the PhD students, Post-docs, engineers and master students that I had the chance to work with, without naming them all, Mohamed Jawad, Nagham Alhadad, Georges Nassopoulos, Valeria Soto Mendoza, Marco Biazzini, Raziel Carvajal Gomez, Alejandra Perez Espinoza, Sara El Hassad, and last but not least Benjamin Moreau. Special thanks to my collegues with whom I supervised their works, Patrick Valduriez, Philippe Lamarre, Yann Busnel, Pascal Molli, Emmanuel Desmontils, Matthieu Perrin, José-Antonio García Macías, Elizabeth Pérez Cortés and Alban Gaignard.

Friendly thanks to my colleagues of the GDD team (past and present), the LS2N laboratory, and the Computer Sciences department with whom I share nice scientific but also friendly moments. Special thanks to Damien Eveillard, the first who motivate me to pass this Habilitation.

Several women encouraged me. I warmly thank Claudia Roncancio, my PhD supervisor, who made me want to have an academic career in France. You are still my best coauthor, according to dblp and hal. Hearty thanks to Pascale Kuntz, who pushed me to defend this Habilitation. Thanks to my friends and colleagues, Mariemad, Dalila, Salima, and Annie, for their encouragement for 14 years now!

Thanks to my friends of the COMAL association, we all know that sharing Mexican moments is very important for all of us. We are a big Franco-Mexican family!

My lovely thanks to my family for their unfailing support. My parents, Francisco and Maria Teresa, my brother and sister, José and Brenda, and my *comadres* and *compadres* (who are also my sisters in law and brothers in law) Avelina, Sabrina, Thierry, and David. All my tenderness to my nieces and nephews, José Salvador, José Franciso, Helena, José Sébastien, Léna, Maxime, and Victoria.

My love and gratitude to Andrée and Michel, who are always ready to help. Thanks Andrée, for fostering me up.

Finally, all my love for Stéphane, since twenty years ago, you are my first reader, who listens to my scientific stories (sometimes half listening). You ask the questions that I had not thought about. Thank you for your love and support.

Contents

Page

1 Introduction **1**

2 Modelizing and evaluating trust in systems **5**
2.1 Introduction 5
2.2 SocioPath: modeling a system 6
2.3 SocioTrust: evaluating trust in a system for an activity using probability theory 17
2.4 SubjectiveTrust: evaluating trust in a system for an activity using subjective logic 22
2.5 Conclusion 31

3 Supporting data privacy and measuring satisfaction in P2P systems **33**
3.1 Introduction 33
3.2 Hippocratic databases in P2P systems 34
3.3 WUW: a distributed P2P framework to enhance users' satisfaction 46
3.4 Conclusion 54

4 Deducing BGPs from logs of Linked Data providers **57**
4.1 Introduction 57
4.2 FETA: a reverse function 59
4.3 SWEEP, BGP deduction from a streaming log of TPF servers 69
4.4 Conclusion 72

5 Combination and classification of privacy policies **75**
5.1 Introduction 75
5.2 PrODUCE policies composition based on data usage context 76
5.3 CaLi: a lattice-based model for license classifications 86
5.4 Conclusion 95

6 On-going work and perspectives **97**

List of Figures **101**

List of Tables **103**

Bibliography **105**

Chapter 1

Introduction

Before joining the LINA laboratory, which became LS2N, my background was a PhD about Distributed Databases where I proposed a transactional model for mobile environments, under the supervision of Claudia Roncancio and Michel Adiba at the LSR laboratory, now LIG. During a post-doctoral position at LIFL laboratory, under the supervision of Philippe Merle, I proposed a functional decomposition of transactional systems to make them adaptable and reconfigurable.

I joined the GDD team, led by Patrick Valduriez at LINA in September 2005. Since then, my research activities have been developed in the field of large scale distributed systems. In particular, I have been interested in how to protect user data in such a context.

Protecting user data in distributed systems

Thanks to advances in communications and computer devices, users are able to access computer systems anytime and anywhere, either in the professional or private context. These systems, often opaque, manage and control user data in their behalf. They can combine user data with other data to deduce new knowledge, keep user data indefinitely (in servers which can be remote), share user data with other services and use them for purposes other than those originally consented by the user. These systems, make users agree privacy policies where they lose control over the use of their data. Users are forced to trust systems entirely, they can not negotiate policies, they can not verify whether they are preserved or not.

In my research work, I am interested in empowering the owners of the data, the ones who produce or are concerned by the data. I argue that data owners must keep control on the use of their data. For this, they must have the capability to express their preferences on the processing of their data, the system must make everything to respect their preferences and be able to prove that owners' preferences have been respected.

European laws and directives go in this direction, in particular, Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016[1], applicable from 25 May 2018. It states, among others, that (a) users must in principle give their consent for the processing of their data (b) data controllers must implement all the technical and organizational measures necessary to respect the protection of personal data, both from

[1] https://www.cnil.fr/fr/reglement-europeen-protection-donnees

the service design and by default (*privacy by design*), and (c) those responsible processors and subcontractors will have to put in place appropriate data protection measures and demonstrate this compliance at any time (*accountability*).

Protecting user data in distributed systems nowadays is very difficult. The challenge is that even when data control is distributed, data owner preferences must be preserved and laws respected by all parties. In my research, I am concerned about various issues related to the protection of user data on distributed data management systems, whose architectures range from client-server architectures to federations of servers or huge peer-to-peer organizations.

Research issues

Convinced that a major concern in our digital society is the protection of user data, in my research I am interested in 3 complementary issues: (a) how to allow data owners to chose the most trustful data management system, (b) how to build distributed data management systems that by default respect the preferences of users (privacy by design) and (c) how to do a verification of the data processing of systems showing compliance with privacy policies and licenses established on the data usage (accountability).

These three issues are the thread of my research that this document synthesizes throughout a 14-year period. I mainly tackled four challenges:

- **Challenge 1.** How to model and evaluate trust in a system for a particular digital activity.
- **Challenge 2.** How to store and share data on Peer-to-Peer (P2P) distributed systems (structured or not), taking into account user preferences.
- **Challenge 3.** In the context of the Linked Data, which is a completely distributed environment (without structure or centralized knowledge), how to verify the use of data.
- **Challenge 4.** How to automatically combine and position a license (or privacy policy) over a set of licenses, in terms of compatibility and compliance.

Organization of this document

In the following, I introduce my main contributions to these 4 challenges. A chapter is devoted to the contributions to each challenge.

Chapter 2 presents my contributions about trust in a system (Challenge 1). First I present SocioPath, a metamodel based on first order logic, that allows to model a system considering entities of the social and digital worlds and their relations. Then I present two approaches to evaluate trust in systems, namely, SocioTrust and SubjectiveTrust. The former is based on probability theory to evaluate users' trust in systems for a given activity. The latter is based on subjective logic to take into account uncertainty in trust values.

This chapter is mainly based on this journal article:

- Nagham Alhadad, Patricia Serrano Alvarado, Yann Busnel, and Philippe Lamarre. *System Modeling and Trust Evaluation of Distributed Systems*. In International Journal Large-Scale Data-and Knowledge-Centered Systems (TLDKS), LNCS 9430, ISBN 978-3-662-48566-8, 2015.
  https://hal.archives-ouvertes.fr/hal-01166896.

Chapter 3 presents my contributions about data privacy in P2P systems (Challenge 2). A first contribution is a privacy model for P2P systems and its implementation. The PriMod model allows data owners to specify their privacy preferences in privacy policies and to associate them with their data. PriServ, a privacy service located on top of DHT-based P2P systems, implements PriMod to prevent data privacy violations. Among others, PriServ uses trust techniques to predict peers behavior. I present a second contribution, WUW (What Users Want), a framework to measure and improve the satisfaction of the users based on personal preferences that reflect their expectations from the P2P system. I then present the design of a distributed P2P service that implements this framework.

This chapter is mainly based on a book chapter and an conference article:

- Mohamed Jawad, Patricia Serrano-Alvarado, Patrick Valduriez. *Supporting Data Privacy in P2P Systems*. Chapter on "Security and Privacy Preserving in Social Networks" book published by Springer ISBN 978-3-7091-0893-2, August 2013. http://hal.archives-ouvertes.fr/hal-00807625.

- Marco Biazinni, Patricia Serrano-Alvarado and Raziel Carvajal-Gomez. *Towards improving user satisfaction in decentralized P2P networks*. International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom), Austin, Texas, USA, October 2013.
  http://hal.archives-ouvertes.fr/hal-00871672.

Chapter 4 presents my contributions to the verification of data access in the Linked Data (Challenge 3). In data management, one way to check how the data is used is by having knowledge of queries executed by the system. In the Linked Data, knowing the queries evaluated is not easy due to the decentralization of data servers, their autonomy and choices in behalf of performance. My contributions are two approaches for deducing BGPs of SPARQL queries in two different contexts: FETA and SWEEP.

This chapter is mainly based on two conference articles:

- Georges Nassopoulos, Patricia Serrano-Alvarado, Pascal Molli, Emmanuel Desmontils. *FETA: Federated QuEry TrAcking for Linked Data*. In International Conference on Database and Expert Systems Applications (DEXA), Porto, Portugal, September 2016. https://hal.archives-ouvertes.fr/hal-01336386.

- Emmanuel Desmontils, Patricia Serrano-Alvarado, Pascal Molli. *SWEEP: a Streaming Web Service to Deduce Basic Graph Patterns from Triple Pattern Fragments*. In International Semantic Web Conference (ISWC), Demonstration paper. October 2017, Vienna, Australia. 2017.
  https://hal.archives-ouvertes.fr/hal-01583513.

Chapter 5 presents my contributions on the combination and classification of privacy policies and licenses (Challenge 4). My first contribution is a mechanism, named PrODUCE, that is based on semantic web technologies, and that allows to compose privacy policies. This approach proposes composition rules that are based on the data usage context and deduce implicit terms of policies. My second contribution is CaLi, a lattice-based model for license classifications. This model allows to position licenses in terms of compatibility and compliance.

This chapter is mainly based on two conference articles:

- Valeria Soto-Mendoza, Patricia Serrano-Alvarado, Emmanuel Desmontils, José - Antonio García-Macías. *Policies Composition Based on Data Usage Context*. In International Workshop on Consuming Linked Data (COLD) at ISWC, Bethlehem, Pennsylvania, United States, October 2015. https://hal.archives-ouvertes.fr/hal-01184660.

- Benjamin Moreau, Patricia Serrano-Alvarado, Matthieu Perrin, Emmanuel Desmontils. *Modelling the Compatibility of Licenses*. In Extended Semantic Web Conference (ESWC), Portoroz, Slovenia, June 2019. https://hal.archives-ouvertes.fr/hal-01816451

These works come mainly from the co-supervision of the PhD thesis of Mohamed Jawad, Nagham Alhadad, Georges Nassopoulos and Benjamin Moreau. As well as the supervision of the post-doctoral work of Marco Biazinni and the research internship of Valeria Soto Mendoza.

Chapter 2

Modelizing and evaluating trust in systems

Contents

2.1 Introduction **5**

2.2 SocioPath: modeling a system **6**

2.2.1 SocioPath metamodel 6

2.2.2 SocioPath definitions 10

2.2.3 Use case of a SocioPath model: GoogleDocs 13

2.3 SocioTrust: evaluating trust in a system for an activity using probability theory **17**

2.3.1 A SocioPath model as a directed acyclic graph (DAG) . . . 17

2.3.2 A probabilistic approach to infer system trust value 18

2.3.3 Experimental evaluations 20

2.4 SubjectiveTrust: evaluating trust in a system for an activity using subjective logic **22**

2.4.1 Inferring user's opinion on a system using subjective logic . . 23

2.4.2 Experimental evaluation 26

2.5 Conclusion **31**

2.1 Introduction

Nowadays, distributed systems are connected through complex architectures. These systems involve persons, physical and digital resources such that we can consider that a system consists of elements from two worlds, the social world and the digital world, and their relations. Users perform activities like chatting, buying, sharing data, *etc.* Evaluating and choosing appropriate systems involve aspects like functionality, performance, QoS, ease of use, or price. Recently, trust appeared as another key factor for such an evaluation. In this context, we raise two issues, (i) how to formalize the entities that compose a system

and their relations for a particular digital activity? and (ii) how to evaluate trust in a system for this activity? This chapter adresses both questions. In Section 2.2, I propose an answer for the first question with SocioPath, a metamodel that allows to model systems for a digital activity. This metamodel formalizes the entities in a system for an activity and the relations between them. To answer the second question, in Section 2.3, I propose SocioTrust, an approach to evaluate trust in a system for an activity that uses probability theory. And in Section 2.4, I propose a second approach, SubjectiveTrust, where I use subjective logic to take into account uncertainty to evaluate trust. Finally, I conclude in Section 2.5.

Contributions of this chapter have been published in [7, 8, 9, 10, 11, 12, 13, 14].

2.2 SocioPath: modeling a system

The most widespread architectures belong to the domain of distributed systems. Most of participants' activities on these systems concern their data (sharing and editing documents, publishing photos, purchasing online, *etc.*). As mentioned above, using these systems implies some implicit and explicit relationships, which may be partly unknown. With SocioPath [10], we aim to answer the following user's questions about her system:

Q1 Who are the persons that have a possibility to access a user's data? And what are the potential coalitions among persons that could allow undesired access to this data?

Q2 Who are the person(s)/resource(s) a user depends on to perform an activity?

Q3 Who are the persons that can prevent a user from performing an activity?

Q4 Who are the persons that a user is able to avoid to perform an activity?

Interesting approaches exist in the domain of Enterprise Architecture (EA). EA Frameworks provide methods and tools that allow to produce enterprise models. Two of the most used are The Open Group Architecture Framework (TOGAF) [43, 63] and the OBASHI Business & IT methodology and framework [112]. In general, what mainly distinguishes SocioPath from EA, is the social world that focuses on the persons who participate to the system. Instead of the social world, EA presents the business layer, which is mainly introduced by the component organization or organization unit. Hence, the analysis of the information in SocioPath focuses on the needs of the person who uses a system including her social, digital and physical dependences. Whereas, in EA, the analysis of the information focuses on the needs of an enterprise including ameliorating its performance, choosing the best person for a particular task, *etc.*

2.2.1 SocioPath metamodel

The SocioPath metamodel allows to describe the architecture of a system in terms of the components that enable people to access digital resources. It distinguishes two worlds; the *social world* and the *digital world*. In the social world, persons or organizations own any kind of physical resources and data. In the digital world, instances of data (including source

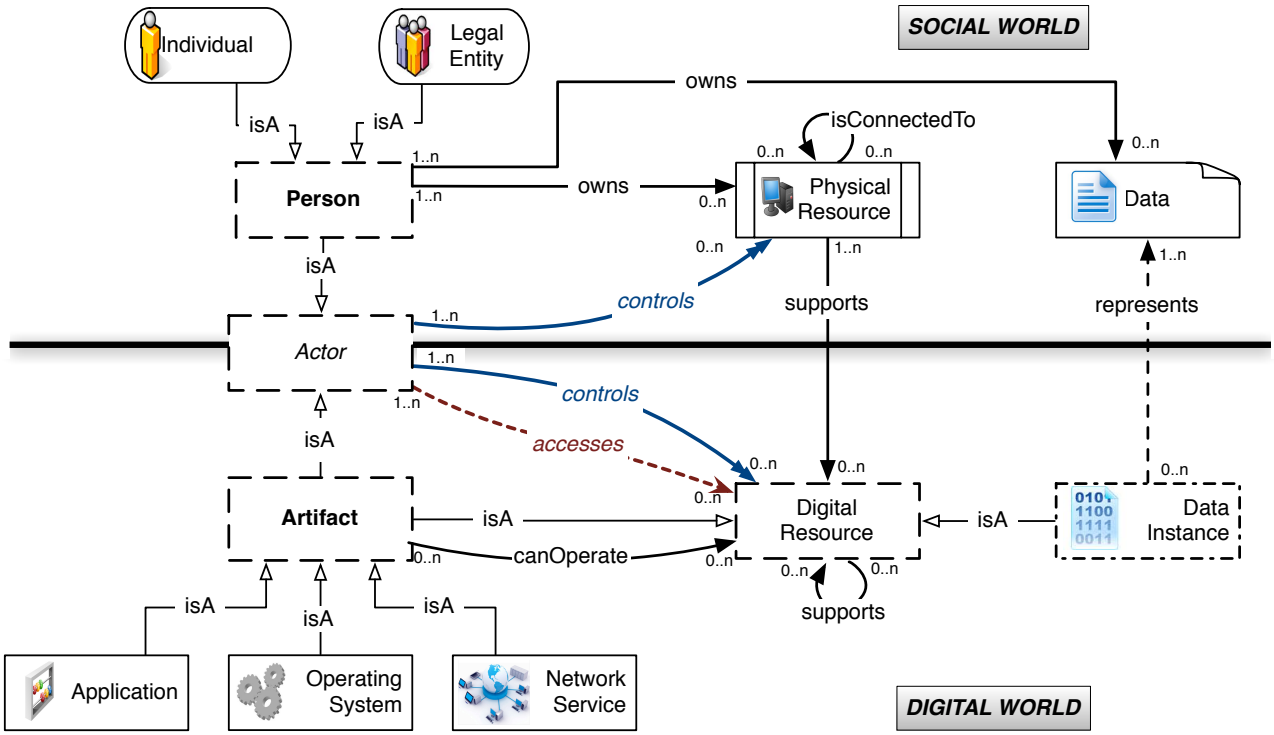


Figure 2.1: Graphical view of SocioPath as a UML class diagram.

codes) are stored and processes are running. Figure 2.1 shows the graphical representation of SocioPath, that we analyze in the next.

The social world includes persons (*e.g.*, users, enterprises, companies), physical resources, data, and relations among them.

- *Person* represents a generic notion that defines an individual like Alice or a Legal Entity like Microsoft.
- *Physical Resource* represents any hardware device (*e.g.*, PC, USB device, network infrastructure).
- *Data* represent an abstract notion that exists in real life, and does not necessarily imply a physical instance (*e.g.*, address, age, software design).

The digital world has entities that are defined as follows:

- *Data Instance* is a digital representation of a *Data* that exists in the social world. A source code is also a *Data Instance* implementing a software (text editor, mailer...) in the digital world.
- *Artifact* represents an abstract notion that describes a "running software". This can be an *Application*, an *Operating System* or a *Network Service*. It may be a single process or a group of processes that should be distributed on different locations, yet defining a single logically coherent entity.

Concept	Notation	Set	Remark
Actor	A	$\mathbb{A}$	$A \in \mathbb{A}$
Artifact	F	$\mathbb{F}$	$F \in \mathbb{F}$
Digital resource	DR	$\mathbb{DR}$	$DR \in \mathbb{DR}$
Physical resource	PR	$\mathbb{PR}$	$PR \in \mathbb{PR}$
Data	D	$\mathbb{D}$	$D \in \mathbb{D}$
Data instance	DI	$\mathbb{DI}$	$DI \in \mathbb{DI}$
Operating system	OS	$\mathbb{OS}$	$OS \in \mathbb{OS}$
Path	σ	Υ	$\sigma \in \Upsilon$
Architecture or system	α	Λ	$\alpha \in \Lambda$
Activity	ω	$\mathbb{W}$	$\omega \in \mathbb{W}$
Activity path	σ^{ω}	Υ^{ω}	$\sigma^{\omega} \in \Upsilon^{\omega}$
Activity minimal path	$\widehat{\sigma^{\omega}}$	$\widehat{\Upsilon^{\omega}}$	$\widehat{\sigma^{\omega}} \in \widehat{\Upsilon^{\omega}}$
Set of activity restrictions	$\mathcal{S}$	$\mathbb{S}$	$\mathcal{S} \in \mathbb{S}$
Person or user	P	$\mathbb{P}$	$P \in \mathbb{P}$

Table 2.1: Glossary of notations (1).

- *Digital Resource* represents an *Artifact* or a *Data Instance*.
- *Actor* represents a *Person* in the social world or an *Artifact* in the digital world. This is the core concept of SocioPath. Indeed, only *Actors* can access or control *Digital Resources* as presented below.

Table 2.1 summarizes the notations we use in the following.

The relations of SocioPath represent in a non naive way how a system is built. They should help to highlight the links between the structure of a system and the confidence of a user within this system.

- *owns* is a relation of ownership between a *Person* and a *Physical Resource* (*owns(P, PR)*), or between a *Person* and some *Data* (*owns(P,D)*). This relation only exists in the social world.
- *isConnectedTo* is a relation of connection between two *Physical Resources* (*isConnectedTo* (PR_1, PR_2)). It means that two entities are physically connected, through a network for instance. This symmetric relation exists only in the social world.
- *canOperate* represents an *Artifact* that is able to process, communicate or interact correctly with a target *Digital Resource* (*canOperate(F,DR)*).
- *accesses* represent an *Actor* that can access a *Digital Resource* (*accesses(A, DR)*).
- *controls* represents an *Actor* that can control a *Digital Resource* (*controls(A, DR)*). There should exist different kinds of control relations. For instance, a legal entity, who provides a resource, controls the functionalities of this resource. The persons who use this resource may have some kind of control on it as well. Each of these actors controls the resource in a different way.

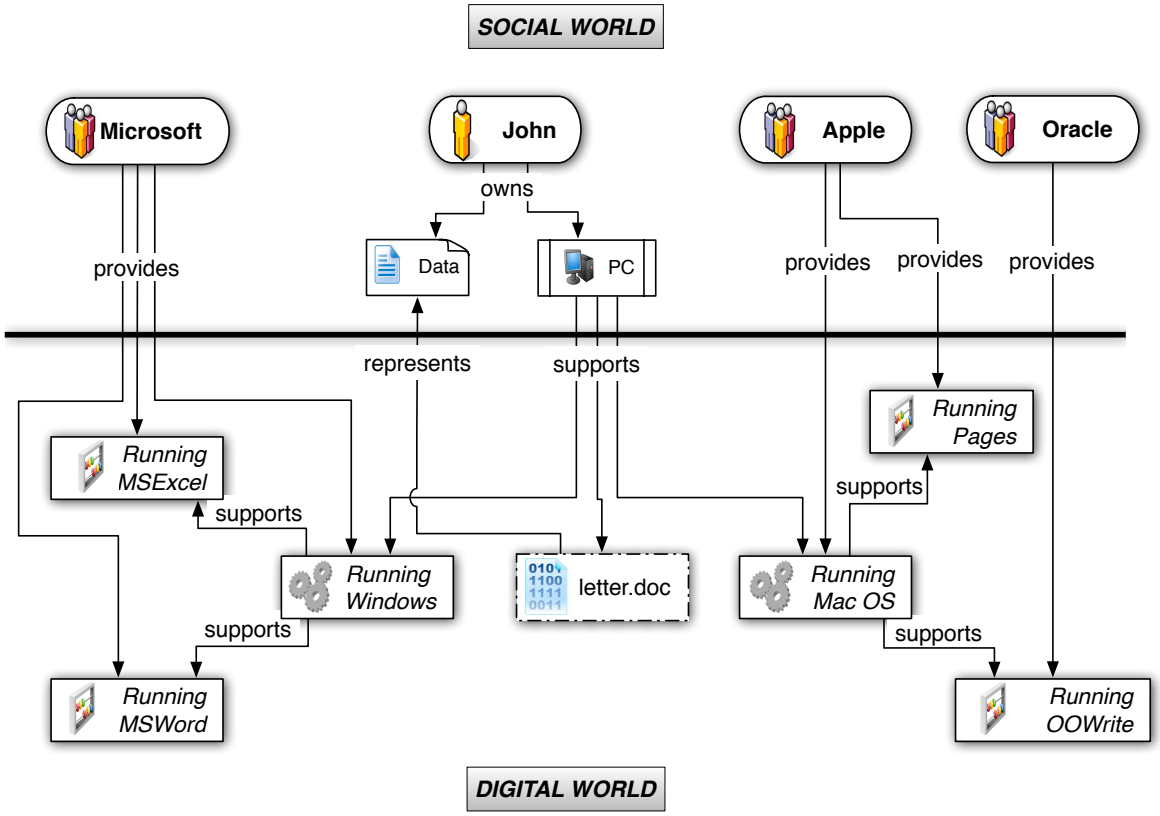


Figure 2.2: Use case 1: isolated PC.

- *supports* is a relation between two *Digital Resources* (*supports*(DR_1, DR_2)), or a *Physical Resource* and a *Digital Resource* (*supports*(PR, DR)). It means that the target entity could never exist without the source entity. We may say that the latter allows the former to exist.

- *represents* is a relation between *Data* in the social world and their *Instances* in the digital world (*represents*(D,DI)).

For sake of simplicity, we consider that a person *provides* an artifact, *if this person owns the data represented by the data instance which supports the artifact.*

Applying SocioPath makes possible non-trivial deductions about relations among entities. For instance, an actor may be able to access digital resources supported by different physical resources connected to each other (*e.g.*, a user can access processes running in different hosts).

Deduced access and control relations. The semantics of the components and the relations of a SocioPath model allow to deduce more *control* and *access* relations. We use, a first order logic to describe the rules allowing such deductions.

The proposed deduction rules of SocioPath are not exhaustive and by no means we pretend they capture the whole complexity of systems. They capture several aspects of a simplified vision of the systems that serves the purpose of building an understandable and expressive model. Table 2.2 shows these rules.

Rule	Formal definition
Rule 1	$\forall F \in \mathbb{F}, \forall DR \in \mathbb{DR}, \forall PR1, PR2 \in \mathbb{PR}: \bigwedge \begin{cases} canOperate(F, DR) \\ supports(PR1, F) \\ \bigvee \begin{cases} supports(PR1, DR) \\ \bigwedge \begin{cases} supports(PR2, DR) \\ isConnectedTo(PR1, PR2) \end{cases} \end{cases} \end{cases} \Rightarrow accesses(F, DR)$
Rule 2	$\forall P \in \mathbb{P}, \forall PR \in \mathbb{PR}, \forall OS \in \mathbb{OS}: \bigwedge \begin{cases} owns(P, PR) \\ supports(PR, OS) \end{cases} \Rightarrow \bigwedge \begin{cases} accesses(P, OS) \\ controls(P, OS) \end{cases}$
Rule 3	$\forall F \in \mathbb{F}, \forall OS \in \mathbb{OS}: \bigwedge \begin{cases} supports(OS, F) \\ canOperate(OS, F) \end{cases} \Rightarrow controls(OS, F)$
Rule 4	$\exists P \in \mathbb{P}, \exists D \in \mathbb{D}, \exists DI \in \mathbb{DI}, \exists F \in \mathbb{F}: \bigwedge \begin{cases} owns(P, D) \\ represents(DI, D) \\ supports(DI, F) \end{cases} \Rightarrow controls(P, F)$
Rule 5	$\forall A \in \mathbb{A}, \forall F \in \mathbb{F}, \forall DR \in \mathbb{DR}: \bigwedge \begin{cases} accesses(A, F) \\ accesses(F, DR) \end{cases} \Rightarrow accesses(A, DR)$
Rule 6	$\forall A \in \mathbb{A}, \forall F_1, F_2 \in \mathbb{F}: \bigwedge \begin{cases} controls(A, F_1) \\ controls(F_1, F_2) \end{cases} \Rightarrow controls(A, F_2)$
Rule 7	$\exists PR1, PR2 \in \mathbb{PR}, \exists OS1, OS2 \in \mathbb{OS}: \bigwedge \begin{cases} isConnectedTo(PR1, PR2) \\ supports(PR1, OS1) \\ supports(PR2, OS2) \end{cases} \Rightarrow accesses(OS1, OS2)$

Table 2.2: Deduced access and control relations.

- Rule 1 states that if an artifact *can operate* a digital resource and either the artifact and the digital resource are *supported* by the same physical resource or they are *supported* by connected physical resources, then the artifact *accesses* the digital resource.
- Rule 2 states that if a person *owns* a physical resource that *supports* an operating system, then the person *accesses* and *controls* this operating system.
- Rule 3 states that if an operating system *supports* and *can operate* an artifact, then it *controls* this artifact.
- Rule 4 states that if a person *owns* data represented in the digital world by a data instance which *supports* an artifact, then this person *controls* this artifact.
- Rule 5 states the transitivity of relation *accesses*.
- Rule 6 states the transitivity of relation *controls*.
- Rule 7 states that if two physical resources are *connected* to each other, and the first one *supports* an operating system and the second one *supports* another operating system, these two operating systems *access* to each other.

2.2.2 SocioPath definitions

We next enrich SocioPath with formal definitions to answer the motivating questions Q1 to Q4 (cf. page 6). Definitions concern activities, paths, and dependences. All of them can be automatically deduced from a SocioPath model.

Definitions for activities and paths. A SocioPath model expresses chains of access and control relations, *i.e.,* paths. A user follows a path to perform an activity in a system. In our analysis, we consider systems enabling users to perform a data-based activity. To do so, restrictions must be defined to impose the presence of particular elements in paths. For instance, if a person wants to read a .doc document, she must use an artifact that can "understand" this type of document (*e.g.*, MSWord or OOWrite). Another example, if a person uses a SVN application, the artifacts "SVN client" and "SVN server" should be used and they should appear in the correct order within the path (usually, the SVN client should precede the SVN server).

Definition 1 (Activity ω)**.**
We define an activity ω as a triple $(P, D, \mathcal{S})$, where P is a person, D is a datum and $\mathcal{S}$ is a set of ordered sets $\mathbb{F}$ in a model. So an activity ω is a subset of $\mathbb{P} \times \mathbb{D} \times \mathbb{S}$. The sets in the $\mathbb{S}$ component of an activity are alternative sets of artifacts to perform the activity, i.e., *each set allows the person to perform his activity. Thus, $\omega = (P, D, \mathcal{S}) \in \mathbb{P} \times \mathbb{D} \times \mathbb{S}$. For instance, the activity "John edits* letter.doc*", in use case 1, is defined as ω=*(John, Data, {{MSWord}, {Pages}, {OOWrite}}).

We call *paths* the lists of actors and digital resources describing the ways an actor may access a digital resource. A person may perform an activity in different ways and using different intermediate digital resources. Each possibility is described by a path.

Definition 2 (Activity path, or ω-path)**.**
A path σ for an activity $\omega = (P, D, \mathcal{S}) \in \mathbb{P} \times \mathbb{D} \times \mathbb{S}$ is a list of actors and digital resources such that:

- $\sigma[1] = P$;
- $\sigma[|\sigma|] = D$;
- $represents(\sigma[|\sigma| - 1], \sigma[|\sigma|])$;
- $\forall i \in [2 : |\sigma| - 1], (\sigma[i] \in \mathbb{F}) \wedge accesses(\sigma[i - 1], \sigma[i])$;
- $\exists s \in \mathcal{S}, s \subseteq \sigma$.

Where $\sigma[i]$, denotes the ith element of σ, and $|\sigma|$ the length of σ.

***Notation**: Assuming that there is no ambiguity on the model under consideration, the set of ω-paths where ω= $(P, D, \mathcal{S})$ is denoted Υ^{ω} and the set of all the paths in the model is denoted Υ.*

Activity paths may contain unnecessary artifacts. To eliminate them, we define the activity minimal paths as follows.

Definition 3 (Activity minimal path, or ω-minimal path)**.**
Let Υ^{ω} be a set of paths for an activity ω.
A path $\sigma^{\omega} \in \Upsilon^{\omega}$ is said to be minimal in Υ^{ω} iff *there exists no path $\sigma' \in \Upsilon^{\omega}$ such that:*

- $\sigma^{\omega}[1] = \sigma'[1]$ *and* ; $\sigma^{\omega}[|\sigma^{\omega}|] = \sigma'[|\sigma'|]$;
- $\forall i \in [2 : |\sigma'|], \exists j \in [2 : |\sigma^{\omega}|], \sigma'[i] = \sigma^{\omega}[j]$;

- $\forall i \in [2 : |\sigma'| - 1], accesses(\sigma'[i-1], \sigma'[i])$.

***Notation**: The set of minimal paths enabling an activity $\omega = (P, D, \mathcal{S})$ is denoted $\widehat{\Upsilon^{\omega}}$. This set represents also an architecture for an activity, denoted by α. For sake of simplicity, we name this set the ω-minimal paths.*

Definitions for dependences. Modeling systems with SocioPath allows to underline and discover chains of *accesses* and *controls* relations. In the following, we introduce the definitions of digital dependences (Definitions 4 and 5) and social dependences (Definitions 6 to 9). Informally, the sets of digital dependences of a person are composed of the artifacts a user passes by to reach a particular element. The sets of social dependences are composed of the persons who control these artifacts and the physical resources that support them. We call digital dependences the sets of artifacts a user depends on, because artifacts belong to the digital world in SocioPath. Similarly, we call social dependences the sets of persons and physical resources a user depends on, because they belong to the social world in SocioPath. In the following, these concepts are defined formally and examples refer to use case 1.

Digital dependences. We say that a person depends on a set of artifacts for an activity ω if each element of this set belongs to one or more paths in the set of the ω-minimal paths.

Definition 4 (Person's dependence on a set of artifacts for an activity)**.**
Let $\omega = (P, D, \mathcal{S})$ be an activity, $\mathcal{F}$ be a set of artifacts and $\widehat{\Upsilon^{\omega}}$ be the set of ω-minimal paths.

$$P \text{ depends } \textit{on } \mathcal{F} \textit{ for } \omega \text{ iff } \exists \mathcal{F} \subset \mathbb{F}, \forall F \in \mathcal{F}, \exists \sigma \in \widehat{\Upsilon^{\omega}} : F \in \sigma.$$

A person does not depend on all the sets of artifacts in the same way. Some sets may be avoidable because the activity can be executed without them. Some sets are unavoidable because the activity cannot be performed without them. To distinguish the way a person depends on artifacts, we define the degree of a person's dependence on a set of artifacts for an activity as the ratio of the ω-minimal paths that contain these artifacts to all the ω-minimal paths.

Definition 5 (Degree of a person dependence on a set of artifacts)**.**
Let $\omega = (P, D, \mathcal{S})$ be an activity, $\mathcal{F}$ be a set of artifacts and $\widehat{\Upsilon^{\omega}}$ be the set of ω-minimal paths and $|\widehat{\Upsilon^{\omega}}|$ is the number of the ω-minimal paths. The degree of dependence of P on $\mathcal{F}$, denoted $d^{\omega}_{\mathcal{F}}$, is:

$$d^{\omega}_{\mathcal{F}} = \frac{|\{\sigma : \sigma \in \widehat{\Upsilon^{\omega}} \wedge \exists F \in \mathcal{F}, F \in \sigma\}|}{|\widehat{\Upsilon^{\omega}}|}$$

Social dependences. From the digital dependences, we can deduce the social dependences as follows. A person depends on a set of persons for an activity if the persons in this set control some of the artifacts the person depends on.

Definition 6 (Person's dependence on a set of persons for an activity).
Let $\omega = (P, D, \mathcal{S})$ be an activity, and $\mathcal{P}$ a set of persons.

$$P \textit{ depends on } \mathcal{P} \textit{ for } \omega \text{ iff } \wedge \begin{cases} \exists \mathcal{F} \subset \mathbb{F} : P \textit{ depends on } \mathcal{F} \textit{ for } \omega \\ \forall F \in \mathcal{F}, \exists P' \in \mathcal{P} : controls(P', F) \end{cases}$$

The degree of a person's dependence on a set of persons for an activity is given by the ratio of the ω-minimal paths that contain artifacts controlled by this set of persons.

Definition 7 (Degree of a person's dependence on a set of persons).
Let $\omega = (P, D, \mathcal{S})$ be an activity, $\mathcal{P}$ be a set of persons and $\widehat{\Upsilon^\omega}$ be the ω-minimal paths. The degree of dependence of P on $\mathcal{P}$, denoted $d^\omega_{\mathcal{P}}$, is:

$$d^\omega_{\mathcal{P}} = \frac{|\{\sigma : \sigma \in \widehat{\Upsilon^\omega} \wedge \exists P' \in \mathcal{P}, \exists F \in \sigma, controls(P', F)\}|}{|\widehat{\Upsilon^\omega}|}$$

We say a person depends on a set of physical resources for an activity if the elements of this set support the artifacts the person depends on.

Definition 8 (Person's dependence on a set of physical resources).
Let $\omega = (P, D, \mathcal{S})$ be an activity, and $\mathcal{PR}$ be a set of physical resources.

$$P \textit{ depends on } \mathcal{PR} \textit{ for } \omega \text{ iff } \wedge \begin{cases} \exists \mathcal{F} \subset \mathbb{F} : P \textit{ depends on } \mathcal{F} \textit{ for } \omega \\ \forall F \in \mathcal{F}, \exists PR \in \mathcal{PR} : supports(PR, F) \end{cases}$$

The degree of a person's dependence on a set of physical resources for an activity is given by the ratio of the ω-minimal paths that contain artifacts supported by this set of physical resources.

Definition 9 (Degree of a person's dependence on a set of physical resources).
Let $\omega = (P, D, \mathcal{S})$ be an activity, let $\mathcal{PR}$ be a set of physical resources, let $\widehat{\Upsilon^\omega}$ be the ω-minimal paths. The degree of dependence of P on $\mathcal{PR}$, denoted $d^\omega_{\mathcal{PR}}$ is:

$$d^\omega_{\mathcal{PR}} = \frac{|\{\sigma : \sigma \in \widehat{\Upsilon^\omega} \wedge \exists PR \in \mathcal{PR}, \exists F \in \sigma, supports(PR, F)\}|}{|\widehat{\Upsilon^\omega}|}$$

These definitions allow awareness of the user's dependences on the digital and social world. Another use case is presented in the next section to illustrate them.

2.2.3 Use case of a SocioPath model: GoogleDocs

Figure 2.3 presents a SocioPath model where John uses GoogleDocs for the activity "John reads document.gtxt". In the social world, John owns some Data, a PC and an iPad. We explicitly name only some legal entities who provide resources and artifacts: Microsoft for Windows and Internet Explorer (so called IExplorer), Google for GoogleDocs and Google Cloud services, SkyFireLabs for SkyFire, Apple, for the iOS operating system and the browser Safari and Linux Providers for Linux.NeufTelecom, Orange and SFR are telecom companies. John's iPad is connected to SFR Servers and John's PC is connected to NeufTelecom Servers and Orange Servers. In the digital world, the operating systems Windows and Linux are

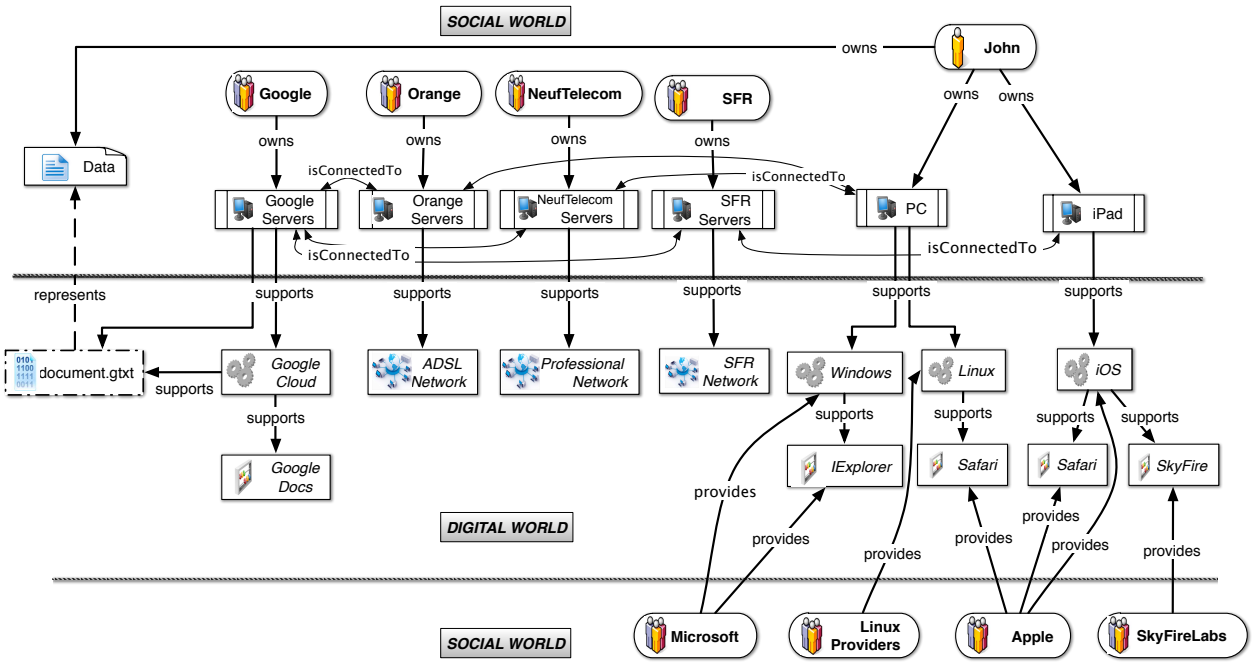


Figure 2.3: Use case 2: GoogleDocs.

running on John's PC. Windows supports IExplorer and Linux supports Safari. John's iPad supports the running iOS, which supports two applications, Safari and SkyFire. John's data are represented in the digital world by document.gtxt, which is supported by the physical resources owned by Google. We consider Google Cloud as the storage system used by Google Docs.

Analysis and results. Through this example we show that SocioPath provides answers to our motivating questions.

Q1 *Who are the persons that have a possibility to access John's data? And what are the potential coalitions among persons that could allow undesired access to this data?*

By applying the deduction rules presented in Section 2.2.1, we deduce the relations of *access* and *control* that exist in this architecture. They are illustrated in Figure 2.4.

By knowing the relations *accesses* in this model, *cf.* Figure 2.4 (a), John is able to know which persons have a possible path to his document. Thus, these persons can[1] access his data. In this example, they are: SFR, NeufTelecom, John, Orange, and Google.

Furthermore, by examining the persons who control the artifacts in the paths, *cf.* Figure 2.4 (b), it is possible to understand which coalitions may be done to access John's data. For example, Google can access document.gtxt directly because it controls all the artifacts of the path that enables it to reach it. Orange, instead, has a possible path to access John's data that passes through artifacts controlled by Google. So it must collude with Google to access John's data.

[1]By *can*, we mean that a user may be able to perform an action, and not that she has the permissions to do it. In this work, we do not analyze access control and user permission constraints.

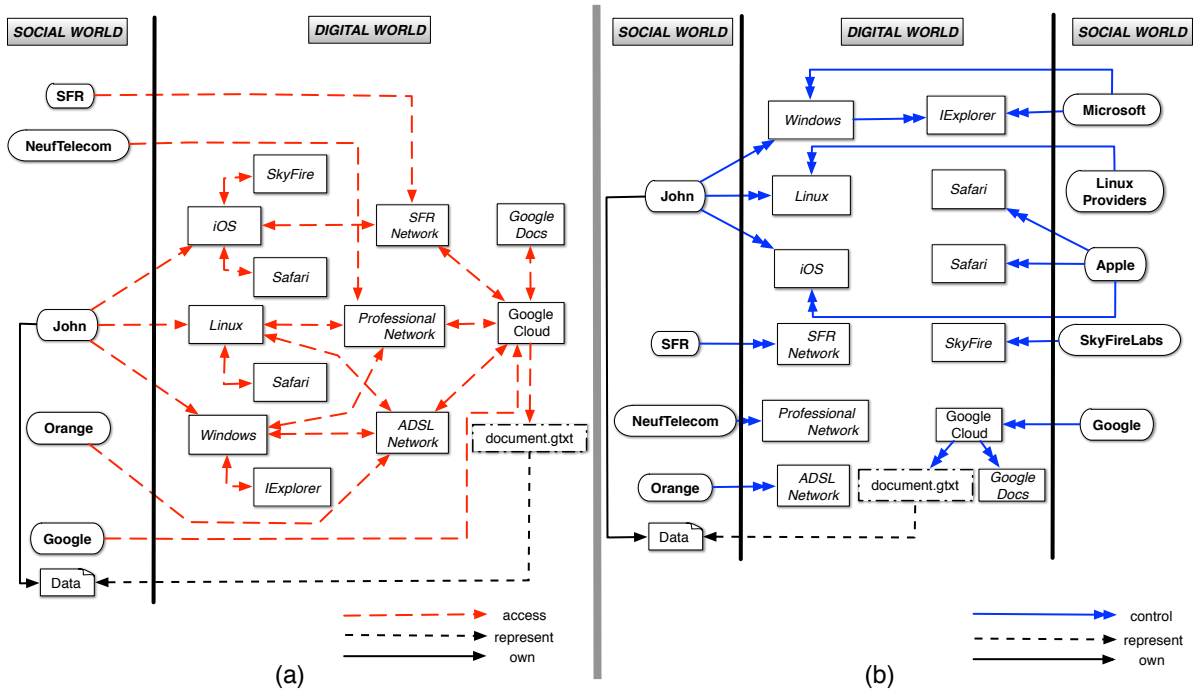


Figure 2.4: Relations of access and control in the use case 2 (GoogleDocs).

Q2 *Who are the person(s)/resource(s) John depends on to perform the activity "John reads* document.gtxt*"?*

If John wants to read document.gtxt, he needs a browser and GoogleDocs. So formally, we define this activity as ω=(John, Data, {{SkyFire, GoogleDocs}, {Safari, GoogleDocs}, {IExplorer, GoogleDocs}}). If we apply Definition 3, we find that John has six ω-minimal paths to read document.gtxt:

1. {John, Windows, IExplorer, Windows, ADSL Network, GoogleCloud, GoogleDocs, document.gtxt, Data};
2. {John, Windows, IExplorer, Windows, Professional Network, GoogleCloud, GoogleDocs, document.gtxt, Data};
3. {John, Linux, Safari, Linux, ADSL Network, GoogleCloud, GoogleDocs, document.gtxt, Data};
4. {John, Linux, Safari, Linux, Professional Network, GoogleCloud, GoogleDocs, document.gtxt, Data};
5. {John, iOS, SkyFire, iOS, SFR Network, GoogleCloud, GoogleDocs, document.gtxt, Data};
6. {John, iOS, Safari, iOS, SFR Network, GoogleCloud, GoogleDocs, document.gtxt, Data}.

By applying the definitions of Section 2.2.2, we obtain John's social and digital dependences, and the degree of these dependences for this activity. We show the results concerning some sets of persons John depends on in Table 2.3 and the degree of dependences on these sets in Figure 2.5. This information reveals how much John is autonomous from a specific person or a set of persons. For instance, the degree of

Group	Sets of persons John depends on	Group	Sets of persons John depends on
G1	{Microsoft}	G12	{Apple,Orange,NeufTelecom}
G2	{Linux Providers}	G13	{Microsoft,SkyFireLabs}
G3	{Apple}	G14	{Orange,SFR}
G4	{SkyFireLabs}	G15	{Apple,Orange}
G5	{SFR}	G16	{Microsoft,NeufTelecom}
G6	{NeufTelecom}	G17	{Microsoft,Orange}
G7	{Orange}	G18	{SkyFireLabs,NeufTelecom}
G8	{Google}	G19	{Microsoft,SFR,Linux Providers}
G9	{Microsoft,Apple}	G20	{Apple,NeufTelecom}
G10	{NeufTelecom,Orange,SFR}	G21	{Linux Providers,SkyFireLabs}
G11	{Linux Providers,SFR}		

Table 2.3: Sets of persons John depends on (use case 2 - GoogleDocs).

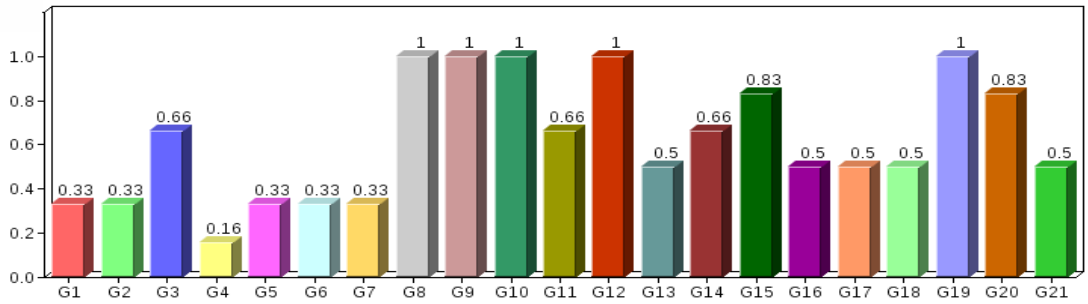


Figure 2.5: Degree of dependence on persons' sets.

dependence on {Microsoft} is 0.33, and the degree of dependence on the set {Apple, NeufTelecom} is 0.83.

Q3 *Who are the persons that can prevent John from performing the activity "John reads document.gtxt"?*

Sets having a degree of dependence equal to 1, are the persons who can prevent John from "reading document.gtxt" because they cross all the ω-paths of this model. These sets are: G8, G9, G10, G12, and G19.

Q4 *Who are the persons that John is able to avoid to perform the activity "John reads document.gtxt"?*

John depends on the sets on which the degree of dependence is less than one, in a less dramatic way (*e.g.*, on the set G8 with a degree of 0.5), because this shows that there are other minimal ω-paths enabling John to read document.gtxt and the persons who belong to this set do not control any artifact in these paths. These sets enlighten the "combinations of persons", which John is able to avoid at will.

SocioPath is then useful in the evaluation process of a system with respect to trust requirements. This leads to the fifth question presented in the introduction of this section, namely *How much a user trusts a system for a specific activity?* We focus on answering this question in the following sections.

2.3 SocioTrust: evaluating trust in a system for an activity using probability theory

Trust has been widely studied in several aspects of daily life. In the trust management community [62, 78, 88, 115, 118, 120], two main issues arise, *(i) how to define the trust in an entity, knowing that entities can be persons, digital and physical resources?* and *(ii) how to evaluate such a value of trust in a system under a particular context?* This second point embodies the main focus of this section.

We argue that studying trust in the separate entities that compose a system does not give a picture of how trustworthy a system is as a whole. Indeed, the trust in a system depends on its architecture, more precisely, on the way the implicit and explicit entities, which the users depend on to do their activities, are organized.

Inspired by this idea, we propose SocioTrust [14], an approach to evaluate trust in a system for an activity. The system definition is based on SocioPath models (*cf.* Section 2.2), which here are simplified to present the architecture of a system as a weighted directed acyclic graph (DAG). Levels of trust are then defined for each node in the graph according to the user who evaluates trust. By combining trust values using the theory of probability, we are able to estimate two different granularities of trust, namely, *trust in a path* and *trust in a system*, both for an activity to be performed by a person.

We begin this section introducing how to present a SocioPath model as a directed acyclic graph in Section 2.3.1. Section 2.3.2 introduces SocioTrust where the main problem we face for trust evaluation is the existence of *dependent paths*. Section 2.3.3, evaluates our contribution with several experiments that analyze the impact of different characteristics of a system on the behavior of the obtained trust values. Experiments realized on both synthetic traces and real datasets validate our approach.

2.3.1 A SocioPath model as a directed acyclic graph (DAG)

We simplify the representation of SocioPath models by aggregating one artifact, the set of persons controlling it, and the set of physical resources supporting it, into only one component. The resulting set of components are the nodes of the DAG and the edges are the *access* relations. A user performs an activity by browsing successive *access* relations through the graph, so-called through *activity minimal paths*.[2]

Definition 10 (A simplified system for an activity, α)**.**
A simplified system that enables a user to achieve an activity, can be expressed as a tuple $\alpha =< \mathbb{N}_\omega, \mathbb{A}_\omega >$ *where:*

- ω *represents the activity the user wants to achieve, as a triple* $(P, D, \mathcal{S})$ *(*cf. *Section 2.2.2).*

- $\mathbb{N}_\omega$ *represents the set of nodes* n *in a system for an activity such that* $\{P, D\} \subset \mathbb{N}_\omega$*, and each triple composed by one artifact, the persons who control it, and the physical*

[2]If there is no ambiguity, we denote an activity minimal path (*i.e.,* ω-minimal path) through the DAG simply by a path σ and each path does not consider the source and the target nodes, *i.e.,* the person and the data instance and the data.

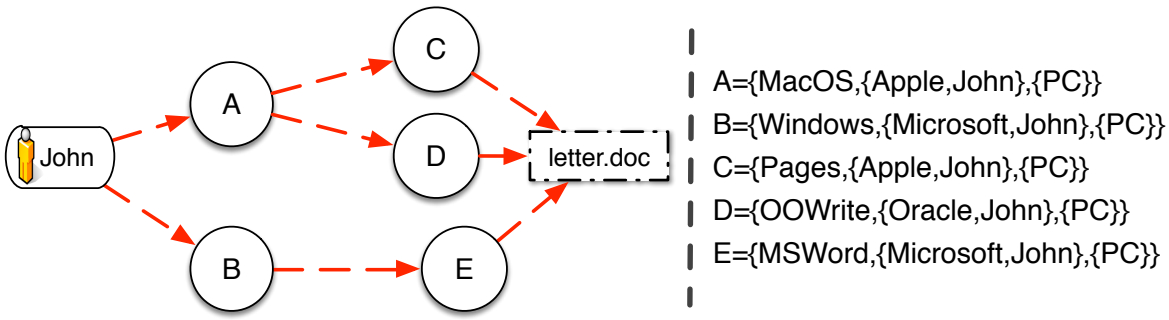


Figure 2.6: The system for the activity "John edits letter.doc" as a DAG of use case 1.

resources that support it, are aggregated into one single node, i.e., $n \in \mathbb{N}_\omega \setminus \{P, D\}$ *such that* $n \supseteq \{F, A, PR\}$ iff $controls(A, F) \wedge supports(PR, F)$.

- $\mathbb{A}_\omega \subseteq \mathbb{N}_\omega \times \mathbb{N}_\omega$ *represents the set of edges in a system. From the rules of* SocioPath *and the aggregation we made for a node, our DAG exhibits only the relation* access.

Graph-based trust approaches [5, 36, 60, 61, 76, 102], are especially used in social networks where the main idea of trust derivation is to propagate trust between two nodes in a graph that represents the social network. In [5], authors propose a general approach where they divide the process of trust evaluation into two steps: (i) trust combination through a path where the main idea is to combine the trust values among the intermediate edges of a path to obtain a trust value for this path and (ii) trust combination through a graph where the main idea is to combine the trust values of all the paths that relate the source with the target, to obtain a single trust value for the graph.

In [60, 61], Jøsang *et al.* raised a problem of graph-based trust approaches when there are dependent paths in a graph, *i.e.*, paths that have common edges in the graph. To face this problem, in our approach we use conditional probability.

2.3.2 A probabilistic approach to infer system trust value

If a user needs to evaluate her trust in a system for an activity, she associates each node in the DAG with a trust value and the DAG becomes a weighted directed acyclic graph (WDAG).

In this work, we adopt the definition of Jøsang *et al.* about trust [62]: *"trust is the probability by which an individual, A, expects that another individual, B, performs a given action on which its welfare depends"*.

Trust in a node for an activity. Trust in a node is evaluated from the point of view of the concerned user. There are several ways to construct this trust level. We can figure out different objective and subjective factors that impact this trust level, like the reputation of the persons who control the artifact, their skills, the performance of the physical resource that supports the artifact or the personal experience with this artifact. We thus have $t(N) = f(t^F_\omega, t^\mathcal{P}_\omega, t^{\mathcal{PR}}_\omega)$, where t^F_ω, $t^\mathcal{P}_\omega$, $t^{\mathcal{PR}}_\omega$ are respectively the trust values assigned to an artifact F, the set of persons $\mathcal{P}$ who control F, and the set of physical resources $\mathcal{PR}$ that supports F for a given activity ω. The meaning of the resulting trust value in a node

depends on the employed function f to compute this value [79]. For instance, if Bayesian inference is employed to evaluate it as is done in [74], the node trust value is considered as *"the probability by which a user believes that a node can perform an expected action for a given activity"* [34].
In this study, we do not address the issue of computing the trust value of a user in a node for an activity but we interpret it as the probability, by which a user P believes that a node N provides her the expected services for ω. Then, we have:

$$t(N) = \mathbb{P}(\lambda^N) \tag{2.1}$$

Trust in a path for an activity. A path in a system represents a way to achieve an activity. The trust level of a person P to achieve an activity through a particular path $\sigma = \{N_1, N_2, \ldots, N_n\}$ is the probability that all nodes $\{N_i\}_{i\in[1..n]}$ provide the expected services for the activity. Thus $\mathbb{P}(\lambda^\sigma)$ is computed as follows:

$$t(\sigma) = \mathbb{P}(\lambda^\sigma) = \mathbb{P}(\lambda^{N_1} \wedge \lambda^{N_2} \wedge \ldots \wedge \lambda^{N_n})$$

The event λ^{N_i} means that N_i provides the expected services for an activity. Since the graph is acyclic (only minimum activity paths are considered), then the nodes $N_1, \ldots, N_n$ are different in the path, thus each λ^{N_i} is independent from all others. Hence, we can rewrite the trust in a path as follows:

$$t(\sigma) = \mathbb{P}(\lambda^\sigma) = \mathbb{P}(\lambda^{N_1}) \times \mathbb{P}(\lambda^{N_2}) \times \ldots \times \mathbb{P}(\lambda^{N_n}) = \prod_{i=1}^{n} \mathbb{P}(\lambda^{N_i}) \tag{2.2}$$

Trust in a system for an activity. In general, a system is composed of several paths that represent the different ways a person has, to achieve an activity. The trust level of a person P in a system α to achieve an activity is the probability that she achieves her activity through at least one of the paths in the system. To evaluate the trust in a system for an activity, two cases have to be considered: *(i)* the paths are independent, *i.e.*, they do not have nodes in common[3] and *(ii)* the paths are dependent, *i.e.*, paths having nodes in common.

Independent paths. Let $\{\sigma_i\}_{i\in[1..m]}$ be independent paths that enable a person P to achieve an activity. The probability of achieving the activity through a system, $\mathbb{P}(\lambda^\alpha)$, is the probability of achieving the activity through at least one of the paths σ_i. Thus $\mathbb{P}(\lambda^\alpha)$ is computed as follows:

$$t(\alpha) = \mathbb{P}(\lambda^\alpha) = \mathbb{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2} \vee \ldots \vee \lambda^{\sigma_m})$$

Since the paths are independent then the equation can be rewritten as follows:

$$t(\alpha) = \mathbb{P}(\lambda^\alpha) = 1 - \prod_{i=1}^{m} (1 - \mathbb{P}(\lambda^{\sigma_i})) \tag{2.3}$$

[3]The dependent paths in our graph are the paths that have common nodes (and not common edges) because the trust value is associated to a node, and not to an edge as in a social network.

Dependent paths. When there are common nodes between paths, Relation (2.3) cannot be applied directly.

Two dependent paths with several common nodes. Let σ_1, σ_2, be two paths that enable a person P to achieve an activity. These two paths have several common nodes. By following the same logic as before, we compute the probability that a person P achieves activity ω through system α as follows:

$$t(\alpha) = \mathbb{P}(\lambda^{\alpha}) = \prod_{N \in \sigma_1 \cap \sigma_2} \mathbb{P}(\lambda^{N}) \times \mathbb{P}(\lambda^{\sigma'_1} \vee \lambda^{\sigma'_2})$$

where $\sigma'_1 = \sigma_1 \setminus \sigma_2$, $\sigma'_2 = \sigma_2 \setminus \sigma_1$.

Several dependent paths. A person may have several paths l with common nodes. Thus $\mathbb{P}(\lambda^{\alpha})$ is computed as follows:

$$t(\alpha) = \mathbb{P}(\lambda^{\alpha}) = \mathbb{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2} \vee \ldots \vee \lambda^{\sigma_l}) =$$
$$\mathbb{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2} \vee \ldots \vee \lambda^{\sigma_{l-1}}) + \mathbb{P}(\lambda^{\sigma_l}) - \mathbb{P}(\lambda^{\sigma_l}) \times \mathbb{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2} \vee \ldots \vee \lambda^{\sigma_{l-1}} | \lambda^{\sigma_l}) \tag{2.4}$$

Let us discuss these terms one by one:

- The term $\mathbb{P}(\lambda^{\sigma_l})$ can be computed directly from Relation (2.2).
- The term $\mathbb{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2} \vee \ldots \vee \lambda^{\sigma_{l-1}})$ can be computed recursively using Relation (2.4).
- The term $\mathbb{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2} \vee \ldots \vee \lambda^{\sigma_{l-1}} | \lambda^{\sigma_l})$ needs first to be simplified. If we follow the same logic as before, the term $\mathbb{P}(\lambda^{\sigma_1} \vee \lambda^{\sigma_2} \vee \ldots \vee \lambda^{\sigma_{l-1}} | \lambda^{\sigma_l})$ can be replaced by the term $\mathbb{P}(\lambda^{\sigma'_1} \vee \lambda^{\sigma'_2} \vee \ldots \vee \lambda^{\sigma'_{l-1}})$ where we obtain each $\lambda^{\sigma'_i}$ by eliminating the nodes in common with σ_l.
- $\mathbb{P}(\lambda^{\sigma'_1} \vee \lambda^{\sigma'_2} \vee \ldots \vee \lambda^{\sigma'_{l-1}})$ can be computed recursively using Relation (2.4), and recursion is guaranteed to terminate while the number of paths is finite.

2.3.3 Experimental evaluations

In this section, we present different experiments, their results, analysis, and interpretation. The main objectives are *(i)* to study the influence of the system organization on the computed trust values and *(ii)* to confront this approach with real users.

Influence of the system architecture on the trust value. This experiment studies the influence of the system organization on the computed trust value. We apply our equations on different systems that have the same number of nodes and the same values of trust assigned to each node, but assembled in different topologies as presented in Table 2.4. The values of trust associated to nodes A,B,C,D,E,F are $0.1, 0.2, 0.3, 0.9, 0.8, 0.7$ respectively.

We compute the trust value $t(\alpha)$ for each system. We obtain very divergent results varying from 0.0144 to 0.9003 as illustrated in Table 2.4. Thus, collecting the values of trust in each separated node in a system is not enough to determine if the system is trustworthy or not for an activity. One must also know how the system is organized. For example, in α_2, all the paths contain the nodes A and B and the trust values in these nodes are quite low, 0.1 and 0.2 respectively, so the system trust value is also low due to the strong dependency on these two nodes in this system.

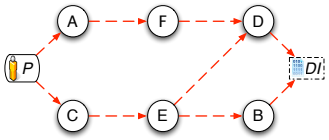
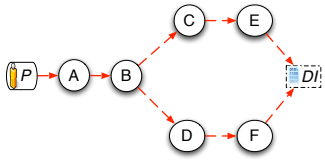
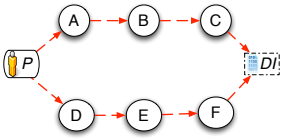
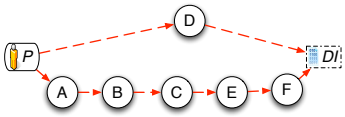
α	$t_\omega(\alpha)$	α	$t_\omega(\alpha)$
 α_1	0.4409	 α_2	0.0144
 α_3	0.507	 α_4	0.9003

Table 2.4: Different systems and their trust values.

Influence of the path length and the number of paths on the trust value. We observed the evolution of the trust value for an activity according to some characteristics of the graph like path's length and number of paths. As a dataset, we consider random graphs composed of 20 to 100 nodes, and 1 to 15 paths. Each node in the graph is associated to a random value of trust from a predefined range.

First, the evolution of trust values according to the paths' lengths in a graph was evaluated. Each simulated graph is composed of 5 paths with lengths varying from 1 to 15 nodes. Different trust values were simulated in the ranges $[0.6, 0.9]$, $[0.1, 0.9]$ *etc.* We notice that the system trust value decreases when the length of paths increases. This reflects a natural intuition we had from the fact that trust values are multiplied.

Second, we set the path lengths to 5 nodes and we increased the number of paths from 1 up to 15 in order to observe the variation of the trust values. Again, different node trust values were simulated in the ranges $[0.7, 0.9]$, $[0.6, 0.9]$, *etc.* As expected, the trust value increases as the number of paths increases. This reflects the intuition that the measure of trust in a system for an activity rises when the number of ways to achieve this activity increases.

Social evaluation (a real case). In order to evaluate our proposal in a real use case, we modeled part of the SVN system of our research laboratory[4] with SocioPath. SVN (Subversion) is a client-server system to manage versions of files. The server allocates repositories of files and clients make copies of repositories. Copies of files contained in repositories can be modified at the client side, modification must be committed to generate new versions. Other clients must frequently update their copies. Persons on which SVN users depend on, are the laboratory that owns the server and the software SVN, the engineer that controls the software at the server side of the SVN, the provider of the software SVN, the computer and the software at the client side, *etc.* We applied the rules of SocioPath on this system for the activity "a user accesses a file on the SVN". Due to privacy issues, Figure 2.7 presents the DAG for this activity with anonymous nodes. For the sake of clarity, we simplify the underlying graph as much as possible.

Based on this context, we conducted an opinion survey among twenty laboratory members including, PhD students, professors and technicians about their level of trust in

[4]https://www.lina.univ-nantes.fr/

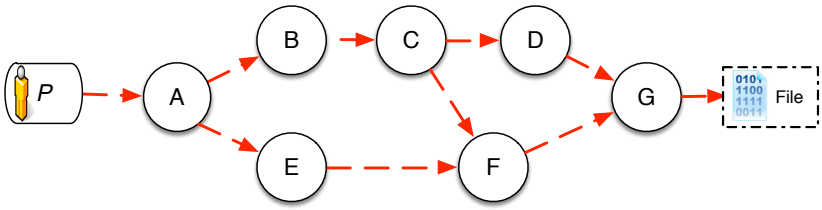


Figure 2.7: LINA's WDAG for the activity "a user accesses a file on the SVN".

each node. For each person, we have computed the system trust value according to the methodology presented in Section 2.3.2. Table 2.5 presents the data of the survey and the computed trust values. In a second phase, we asked each user if the SocioTrust proposal reflects her trust towards the SVN system used in our laboratory. The possibilities of answer were simply `Yes` or `No`. The last column of Table 2.5 presents this feedback, where ✓ means that they are satisfied, and × means that they are not satisfied. 75% of the users are satisfied with the computation. Unsatisfied users argue that they expected a higher trust value. Some of the trust values associated to the nodes of the unsatisfied users, have relatively low values (around 0.5 or 0.6) compared to other users. These users explained that the lack of knowledge about some nodes leads them to define what they called *a neutral value* (*i.e.,* 0.5 or 0.6) that they considered neither trustworthy, nor untrustworthy. Clearly, such a behavior is not compatible with a probabilistic interpretation where 0.5 is like any other possible value between 0 and 1 and has nothing of neutral.

The explanations provided by users revealed an interesting point: even in a small environment and considering advanced users, no one is in possession of all the information necessary to construct an informed assessment. To conform to this reality and model this phenomenon, it is necesary to use a formalism allowing to express uncertainty related to incompleteness of available information. Extending our approach to use subjective logic [59], which can express uncertainty or ignorance, is the objective of the next section.

2.4 SubjectiveTrust: evaluating trust in a system for an activity using subjective logic

SocioTrust is oriented to full-knowledge environments. However, in uncertain environments, users might not be in possession of all the information to provide a dogmatic opinion and traditional probability cannot express uncertainty. With subjective logic [59], trust can be expressed as subjective opinions with degrees of uncertainty.

The main contribution of this section is proposing a generic model named SubjectiveTrust [8], for evaluating trust in a system for an activity taking into account uncertainty. By combining the user's opinion on a node, we are able to estimate two different granularities of trust, namely, *opinion on a path* and *opinion on a system*, both for an activity to be performed by a person. As we know, the main problem that faces trust evaluation based on a graph is the existence of *dependent paths*. To solve this problem, we propose two methods: **Copy** and **Split**.

Subjective logic consists of a set of logical operations which are defined to combine opinions.

	A	B	C	D	E	F	G	System trust value	User's feedback
P_1	0.5	0.5	1	0.5	0.5	1	1	0.4375	✓
P_2	0.7	1	1	0.7	0.7	1	1	0.847	✓
P_3	0.5	0.5	1	0.7	0.5	1	1	0.4375	×
P_4	0.6	0.6	0.8	0.7	0.6	0.8	0.6	0.3072	×
P_5	0.8	0.8	1	0.8	0.8	1	0.9	0.8202	✓
P_6	0.9	0.9	1	0.9	0.9	0.9	0.9	0.9043	✓
P_7	0.6	0.6	0.7	0.6	0.6	0.6	0.7	0.2770	×
P_8	0.8	0.6	1	0.9	0.8	0.8	1	0.7416	✓
P_9	0.7	0.5	1	0.4	0.7	0.6	0.9	0.4407	✓
P_{10}	0.8	1	0.7	0.8	0.8	0.9	0.8	0.6975	✓
P_{11}	0.5	0.5	0.9	0.5	0.5	0.5	0.9	0.2473	×
P_{12}	0.95	0.95	0.8	0.8	0.95	0.95	0.8	0.8655	✓
P_{13}	0.8	0.9	0.8	0.7	0.95	0.8	0.7	0.6433	✓
P_{14}	0.8	0.7	0.9	0.7	0.9	0.8	0.8	0.6652	✓
P_{15}	0.9	0.8	0.8	0.9	0.9	0.9	0.8	0.7733	✓
P_{16}	0.7	0.6	0.6	0.6	0.8	0.7	0.6	0.337	✓
P_{17}	0.5	0.9	0.8	0.7	0.9	0.5	0.8	0.3807	×
P_{18}	0.7	0.7	1	0.7	0.6	0.7	1	0.6088	✓
P_{19}	0.8	0.8	1	1	1	0.8	1	0.8704	✓
P_{20}	0.9	0.9	0.8	0.9	0.9	0.9	0.8	0.7971	✓

Table 2.5: User's trust value in the system SVN in LINA.

- Conjunction operator ($\wedge$) represents the opinion of a person on several propositions.
- Disjunction operator ($\vee$) represents the opinion of a person on one of the propositions or any union of them.
- Discounting operator ($\otimes$) represents the transitivity of the opinions.
- Consensus operator ($\oplus$) represents the consensus of opinions of different persons.

In our work, we rely on a graph to evaluate trust like in the social network domain, but our interpretation of the graph is different. For us, a graph represents *a system for a digital activity* and not *a social network*. This assumption plays an important role in the operations we apply for trust evaluation. That is why, in a social network, to evaluate trust through a path using subjective logic, the operator of discounting ($\otimes$) is used to compute the transitivity through a path, whereas, in our work, evaluating trust in a path is the trust in the *collection* of the nodes that form this path, *i.e.,* conjunction. In the same manner, to evaluate trust through a graph in a social network, the operator of consensus ($\oplus$) is used to evaluate the consensus of opinions of different persons through the different paths that form the graph, whereas, in our work, paths represent the ways one person disposes to achieve an activity, so evaluating trust in a graph is the trust in at least one of the paths or any union of them, *i.e.,* disjunction.

Next section presents SubjectiveTrust and some experiments are presented in Section 2.4.2.

2.4.1 Inferring user's opinion on a system using subjective logic

To focus on trust in the system, the SocioPath model is abstracted in a DAG as in SocioTrust (*cf.* Section 2.3.1). In subjective logic, trust is expressed as an *opinion*, thus

in SubjectiveTrust, the DAG is weighted with opinions, *i.e.*, each node is associated with an opinion in the form (b, d, u, a), where:

- b_x (belief) is the belief that x is true.
- d_x (disbelief) is the belief that the x is false.
- u_x (uncertainty) is the amount of uncommitted belief.
- a_x is called the base rate, it is the a priori probability in the absence of evidence.

We remark that $b_x, d_x, u_x, a_x \in [0, 1]$ and $b_x + d_x + u_x = 1$. a_x is used for computing an opinion's probability expectation value that can be determined as $\mathbb{E}(O_x) = b_x + a_x u_x$. More precisely, a_x determines how uncertainty shall contribute to the probability expectation value $\mathbb{E}(O_x)$.

Several approaches have been proposed to obtain opinion on a node. In [59], authors translate the user's negative or positive observations to opinions. In [73, 74], the opinion parameters are estimated by Bayesian inference. In this study, we do not address the issue of obtaining this opinion, we focus on combining the opinions associated on the nodes to obtain an opinion on a path and on a system for an activity.

Next sections show how an opinion on a path and an opinion on a system are evaluated by combining respectively the opinions on the nodes and the opinions on the paths, using the appropriate operators of subjective logic.

More details about proposed equations and their proofs are in [7, 13]

Opinion on a path for an activity. A path in a system represents a way to achieve an activity. An opinion on a path that contains several nodes can be computed by combining the opinions on the nodes that belong to it.

In trust propagation, the operator to build an opinion on a path is discounting because it allows to compute the transitivity of an opinion along a path [60, 61]. However, if a person needs to achieve an activity through a path, she needs to pass through all the nodes composing this path. Hence, an opinion on a path is the opinion on all nodes composing this path. As the conjunction operator represents the opinion of a person on several propositions, it is appropriate to compute an opinion on a path.

Let $\sigma = \{N_1, N_2, \ldots, N_n\}$ be a path that enables a user P to achieve an activity. P's opinion on the nodes $\{N_i\}_{i\in[1..n]}$ for an activity are denoted by $O_{N_i} = (b_{N_i}, d_{N_i}, u_{N_i}, a_{N_i})$. P's opinion on the path σ for achieving an activity, denoted by $O_\sigma = (b_\sigma, d_\sigma, u_\sigma, a_\sigma)$, can be derived by the conjunction of P's opinions on $\{N_i\}_{i\in[1..n]}$. $O_{\sigma=\{N_1,\ldots,N_n\}} = \bigwedge\{O_{N_i}\}_{i\in[1..n]}$.

$$\begin{cases} b_{\sigma=\{N_1,\ldots,N_n\}} = b_{\bigwedge\{N_i\}_{i\in[1..n]}} = \prod_{i=1}^{n} b_{N_i} \\ d_{\sigma=\{N_1,\ldots,N_n\}} = d_{\bigwedge\{N_i\}_{i\in[1..n]}} = 1 - \prod_{i=1}^{n} (1 - d_{N_i}) \\ u_{\sigma=\{N_1,\ldots,N_n\}} = u_{\bigwedge\{N_i\}_{i\in[1..n]}} = \prod_{i=1}^{n}(b_{N_i} + u_{N_i}) - \prod_{i=1}^{n}(b_{N_i}) \\ a_{\sigma=\{N_1,\ldots,N_n\}} = a_{\bigwedge\{N_i\}_{i\in[1..n]}} = \frac{\prod_{i=1}^{n}(b_{N_i}+u_{N_i}a_{N_i})-\prod_{i=1}^{n}(b_{N_i})}{\prod_{i=1}^{n}(b_{N_i}+u_{N_i})-\prod_{i=1}^{n}(b_{N_i})} \end{cases} \tag{2.5}$$

Opinion on a system for an activity. In trust propagation, to build an opinion on a target node in a graph, the consensus operator is used because it represents the consensus of the opinions of different persons through different paths [60, 61]. In our work, an opinion on a system is the opinion of a person on one or several paths. Thus, the disjunction operator is appropriate to evaluate an opinion on a system. In the following, we show how to build an opinion on a system when *(i)* the system has only independent paths and *(ii)* the system has dependent paths.

Independent paths. Let $\{\sigma_1, \sigma_2, \ldots, \sigma_m\}$ be the paths that enable a user P to achieve an activity. The user's opinion on the paths $\{\sigma_i\}_{i\in\{1..m\}}$ for an activity are denoted by $O_{\sigma_i} = (b_{\sigma_i}, d_{\sigma_i}, u_{\sigma_i}, a_{\sigma_i})$. The user opinion on the system α for achieving the activity, denoted by $O_\alpha = (b_\alpha, d_\alpha, u_\alpha, a_\alpha)$ can be derived by the disjunction of P's opinions in $\{\sigma_i\}_{i\in\{1..m\}}$. Thus, $O_\alpha = \bigvee\{O_{\sigma_i}\}_{i\in\{1..m\}}$.

From [59], we obtain the following generalization: $O_{\alpha=\{\sigma_1,\ldots,\sigma_m\}} =$

$$\begin{cases} b_{\alpha=\{\sigma_1,\ldots,\sigma_m\}} = b_{\bigvee\{\sigma_i\}} = 1 - \prod_{i=1}^{m}(1 - b_{\sigma_i}) \\ d_{\alpha=\{\sigma_1,\ldots,\sigma_m\}} = d_{\bigvee\{\sigma_i\}} = \prod_{i=1}^{m} d_{\sigma_i} \\ u_{\alpha=\{\sigma_1,\ldots,\sigma_m\}} = u_{\bigvee\{\sigma_i\}} = \prod_{i=1}^{m}(d_{\sigma_i} + u_{\sigma_i}) - \prod_{i=1}^{m}(d_{\sigma_i}) \\ a_{\alpha=\{\sigma_1,\ldots,\sigma_m\}} = a_{\bigvee\{\sigma_i\}} = \frac{\prod_{i=1}^{m}(d_{\sigma_i}+u_{\sigma_i}) - \prod_{i=1}^{m}(d_{\sigma_i}+u_{\sigma_i}-u_{\sigma_i}a_{\sigma_i})}{\prod_{i=1}^{m}(d_{\sigma_i}+u_{\sigma_i}) - \prod_{i=1}^{m}(d_{\sigma_i})} \end{cases} \tag{2.6}$$

Dependent paths. As we know, in subjective logic, as in probabilistic logic, the conjunction is not distributive over the disjunction. In SocioTrust, this problem has been resolved by using conditional probability. As there is not a similar formalism in subjective logic, for evaluating trust in a system we propose to transform a graph having dependent paths to a graph having independent paths. Figure 2.8 illustrates this transformation. The left side of this figure shows a graph that has three dependent paths. The dependent paths are[5]: $\sigma_1 = \{A, B, C\}$, $\sigma_2 = \{A, E, F\}$ and $\sigma_3 = \{D, E, F\}$. The common nodes are A, E and F. For instance, A is a common node between σ_1 and σ_2. In that transformation, A is duplicated in A_1 and A_2, such that in the new graph, $A_1 \in \sigma'_1 = \{A1, B, C\}$, and $A_2 \in \sigma'_2 = \{A2, E, F\}$, so is the case for the nodes E and F. The right part of Figure 2.8 shows the new graph after duplicating the common nodes. The new graph contains the paths $\sigma'_1 = \{A1, B, C\}$, $\sigma'_2 = \{A2, E1, F1\}$ and $\sigma'_3 = \{D, E2, F2\}$. Once this transformation is made, we can apply the Relations (2.5) and (2.6). To do so, we propose the following methods.

Copy. In this method, once the graph is transformed to obtain independent paths, we associate the opinion on the original node to the duplicated nodes. This method is based on the idea that the new produced path σ' maintains the same opinion of the original path σ. In this case $O_{\sigma_1} = O_{\sigma'_1}$ and $O_{\sigma_2} = O_{\sigma'_2}$.

Split: In this method, once the graph is transformed to obtain independent paths, in order to maintain the opinion on the global system, we split the opinion on the original dependent node into independent opinions, such that their disjunction produces the original

[5]We recall that the person, the data instance, and the data are not considered in paths of the DAG.

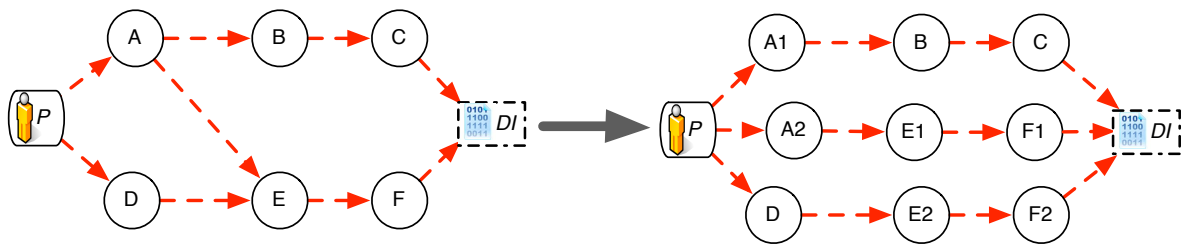


Figure 2.8: Graph transformation.

opinion. Formally speaking, if node A is in common between σ_1 and σ_2, and the opinion on A is O_A, A is duplicated into $A_1 \in \sigma'_1$ and $A_2 \in \sigma'_2$ and the opinion O_A is split into O_{A_1} and O_{A_2}, where O_{A_1} and O_{A_2} satisfy the following relations: $O_{A_1} = O_{A_2}$ and $O_{A_1} \vee O_{A_2} = O_A$.

2.4.2 Experimental evaluation

In this section, we compare **Copy** and **Split** to a modified version of an approach of the literature named TNA-SL [61]. The latter approach is based on simplifying the graph by deleting the dependent paths that have high value of uncertainty, then, trust is propagated. In our work, trust is not propagated and a comparison to a propagation approach has no sense. Thus, we modify TNA-SL such that trust evaluation is made by applying Relations (2.5) and (2.6) introduced in Section 2.4.1. We call this method "modified TNA-SL", denoted **mTNA** in the following.

We present different experiments, their results, analysis and interpretation. The main objectives are *(i)* to compare the proposed methods and evaluating their accuracy and *(ii)* to confront this approach with real users.

Comparing the proposed methods. To tackle the first objective, we experiment with a graph that contains only independent paths. The three methods, **mTNA**, **Copy** and **Split** give the same exact results as expected because the three of them follow the same computational model when graphs contain only independent paths. Then, we experiment on a graph that has relatively high rate of common nodes and dependent paths. 75% of the paths of the chosen graph are dependent paths and 60% of nodes are common nodes.

In our experiments, random opinions $O_N = (b_N, d_N, u_N, a_N)$ are associated to each node, and the opinion's probability expectation value of the graph, $\mathbb{E}(O_\alpha) = b_\alpha + a_\alpha u_\alpha$ is computed using the three methods, **mTNA**, **Copy** and **Split**. This experiment is repeated 50 times where each time represents random opinions of a person associated to the different nodes that compose the graph. We analyze the opinion's probability expectation values of the graph, $\mathbb{E}(O_\alpha) = b_\alpha + a_\alpha u_\alpha$ and not all the opinion parameters $O_\alpha = (b_\alpha, d_\alpha, u_\alpha, a_\alpha)$ for simplicity.

Figure 2.9 shows obtained results. We notice that the three methods almost have the same behavior, when the $\mathbb{E}(O_\alpha)$ increases in one method, it increases in the other methods, and vice versa. We also observe some differences among the three methods that are not always negligible like in experience 9 and 40 in Figure 2.9. This observation leads us to the question: *which of these methods give the most accurate results?* To evaluate the accuracy of **Split**, **Copy** and **mTNA**, we conduct the next experiments.

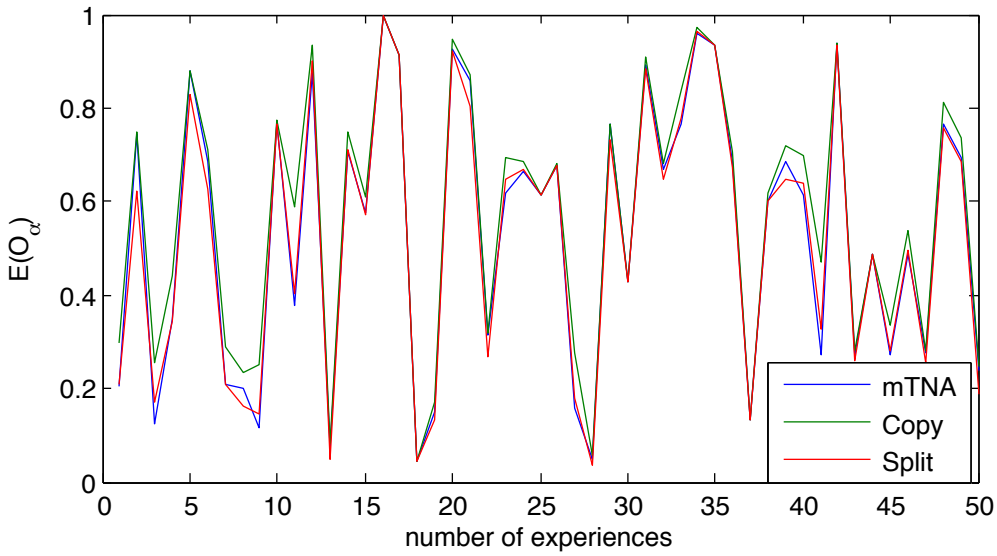
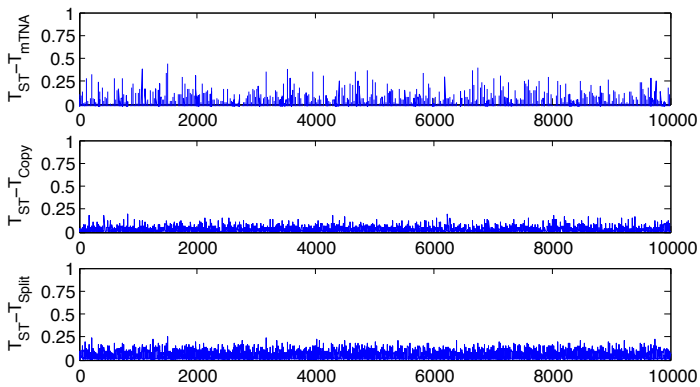


Figure 2.9: Value of the probability expectation for 50 persons using the three methods **mTNA**, **Copy** and **Split**.



Method	Avg	SD
$\mid T_{ST} - T_{\text{mTNA}} \mid$	0.024	0.045
$\mid T_{ST} - T_{\text{Copy}} \mid$	0.014	0.020
$\mid T_{ST} - T_{\text{Split}} \mid$	0.032	0.037

Figure 2.10: Difference between the opinion's probability expectation of a graph using **mTNA**, **Copy**, and **Split** when $u = 0$ and the trust value resulting from using SocioTrust.

Studying the accuracy of the proposed methods. SocioTrust that uses theory of probability to evaluate trust in a system, has the advantages that it has no approximations in case there are dependent paths thanks to conditional probability (*cf.* Section 2.3). Thus it works perfectly if users are sure of their judgments of trust, *i.e.*, the values of uncertainty are equal to 0.

Subjective logic is equivalent to traditional probabilistic logic when $b + d = 1$ such that the value of uncertainty is equal to 0. When $u = 0$, the operations in subjective logic are directly compatible with the operations of the traditional probability. In this case the value of $\mathbb{E}(O) = b + au = b$ corresponds to the probability value.

Since SocioTrust is based on probability theory, the obtained results by applying subjective logic if $u = 0$ should be equal to the ones using probability theory. We can

evaluate the accuracy of the proposed methods by setting $u = 0$ and comparing the value of $b_\alpha = \mathbb{E}(O_\alpha)$ resulted from applying the three methods to the trust value obtained by applying SocioTrust.

The experiments are conducted on the graph used in Figure 2.9. Random opinions $O_N = (b_N, d_N, 0, a_N)$ are associated to each node, and the probability expectation of the graph $\mathbb{E}(O_\alpha) = b_\alpha + a_\alpha u_\alpha = b_\alpha$ is computed. The notation T_{ST}, denotes system's trust value resulting from applying SocioTrust. While T_{mTNA}, T_{Copy}, and T_{Split} respectively denote system's opinion probability expectation resulting from applying **mTNA**, **Copy**, and **Split**.

To compare T_{ST} to T_{mTNA}, T_{Copy}, and T_{Split}, we simply compute the subtractions between them *i.e.*, $T_{ST} - T_{\text{mTNA}}$, $T_{ST} - T_{\text{Copy}}$, $T_{ST} - T_{\text{Split}}$. The average of each of the previous values are computed through 10,000 times to obtain a reliable value. The standard deviation (SD) is also computed to show how much variation from the average exists in the three cases. Figure 2.10 shows obtained results.

As we notice from Figure 2.10, **Copy** is the method that gives the closest results to SocioTrust, the average of the difference of its result when $u = 0$ and the result of traditional probability over $10,000$ times is equal to 0.014, which is an indication that this method gives the nearest result to the exact result and its average error rate is around 1.4%. **Copy** shows the most convincing result, with a standard deviation equals to 0.02.

The average error rate of **mTNA** (2.4%) is less than **Split** (3.2%), but the standard deviation of **mTNA** is 0.045 where in **Split**, it is 0.037. That means that in some cases, **mTNA** can give results that are farther than **Split** from the exact results. Thus, **Split** shows a more stable behavior than **mTNA**.

The objective of this experiment is not criticizing the proposed methods in the literature for the problem of dependent paths. These methods are proposed to deal with the problem of trust propagation through a graph, whereas, in our work we focus on evaluating trust towards the whole graph. The employed operators in our case are different from the employed operators in trust propagation. TNA-SL or any proposed method in the literature can work properly in their context.

In this experiment, we show that **Copy** is the method the more adaptable to be used with respect to the context of our work. Extensive simulations on different types of graphs are provided in [7] and follow the same behavior presented above.

Social evaluation (a real case). In this experiment we use the SVN system of our research laboratory introduced in Section 2.3.3. Since subjective logic is not used yet in real applications, users are not used to build an opinion directly using this logic. We build these opinions ourselves from users' positive or negative observations as it is proposed in [59]. To do that, a survey is executed to collect the user's observations about the nodes. The proposed questions collect information about the user's usage of a node and the quantity of using it and their observations. A local opinion on each entity is built for each user. The opinion and the opinion's probability expectation of the system are then computed using **Copy** for each user. The results are shown in Table 2.6.

We asked each user for a feedback about their opinion on the nodes and in the system. We were glad to notice that users were satisfied of the obtained results, whereas when using SocioTrust (*cf.* Section 2.3.3), 25% of users were not satisfied. In the latter approach, when users do not have enough knowledge about a node, they assign the value 0.5, that

they consider as neutral value. That leads to incorrect inputs that produce low trust values in a system. In SubjectiveTrust, uncertainties are expressed in the opinions on the nodes and computing an opinion on a system is made considering these uncertainties. That shows that, in uncertain environments, it is more suitable to use subjective logic than probabilistic metrics for trust evaluations.

	O_A (b,d,u,a)	O_B (b,d,u,a)	O_C (b,d,u,a)	O_D (b,d,u,a)	O_E (b,d,u,a)	O_F (b,d,u,a)	O_G (b,d,u,a)	O_α (b,d,u,a)	$\mathbb{E}(O_\alpha)$
P_1	$(1,0,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.99,0.01,0,0.5)$	$(0.83,0,0.17,0.5)$	$(0.99,0.01,0,0.5)$	$(0.83,0,0.17,0.5)$	$(0.6820,0,0.3810,0.8389)$	0.9488
P_2	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(0.83,0,0.17,0.5)$	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(1,0,0,-)$	1
P_3	$(0.99,0.01,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.99,0.01,0,0.5)$	$(0,0,1,0.5)$	$(0.99,0.01,0,0.5)$	$(0.6,0,0.4,0.5)$	$(0,0,1,0.7753)$	0.7753
P_4	$(1,0,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(0.83,0,0.17,0.5)$	$(0.99,0.01,0,0.5)$	$(0.96,0,0.04,0.5)$	$(0.7888,0,0.2112,0.8604)$	0.9705
P_5	$(0.99,0.01,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.99,0.01,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.5,0,0.5,0.5)$	$(0,0,1,0.7500)$	0.75
P_6	$(0.99,0.01,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.9,0.1,0,0.5)$	$(0.5,0,0.5,0.5)$	$(1,0,0,0.5)$	$(0.6,0,0.4,0.5)$	$(0.2970,0,0.7030,0.7755)$	0.8422
P_7	$(0.99,0.01,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.9,0.1,0,0.5)$	$(0.83,0,0.17,0.5)$	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(0.8217,0,0.1783,0.8522)$	0.9736
P_8	$(1,0,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.9,0.1,0,0.5)$	$(0.5,0,0.5,0.5)$	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(0.5000,0,0.5000,0.8625)$	0.9313
P_9	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(0.95,0.05,0,0.5)$	$(0,0,1,0.5)$	$(0.99,0.01,0,0.5)$	$(0.96,0,0.04,0.5)$	$(0.9956,0,0.0044,0.7583)$	0.9989
P_{10}	$(0.99,0.01,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.9,0.1,0,0.5)$	$(0,0,1,0.5)$	$(0.8,0.2,0,0.5)$	$(0.98,0,0.02,0.5)$	$(0,0.0047,0.9953,0.7972)$	0.7934
P_{11}	$(0.99,0.01,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.95,0.05,0,0.5)$	$(0.72,0,0.28,0.5)$	$(0.99,0.01,0,0.5)$	$(0.96,0,0.04,0.5)$	$(0.6774,0.0001,0.3225,0.8489)$	0.9512
P_{12}	$(1,0,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.95,0.05,0,0.5)$	$(0.83,0,0.17,0.5)$	$(0.95,0.05,0,0.5)$	$(1,0,0,0.5)$	$(0.7885,0,0001,0.2114,0.8301)$	0.9640
P_{13}	$(1,0,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.95,0.05,0,0.5)$	$(0.83,0,0.17,0.5)$	$(0.95,0.05,0,0.5)$	$(0.83,0,0.17,0.5)$	$(0.6545,0.0001,0.3545,0.8110)$	0.9346
P_{14}	$(1,0,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.99,0.01,0,0.5)$	$(0.72,0,0.28,0.5)$	$(0.99,0.01,0,0.5)$	$(0.72,0,0.28,0.5)$	$(0.5132,0,0.4868,0.8186)$	0.9117
P_{15}	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(0.99,0.01,0,0.5)$	$(0.72,0,0.28,0.5)$	$(0.99,0.01,0,0.5)$	$(0.83,0,0.17,0.5)$	$(0.9870,0,0.0130,0.8492)$	0.9980
P_{16}	$(0.99,0.01,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.9,0.1,0,0.5)$	$(0.5,0,0.5,0.5)$	$(1,0,0,0.5)$	$(0.72,0,0.28,0.5)$	$(0.3564,0,0.6436,0.8011)$	0.8719
P_{17}	$(1,0,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.99,0.01,0,0.5)$	$(0.83,0,0.17,0.5)$	$(1,0,0,0.5)$	$(0.83,0,0.17,0.5)$	$(0.6889,0,0.3111,0.8447)$	0.9517
P_{18}	$(1,0,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.99,0.01,0,0.5)$	$(0.83,0,0.17,0.5)$	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(0.8300,0,0.1700,0.8737)$	0.9785
P_{19}	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(1,0,0,0.5)$	$(0.99,0.01,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.72,0,0.28,0.5)$	$(0.9196,0,0.0804,0.8525)$	0.9811
P_{20}	$(1,0,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.95,0.05,0,0.5)$	$(0,0,1,0.5)$	$(1,0,0,0.5)$	$(0.83,0,0.17,0.5)$	$(0,0,1,0.8836)$	0.8336

Table 2.6: Users' opinions in the system for the activity "a user access a file on the SVN" of our research laboratory.

2.5 Conclusion

In this chapter, I proposed SocioPath, a simple model that allows to formalize the entities in a system and the relations among them. In this contribution, we observed that the entities that compose a digital system can be digital, physical or human entities. We provided this model with the rules that discover some implicit relations in a system and enriched it with definitions that illustrate some main concepts about the used system.

Trust works in the literature focus on one granularity of trust, *i.e.*, in one entity: trusting a person, a product, a resource, *etc.* Trusting a system as a composition of a set of entities and relations between them has not been studied deeply.

From SocioPath models, we can obtain a directed acyclic graph (DAG) where nodes represent a set of entities that plays a role for achieving the users' activity and the set of edges represents the paths a user follows to achieve her activity.

Based on this DAG we proposed two approaches to evaluate trust in a system for an activity. The first one, SocioTrust, is based on probability theory. It can be used in the field of full-knowledge environments. In presence of uncertainty, the second approach based on subjective logic, SubjectiveTrust, is more suitable. The necessary relations and algorithms for combining the trust values towards the entities in the DAG have been provided and proved, and experiments have been conducted to validate these approaches.

All the evaluations of trust in a system we proposed are static. This is a limitation. To achieve a better comprehension of trust in a system and the parameters that can influence it, it will certainly be necessary to consider the evolution of trust over the time. We are convinced that such understanding is a a challenging issue. For this purpose, it is also necessary to compare synthetic trust and real trust of a user. Yet, to the best of our knowledge, there is no method to measure a distance or similarity between an assessment of confidence and the one felt by users. It is certainly possible to build on work already carried out in the fields of Information Retrieval or Social Sciences, but this is a problem we encountered without providing a complete answer. Indeed in our work, we collected users' impressions through a form and showed they feel closer to a proposal than the other. However, a general method of comparison and measurement between an assessment of the trust and the trust really felt remains to build.

It is also interesting to note that SocioPath is not restricted to trust evaluation. Indeed, pointing out accesses and controls relations within an architecture is also related to privacy. Thus, as future work, it could be interesting to use SocioPath to study the compliance of system with users' privacy policies.

Chapter 3

Supporting data privacy and measuring satisfaction in P2P systems

Contents

3.1 Introduction **33**
3.2 Hippocratic databases in P2P systems **34**
3.2.1 PriMod: a privacy model for P2P systems 36
3.2.2 PriServ: a privacy service for P2P systems 40
3.3 WUW: a distributed P2P framework to enhance users' satisfaction . . **46**
3.3.1 Defining strategies and their evaluation 47
3.3.2 Implementing WUW 50
3.4 Conclusion **54**

3.1 Introduction

Peer-to-peer (P2P) networks are generally classified into two main categories: pure and hybrid [94]. In pure P2P networks, all peers are equal and they can be divided in structured and unstructured overlays. In hybrid P2P networks (also called super-peer P2P networks), some peers act as dedicated servers for some other peers and have particular tasks to perform.

In my work, I focus in pure P2P systems for data-centered applications because of their valuable characteristics: (a) decentralized storage and control, so there is no need to trust one particular server; (b) data availability and fault tolerance, thanks to data replication; (c) scalability to store large amounts of data and manage high numbers of users; (d) autonomy, as peers can join and leave the network at will.

However, despite their assets, these P2P systems offer limited guarantees concerning data privacy. They can be considered as hostile because data, that can be sensitive or confidential, can be accessed by everyone (by potentially untrusted peers) and used for

everything (*e.g.*, for marketing, profiling, fraudulence, or for activities against the owner's preferences or ethics). Several P2P systems propose mechanisms to ensure privacy such as OceanStore [69], Past [105], and Freenet [22]. However, these solutions remain insufficient. Data privacy laws have raised the respect of user privacy preferences where *purpose-based access* is cornerstone[1]. Managing data sharing, with trustworthy peers, for specific purposes and operations, is not possible in current P2P systems without adding new services.

In addition, in P2P architectures, users' resources are the wealth of the system, thus *users satisfaction* over the system is essential to avoid resources decrease. Incentive mechanisms based on QoS characteristics, offer personalised system performance and reliability. Nonetheless, users are individuals having different preferences and interests unrelated to bandwidth consumption or number of connections. We argue that users should be able to define personal *strategies*, according to their expectations. They should be able to influence the behavior of their system beyond the choice of technical parameters. For instance, a user may prefer to exchange mainly with users which interests are similar to hers, or with users that are located in regions of the world where human rights are respected, or where digital data protection is well regulated, *etc.*

In my work I, propose services that do not impose high load on single components, nor it requires centralized management. On the one hand, I propose a solution to support data privacy in structured P2P systems. The key idea is to apply Hippocratic database (HDB) principles to data sharing in P2P systems. Inspired by the Hippocratic oath and its tenet of preserving privacy, HDBs [4] incorporated purpose-based privacy protection, which allows users to specify the purpose for which their data are accessed. However, HDBs were proposed for centralized relational database systems. Thus my contribution is to apply it to structured P2P networks.

On the other hand, I propose a solution to measure and improve the satisfaction of the P2P users based on their personal preferences that reflect their expectations from the P2P system. The proposed solution was based on the Satisfaction-based Query Load Balancing framework (SQLB) introduced in [98] as a generic framework to measure participants' satisfaction in the context of "query allocation" in a client-server context. My contribution is to adapt SQLB to the P2P context.

This chapter is organized as follows. Section 3.2 introduces PriMod, a privacy model that applies HDB principles to data sharing in P2P systems and describes PriServ, a privacy service that supports PriMod in structured P2P networks. Section 3.3 introduces WUW (What Users Want), a framework that allows to compare and evaluate in a distributed way users' satisfaction in a P2P overlay. Section 3.4 concludes.

Contributions of this chapter were published in [16, 17, 51, 52, 53, 54, 55, 56, 57, 104].

3.2 Hippocratic databases in P2P systems

Inspired by the Hippocratic oath and its tenet of preserving privacy, Hippocratic databases (HDB) [4] aim at incorporating privacy protection within relational database systems. HDBs define ten founding principles to protect data privacy according to users preferences. 1. Purpose Specification, 2. Consent, 3. Limited Collection, 4. Limited Use, 5. Limited Disclosure, 6. Limited Retention, 7. Accuracy, 8. Safety, 9. Openness, and 10. Compliance.

[1] https://www.cnil.fr/fr/reglement-europeen-protection-donnees

In an HDB, queries are submitted along with their intended purpose. Query execution preserves privacy by using query rewriting and restrictions by column, row, or cell.

Purpose specification. Purpose specification is the cornerstone of an HDB. It states that purposes should be specified and attached to data items to control their usage. In order to do this, simple specification languages such as Platform for Privacy Preferences (P3P) [25] can be used as a starting point. P3P is a standard developed by the World Wide Web Consortium. Its goal is to enable users to gain more control over the use of their personal information on web sites they visit. P3P provides a way for a Web site to encode its data-collection practices in a machine-readable XML format, known as a P3P policy [25], which can be programmatically compared against a user's privacy preferences [70]. In [66, 67] authors propose ideas for reducing the complexity of the policy language which include arranging purposes in a hierarchy. Subsumption relationships may also be defined for retention periods and recipients.

Limited disclosure. Limited disclosure is another vital component of an HDB system. It states that the private data shall not be disclosed for purposes other than those defined by the data owner. A scalable architecture for enforcing limited disclosure rules and conditions at the database level is proposed in [71]. For enforcing privacy policies in data disclosure, privacy policies can be stored and managed in the database. These policies are expressed in high-level privacy specification languages (*e.g.*, P3P). Enforcing privacy policies does not require any modification to existing database applications. Authors provide techniques for enforcing a broad class of privacy policies by automatically modifying all queries that access the database in a way that the desired disclosure semantics is ensured. They examine several implementation issues, including privacy metadata storage, query modification algorithms, and structures for storing conditions and individual choices.

HDB implementation. Subsequent works have proposed solutions for implementing HDBs. In [3], authors address the problem of how current relational DBMS can be transformed into their privacy-preserving equivalents. From specifications of privacy policies, they propose an algorithm that defines restrictions (on columns, rows, and cells) to limit data access. In [19], authors propose query modification techniques and access control to ensure data privacy based on purposes. They propose to organize purposes in a tree hierarchy where the root is the most general purpose and the leafs the more specific ones. In this way, if data access is allowed for a purpose x, all descendant purposes of x are also allowed. They also propose data labeling (with allowed purposes) at different granularity levels (table, column, row, or cell). In addition, they propose some SQL modifications to include purposes, for instance **Select** column-name **From** table-name **For** purpose-name.

HDBs are the first privacy techniques that include the notion of purpose in relational databases. They are essential to users who would like to know for which purpose their data are used. However, enforcing HDBs in P2P systems is a challenge which we address in the next sections.

3.2.1 PriMod: a privacy model for P2P systems

I propose PriMod, a data privacy model for P2P systems, to answer the need of data owners to share their sensitive data in a P2P system while preserving data privacy. It makes no assumptions about the P2P system organization. The unique important hypothesis is that each peer has a unique identifier in the system for all its connections. The purpose notion is mainspring of PriMod functionalities. PriMod allows owners to specify their privacy policies (PPs) and to publish data for specific purposes and operations. They can choose between publishing only their data references (*e.g.*, filenames, primary keys, *etc.*) or publishing encrypted data content. Requesters can search for sensitive data but must specify the access purpose and operation in their requests, thus they are committed to their intended and expressed use of data. Requesters can also ask which sensitive data they can access for a particular purpose.

To summarize, the PriMod assets are the following:

- It benefits from P2P assets in data publishing and sharing while offering data privacy protection based on access purposes.
- It can be easily integrated to any P2P system.
- It proposes/uses privacy policies concepts and defines models for trust and data management.
- It offers operations for publishing data content, publishing references, requesting, and purpose-based searching.
- Data owners can define their privacy preferences in privacy policies.
- Sensitive data are associated with privacy policies. This association creates private data ready to be published in the system.
- Requests are always made for particular purposes and operations.
- Trust techniques are used to verify trustworthiness of requester peers.

Privacy policy model In PriMod, each data owner should define her privacy preferences. Those privacy preferences are registered in PPs independently of data. Once defined, they are attached to appropriate data. PPs are dynamic because they can vary with time. For instance, at the end of the medical treatment, a doctor will only allow reading access to other doctors for analyzing the patient medical record. Updating diagnosis will not be allowed anymore. We consider that each owner is responsible for defining and maintaining her PPs in an independent way.

Inspired from the Platform for Privacy Preferences (P3P) [25], Figure 3.1 shows a PP model. This model does not claim to be exhaustive, but shows information about PPs that can include the next information.

Authorized Users. It is a list of users who are authorized to access data, a kind of ACL. A user can be an individual or a group.

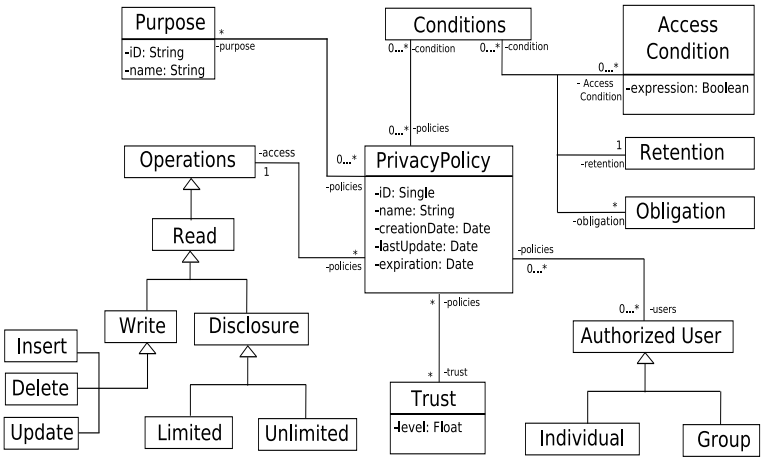


Figure 3.1: Privacy policy (PP) model

Purpose. An access purpose states the data access objective. With this concept, an owner is able to specify the purposes for which its data can be accessed by users.

Operations. An operation determines what a peer can do with data. We use three basic operations, read, write, and disclose, but others can be defined.

- **Read.** A peer can read the data content.
- **Write.** A peer can modify the data content with the following operations: insert, update, delete.
- **Disclose.** A peer is able to disclose shared data for other peers. Disclosure can be limited or unlimited. If a peer disclosure right is limited, it cannot give disclosure rights on the data to other peers.

Conditions. Conditions state the access conditions a user should respect, the obligations a user should accomplish after accessing data, and the limited time for data retention.

- **Access condition.** Conditions state under which semantic condition data can be accessed. This may concern data values, for example age>10.
- **Obligation.** Obligations state the obligation a user must accomplish after the data access. For example, researcher Ri should return research results after using the patient records.
- **Retention time.** The retention time states the time limit of retention of the data. For example, the local copy obtained by a requester of a patient record should be destroyed after 1 year of use.

Minimal trust level. It is the minimal trust level a requester peer should have in order to gain access to data.

Purpose-based data reference table j (PBDRT j)		
Key (PK)	**requesterID**	**DataRefList**
hash(diagnosis, write)	Doctor1	{DataRef1}
	Doctor2	{DataRef1, DataRef2}
hash(research, read)	Doctor1	{DataRef1}
	Scientist1	{DataRef1}
	Scientist2	{DataRef1, DataRef3, DataRef5}
hash(accessing, read)	Patient1	{DataRef1}
	Patient2	{DataRef2}
	Patient3	{DataRef3, DataRef5}

Table 3.1: Purpose-based data reference table (PBDRT) of doctor Dj.

Data model In order to respect PPs, they should be associated with data. In the following, we use relational tables, however any type of data can be considered (files, XML documents, rich text files, *etc.*).

Data table. Each owner peer stores locally the data it wants to share. Those data can be stored in relational tables called *data tables*. The unique restriction about data tables is that primary keys should be generic and impersonal to respect privacy and to not disclose any information. If considered data are files, their identifiers or names should be impersonal.

Purpose table. Information about the available purposes are stored in a table named *purpose table*. A tuple of the purpose table contains the purpose identifier, the purpose name, and the purpose description. We recall that in HDB, purposes can be organized in a hierarchy. To simplify, in PriMod, we make abstraction of such hierarchy.

Trust table. Each peer maintains a local *trust table* that contains the trust level of some peers in the system. A tuple of the trust table contains the identifier of a peer, its trust level, and a cell defining if this peer is consider as a *friend* or not locally.

Private data table. This table joins data to privacy policies. It allows fine-grained access control by specifying which table, column, line, or cell can be accessed by preserving which privacy policy.

Purpose-based data reference table. To ease data searching, a purpose-based index is necessary. Information about the references of data allowed for particular purposes and operations for particular requesters are stored in a table named *purpose-based data reference table* (PBDRT for short). This purpose-based index allows requesters to know which data they can access for a particular purpose and operation. Each tuple of this table is identified by a *key*, obtained for instance by hashing the couple $(purpose, operation)$ (see Table 3.1). Besides the key, a tuple contains the identifiers of requesters and the list of data references the are allowed to access.

PriMod proposes the next set of functions.

Publishing data. PriMod provides two ways of publishing sensitive data indicating the PP that users should respect. An owner may choose to publish her data content or only data references. In the first case, data storage is protected from malicious servers by using cryptography techniques. In the second case, there is no need of data encryption since references do not show any private information about the data if they are well-chosen (*i.e.*, personal information such as security numbers and addresses should not be used in references).

Boolean publishData(data, PPId). Owner peers use this function to publish data content in the system. The second parameter is the privacy policy that dictates the usage conditions and access restrictions of the published data. This function returns true if data content is successfully distributed, false otherwise. It is similar to a traditional publishing function. To protect data privacy against potential malicious servers, before distribution, data content is encrypted (by using symmetric cryptography) and digital checksums are used to verify data integrity. Symmetric keys are stored locally by the owner. Requesters must contact owners to retrieve keys and decrypt requested data.

Boolean publishReference(data, PPId). Owner peers use this function to publish data references in the system while data content are stored locally. This function returns true if data references are successfully distributed, false otherwise. Servers store data references and help requesters to find data owners to obtain data content. Publishing only data references allows owners to publish private data while being sure that data content will be provided to right requesters. This hypothesis can not be guaranteed in the previous function because malicious servers may misbehave by returning encrypted data to unauthorized peers.

Requesting data. For requesters, how data have been published is transparent and a unique function to request data is proposed by PriMod.

Data request(dataRef, purpose, operation). Requester peers use this function to request data (*dataRef*) for a specific purpose (*e.g.*, researching, diagnosis, or analysis) to perform a specific operation (*i.e.*, read, write, or disclose). This function returns the requested data if the requester has corresponding rights, otherwise it returns null. This function compels requesters to specify the access purposes and the operations that they have the intention to apply to requested data.

This explicit request is the cornerstone of this work, it commits requesters to use data only for the specified purposes and to perform only specified operations. Legally, this commitment may be used against malicious requesters if it is turned out that obtained data have been used differently.

TrustLevel searchTrustLevel(requesterID). Owner peers use this function to search the trust level of the requester *requesterID*. This function returns the trust level of the requester if it is found otherwise it returns null. This trust level is used in the requesting process to verify the trustworthiness of the requester in order to give him access rights.

Purpose-based reference searching. PriMod provides users with a function for purpose-based reference searching based on the PBDRT. This function allows requesters to know which data they are authorized to request for a particular purpose and operation. This prevents users from denying knowledge about their access rights. The allowed data

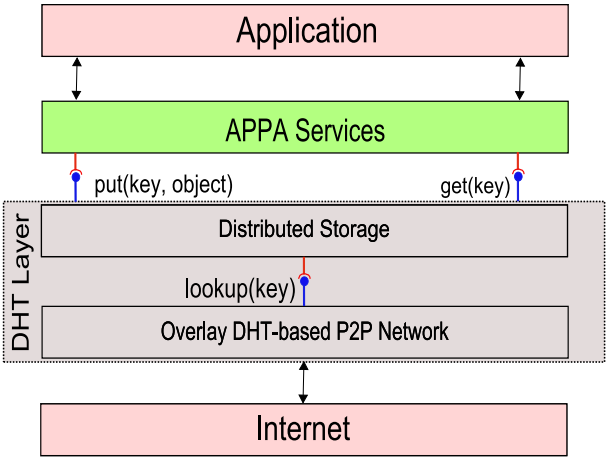


Figure 3.2: Global architecture

reference lists (contained in the PBDRT) can be created transparently while publishing data. These lists can be published periodically in the system.

DataRefList dataRefSearch(purpose, operation). Requester peers use this function to know the data they are authorized to access for a specific purpose and operation. This function returns a list of data references of data the requester is authorized to acces (*dataRefList*). If the list is empty, the function returns null. This request by peer protects privacy because it avoids that all users know which are the access rights of other users and know the complete list of available data (global schema).

3.2.2 PriServ: a privacy service for P2P systems

PriServ is a privacy service that implements PriMod. Figure 3.2 shows the Priserv's architecture, which is on top of a DHT layer. This DHT layer has two functional components: one is in charge of the routing mechanism that supports the *lookup()* function as well as the dynamicity of peers (join/leave of peers); the other ensures key-based data searching and data distribution by implementing the *put()* and *get()* functions. These two layers provide an abstraction from the DHT.

Conceptually, PriServ is an APPA (Atlas Peer-to-Peer Architecture) service [6]. APPA is a data management system for large-scale P2P and Grid applications. The PriServ implementation uses Chord[111] for its efficiency and simplicity, however, any DHT can be used. PriServ uses the traditional get() and put() functions of DHTs to locate and publish data, each incurring $O(logN)$ messages.

- $put(key, data)$ stores a key k and its associated data object in the DHT.
- $get(key)$ retrieves the data object associated with k in the DHT.

Data keys in PriServ are created by hashing the triplet (*dataRef, purpose, operation*). We consider that *dataRef* is a unique data reference, *purpose* is the data access purpose and *operation* is the operation that can be executed on requested data with respect to the corresponding privacy policy. Thus, the same data with different access purposes and different operations have different keys.

Main functions PriServ implements the PriMod functions so it offers to the application layer two ways for publishing and allows searching data and data references for a particular purpose and operation. The main procedures are publishReference(), publishData(), request(), dataRefSearch(), dataRefPut(), and searchTrustLevel(). All but the last function use the DHT organization. The searchTrustLevel() function uses instead an unstructured P2P approach as we will see latter.

Boolean publishReference(data, PPId). Owners use this function to publish data references under a particular PP. Publishing only data references and storing data locally allow owners to provide themselves their data to right requesters.

Boolean publishData(data, PPId). Owners use this function to publish data content under a particular PP (PPId). To protect data privacy against potential untrusted servers, before distribution, data content is encrypted (by using symmetric cryptography), and digital checksums are used to protect data integrity from servers.

Data request(dataRef, purpose, operation). Requesters use this function to request data (*dataRef*) for a specific *purpose* to perform a specific *operation*. This function compels requesters to specify the access purposes and the operation that they will apply to requested data. For requesters, the way data have been published is transparent (publishData or publishReference), so they always use this request function.

During the request process, the servers do an access control based on the ACL sent by the owner during the publishing process. The data owner is always contacted by the requester either to request the data content (if only references have been published) or the decryption key (when encrypted data content have been published). Before retrieving data, owners check the trust level of requesters as you will see in the function searchTrustLevel().

To use this request function, it is necessary to know the references of available data in the system. This information is maintained in an index that can be centralized or distributed. A centralized index represents a point of failure and potential bottlenecks. Besides, it implies to trust one single peer (or server) that has the control and the responsibility of maintaining this important meta-information. We argue that when preserving privacy, the distribution of control is essential, which is why PriServ implements a distributed index.

DataRefList dataRefSearch(purpose, operation). PriServ implements the purpose-based reference searching function of PriMod. The index represented by PBDRT (see Table 3.1) in PriServ is implemented in a distributed way by using the DHT organization. The couple $(purpose, operation)$ is hashed to create the keys of the index. Keys are assigned to peers that are responsible of maintaining information about the peers that can request data for the purpose and operation represented by the key. The hash function used to produce these keys may be different from the one used to publish data. Thus, each peer maintains a PBDRT of all the keys for which its id is the closest in the DHT organization, which gives a partial view of the global index.

To optimize the update of the index, a periodic publication of references can be done by using a dataRefPut() function. The idea is that, if an owner publishes many data for the same $(purpose, operation)$ only one update is done. For that, a local PBDRT is maintained and *flushed* periodically.

The last function that PriServ implements focuses on searching the trust level of requesters. PriServ uses trust levels to make the final decision of sharing or not data. The trust level reflects a peer reputation wrt other peers. A peer can have different trust levels at different peers. The peer reputation influences its trust level. Peers which are suspicious

have lower trust level than peers considered as honest. A peer can have locally the trust levels of some well known peers or peers it has interacted with. If a peer P does not have a particular trust level it can ask for it to its *friends*. A friend is a peer considered as honest from the point of view of P and the number of friends can vary from one peer to another.

The implementation of the searchTrustLevel() function does not use the DHT organization, but uses an unstructured overlay. Thus, this function has been redefined to take into account a Time To Live (TTL).

TrustLevel searchTrustLevel(requesterID, nestedLevel). Trust levels are considered in the range [0..1]. A peer with a trust level of 1 is completly trustworthy. A peer with a trust level of 0 has very bad reputation. During requesting, if the trust manager of the data owner has the trust level of the requester it does not have to contact other peers. Otherwise, PriServ defines three methods for searching the requester trust level. Choosing one of them depends on the number of friends of the owner. Briefly, the three methods are explained below, and only the algorithm of the first one is presented. More details can be found in [53].

With-friends algorithm. This version of the searchTrustLevel function considers that each peer has at least one friend. With this assumption, the data owner asks its friends for the trust level of the requester. Each received trust level (RTL) is weighted with the trust level (FTL) of the sending friend. The final trust level is computed from the received trust levels as the average, the maximum, the minimum, *etc.* This searching is recursive. If a friend does not have the requested trust level, it asks for it to its friends and the number of nested levels (*nestedLevel*) is incremented. Recursion is limited to *maxDepth*. The maximum number of contacted friends can also be limited to a predefined number.

Without-friends algorithm. In this algorithm, we consider that peers have no friends. In this case, data owners ask for the trust level of the requester to the subset of known peers from the DHT, *i.e.*, their finger table .

With-or-without-friends algorithm. Here, peers may have friends or not and priority is given to ask for trust levels to friends. If a data owner has some friends, it asks them for the trust level by using the with-friends algorithm, else it asks the peers in its finger table by using the without-friends algorithm.

Cost analysis

Publishing costs. Publishing data in the system conserves the logarithmic cost of the traditional put function. By using the DHT, $O(logN)$ messages are needed to publish each key. In PriServ, the number of keys is equal to the number of entries (*ept*) of the private data table. Additional costs induced by the cipher key generation and the data encryption are negligible wrt the network costs. Thus, the publishing cost is:

$$C_{Publish} = \sum_{i=1}^{ept} O(logN) = O(ept * logN)$$

The maximum value of *ept* is equal to the number of shared data (*nbData*) multiplied by the number of purposes (*nbPurpose*) multiplied by the number of operation (*nbOperation*). At worst, each data item is shared for all purposes and all operations:

$$CMax_{Publish} = O(nbData * nbPurpose * nbOperation * logN)$$

We can see that the number of purposes and operations affects the publishing cost. Previous studies have shown that considering ten purposes allows to cover a large number of applications [1, 75]. Used with ten purposes (by data item) and three operations (read, write, and disclosure), PriServ incurs a small overhead. Overall, the publishing cost remains logarithmic.

Requesting costs. Concerning the requesting cost, it is the addition of two costs: get() cost and the retrieving cost. We disregard access control, checksum calculation, and decryption costs, which are negligible wrt network costs. The get() cost is in $O(logN)$ and the server returns its answer in one message. For data retrieval, a requester needs one additional message to contact the data owner that answers in another message.

To summarize, the requesting cost is:

$$\begin{aligned} C_{Requesting} &= C_{DHTGet} + C_{Retrieving} \\ &= O(logN) + 1 + 2 \\ &= O(logN) + 3 \\ &= O(logN) \end{aligned}$$

Trust level searching cost. The trust level searching cost (C_{STL}) depends on the trust searching algorithm:

- *With-friends algorithm.* In this case, the owner sends a message to each of its friends that in turn do the same in a nested search. This cost depends on the number of friends (NF) and the maximum depth of the nested search (MaxDepth).

$$C_{STL_{WF}} = \sum_{i=1}^{MaxDepth} NF^{\ i} = O(NF^{MaxDepth})$$

- *Without-friends algorithm.* In this case, the owner sends a message to each of the peers in its finger table, which in turn do the same in a nested search. This cost depends on the number of fingers, which is log N, and the maximum depth of the nested search (MaxDepth).

$$C_{STL_{WOF}} = \sum_{i=1}^{MaxDepth} (logN)^{\ i} = O((logN)^{MaxDepth})$$

- *With-or-without-friends algorithm.* In this case, if the owner has friends, it sends a message to each of its friends. Otherwise, it sends a message to each of the peers in its finger table. A peer contacted by an owner does the same in a nested search. The trust level searching cost depends on the number of friends (NF), the number of fingers, which is log N, and the maximum depth of the nested search (MaxDepth).

$$C_{STL_{WWF}} = O((max(logN, NF))^{MaxDepth})$$

The trust level searching cost C_{STL} can be one of the three costs $C_{STL_{WF}}$, $C_{STL_{WOF}}$, or $C_{STL_{WWF}}$. Note that if NF > log N, $C_{STL_{WWF}}$ is equal to $C_{STL_{WF}}$, else it is equal to $C_{STL_{WOF}}$. In all cases $C_{STL_{WWF}}$ can be used for C_{STL}:

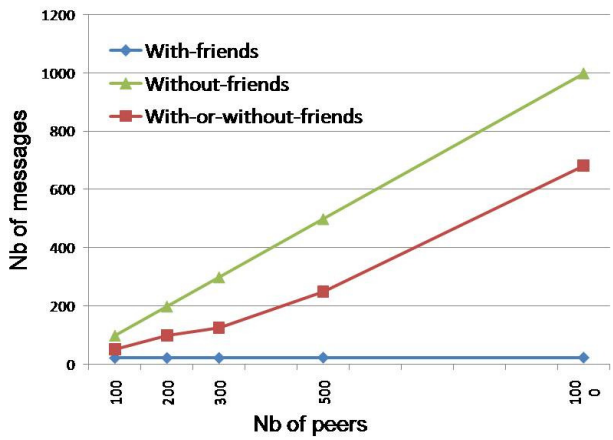


Figure 3.3: Comparison of the three algorithms of trust level searching

$$C_{STL} = O((max(logN, NF))^{MaxDepth})$$

To summarize,

$$C_{publishing} = O(nbData * nbPurpose * nbOperation * logN)$$

$$C_{Requesting} = C_{Request} + C_{Retrieve} = O(logN)$$
$$C_{STL} = O((max(logN, NF))^{MaxDepth})$$

For the simulation, we used SimJava [48] and the Chord protocol was simulated with some modifications in the put() and get() functions. Tests consider N peers, peer keys are selected randomly between 0 and 2^n. N is set to 11, which corresponds to 2^{11} peers. This number of peers is enought to simulate collaborative applications like the medical one. MaxDepth is set to 11 and the number of friends is set to 2.

Trust level searching introduces a large overhead because of flooding in the unstructured network. Figure 3.3 compares the three algorithms seen above. The with-friends case introduces the smallest cost while the without-friends case introduces the highest cost. However, intuitively, the probability to find the trust level is higher in the without-friends algorithm than in the with-friends algorithm. This is due to the fact that the number of contacted peers is higher in the without-friends algorithm, which increases the probability to find the trust level. We estimate that the with-or-without-friends algorithm is the most optimized because it is a tradeoff between the probability to find the requester trust level and the trust level searching cost.

Figure 3.4 shows that the trust level searching cost decreases with the number of requests and stabilizes. When peers ask for a trust level, answers are returned in the requesting order and the trust tables are updated with the missing trust level. Thus, the trust tables evolve with the number of searches. After a while, these tables stabilize. Thus, the number of messages for searching trust levels is reduced to a stable value. This value is not null because of the dynamicity of peers. Simulations consider that the number of peers joining the system is equal to those leaving the system. Thus, there are always new peers which do not know the requester trust level. We also observe in the figure that the trust level searching cost in the without-friends algorithm stabilizes first. This is due to the

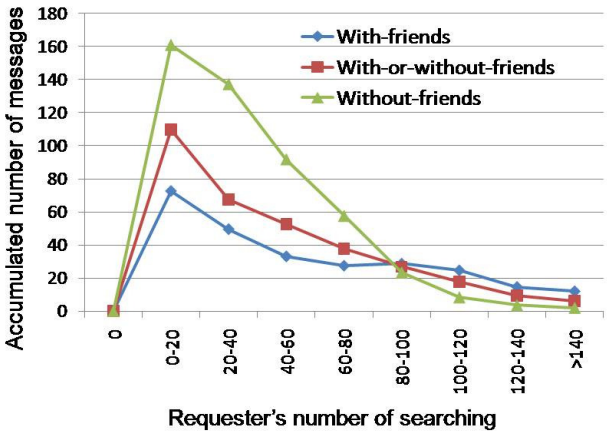


Figure 3.4: Stabilization of the cost of trust level searching

fact that a larger number of peers are contacted. The with-or-without-friends algorithm comes in second place, and the with-friends algorithm comes last. As can be seen in the comparison of the three algorithms, we find again that the with-or-without-friends algorithm is the most optimized because it is a tradeoff between the time to stabilization and the trust level searching cost.

PriServ prototype A prototype of PriServ for privacy-preserving data sharing applications for online communities was developed [57]. The prototype uses the Java language, SCA (Service Component Architecture) tools[2], and RMI (Remote Method Invocation).

PriServ was tested and validated by PeerUnit[3] on Grid5000[4]. Grid5000 is a scientific instrument for the study of large scale parallel and distributed systems. The tests were done on a population of 180 peers on 42 Grid5000 nodes.

PriServ is illustrated with a collaborative medical research application. The participants of this applicaiton are scientists, doctors, and patients. In order to control disclosure of sensitive information (*e.g.*, medical records owned by doctors, research results owned by scientists) without violating privacy, data access should respect the privacy preferences of their owners. In this medical PPA:

- Patients and doctors who own and manage private medical records can be considered as owners. Scientists who may use medical records for scientific research can be considered as requesters. Servers are peers of the storage system.
- Doctors may define the privacy preferences of patients in privacy policies and attach them to their medical records. For instance, a doctor may allow writing access on her information to scientists for adding comments on her diagnosis.
- Scientists may define their own privacy preferences and attach them to their research results. For instance, a scientist may allow reading access on her results to doctors for giving diagnosis.

[2] http://www.oasis-opencsa.org/sca
http://www.obeo.fr/pages/sca/
[3] http://peerunit.gforge.inria.fr/
[4] http://www.grid5000.fr/

PriServ is used for these scenarios:

- After defining their privacy preferences on their medical records, doctors and patients can publish their data in the system while preserving their privacy.
- Scientists can search for data by using procedures that respect the privacy preferences of the data owners.

Through a GUI, we show scenarios that exhibit important aspects of private data management: privacy policy management, data publishing, data searching, and data reference searching.

In [54, 57] you can find all the details about this prototype.

3.3 WUW: a distributed P2P framework to enhance users' satisfaction

Content Delivery Networks (CDN) [18] distribute content to end users as files (multimedia, software, documents), live-streaming, on demand streaming, *etc.* P2P architectures are increasingly used in CDN because traditional client-server architectures generate high distribution and maintenance cost, whereas in P2P systems those costs are almost negligible. Besides, P2P architectures are more performing than client-server ones when high demanded content has to be distributed in short period of time (*e.g.*, sportive or cultural events).

P2P systems are highly scalable because in those systems peers share their resources automatically (bandwidth, storage, *etc.*) and not only download content but also upload content to other peers (we call it data sharing). Thus the more peers are in the system, the more resources the system has and the more performing it is. The P2P architecture is built on top of a physical network. Peers are organized in overlays (neighborhoods) composed of peers sharing the same resources, for instance, peers downloading the same file or watching the same TV program.

We consider that peers are under control of users that are autonomous and have preferences, interests, *etc.* As users' resources are the richness of P2P systems, it is important to satisfy their preferences concerning the usage of their resources. As an example, consider a platform which builds user profiles containing the following information: geographic location, favorite music genre and a reputation score provided by an online community. Additionally, the platform is able to know with whom a given user has already exchanged content in previous sessions. Corresponding settings that users can configure could be: choice of the location of remote users, priority for unknown users, choice of affinity with remote users with respect to music tastes, choice of minimal reputation scores of remote users. A user could then chose users who like music that she likes, people with high reputation, living in a country where human rights are respected or where digital data protection is well regulated, *etc.*

In general, CDN applications based on P2P architectures do not take into consideration user preferences, but only QoS-related parameters. In my work, I propose WUW (What Users Want) [16, 17, 21], a framework that: (a) takes into account user preferences at each peer in order to shape neigbourhoods accordingly and (b) provides a quantified feedback

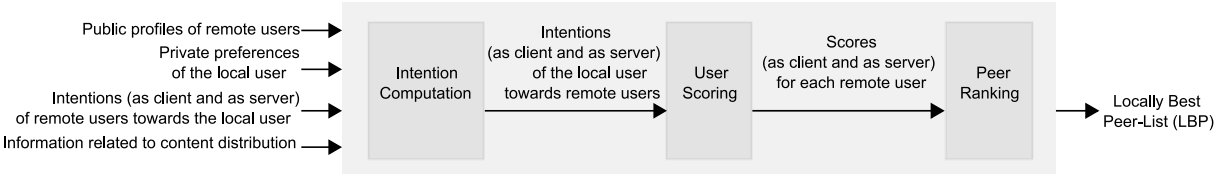


Figure 3.5: WUW Strategy.

to users, which are then able to evaluate the impact of their choices on their satisfaction and adequation to the system.

In the next, Section 3.3.1 introduces WUW definitions and Section 3.3.2 shows the implemented service and experimental results.

3.3.1 Defining strategies and their evaluation

Each authenticated user can chose to publicly disclose (completly or not) her profile that consists in a set of configurable personal preferences, QoS characteristics, and recent history of content exchanges. A peer collects available profiles of remote peers and compares them with his profile to stablish *strategies* for data sharing. A strategy defines a neigbourhood and can be defined for each peer as client, as server and by content. Each peer can define her local strategies and modify them if they are not satisfactory. To know if a strategy is satisfactory, periodically, WUW evaluates to wich extent, a local peer is effectively sharing content with peers she prefers the most and that were identified by her strategy. Such evaluation consists in feedbacks quantified in mesures about *satisfaction*, *adequation*, and *system evaluation*. They are inspired by the SQLB framework, but redefined to be computed in a fully decentralized context.

Strategy definition. With WUW, a peer *strategy* is stablished in three steps as shown in Figure 3.5.

1. *Intention computation* maps a local complet peer profile with every remote public peer profile and produces *intentions* whose values range from -1 to 1. Intentions are computed for the local peer as client and as server and represent to which extent the local peer whishes to share a content with every remote peer. The less the local user wish to exchange with a remote user, the smaller the value of her intentions towards him. Intentions are public whereas the complete profile of user may not be public.

2. *User scoring* assigns scores to remote users, as clients and as servers, according to the intentions of the local peer (towards remote peers) and the ones of remote peers (towards the local peer). Thus, to calculate the scores of a local user as client (resp. as server), for every known peer, WUW uses the intentions of remote users as servers (resp. as clients) and the intentions of the local user as client (resp. as server)

3. *Peer ranking* assigns to remote peers a position in a ranking. The K best ranked peers will compose the *Locally Best Peer-list* (LBP) for a content. The $LBPs$ for all contents to be shared are given to the P2P application to "tune" local neighborhoods.

Feedback computation It is important to understand to which extent user preferences and her strategies are effective, *i.e.*, they are making the local peer to share content with the K best ranked peers o her $LBPs$.

To evaluate to which degree the P2P application considers the local user preferences during the content exchange, WUW computes a $feedback$ periodically. To do this, WUW needs information about the content exchange (download/upload activities) made in the P2P application. For that, we assume that any content C can be logically split in a set of *items* $i_1, ..., i_n$ and the content exchange is made by item. The feedback measures are defined in terms of $satisfaction$, *adequation* and *system evaluation*. They are computed periodically, every n seconds, for the last download/upload activities. Like intentions, each measure is computed twice:

- the *satisfaction as a client* of a local user measures to which extent neighbors she prefers the most were chosen by the P2P application to give her content;
- the *satisfaction as a server* of a local user measures to which extent neighbors she prefers the most were chosen by the P2P application to receive her content;
- the *adequation as a client* of a local user measures to which extent pieces requested (and downloaded) by her were available for downloading among neighbors she prefers the most;
- the *adequation as a server* of a local user measures to which extent pieces she has were requested (and uploaded) for uploading by neighbors she prefers the most.

Feedback values can vary between 0 and 1. Intuitively, for satisfaction, 1 denotes the best choices were made and the peer is completely satisfied by the P2P system. For adequation, 1 denotes the local peer is giving the best scores to peers it is exchanging with.

Finally, we calculate a *system evaluation* that measures how much the local user can be happy about the impact of her strategy on the ongoing content distribution, both as client and as server. Its value may vary in the interval $[0...\infty]$ (the higher, the better), with 1 denoting a neutral impact.

In the following I introduce more formally these measures. Resulting value may vary between 0 and 1, with 1 denoting the best possible matching are always made by WUW. Table 3.2 introduces the used notation where I consider a content C as a set of items $i_1, \ldots, i_n$.

The *Satisfaction* S_c of a (local) user as client (*i.e.*, as a "downloader") is computed as follows. For each item $i \in C$ whose download the local user has completed, let $S_c[i]$ be the sum of the local user's intentions towards the users who have provided her item i, multiplied by the number of successful download events related to i, divided by the number of times i, or part of it, has been requested by the local user. That is :

$$S_c[i] = \frac{\sum_{u \in P_i}(((Ic_C^u + 1)/2) \cdot \mid D_i^u \mid)}{\mid LQ_i \mid} \tag{3.1}$$

Then S_c is the average computed by aggregating the values $S_c[i]$ of the latest items downloaded by the local user:

$$S_c = \frac{\sum_i S_c[i]}{\mid D \mid} \tag{3.2}$$

Symbol	Meaning
H_i	Set of all remote users who currently have item i
P_i	The set of users who provided item i ‡
Ic_C^u	The local user's Intention "as client" toward the remote user u for content C ‡
Is_C^u	The local user's Intention "as server" toward the remote user u for content C ‡
D_i^u	Set of all download events from a remote user u related to item i ‡
D	Set of all download events ‡
LQ_i	Set of all the request events issued by the local user related to item i ‡
LQ	Set of the request event issued by the local user ‡
RD_i^u	Set of all the complete download events to a remote user u related to item i ‡
RD_i	Set of all the complete download events to remote users related to item i ‡
RD	Set of all the complete download events to remote users ‡
RQ_i^u	Set of all the request events issued by a remote user u related to item i ‡
RQ_i	Set of all the request events issued by remote users related to item i ‡
RQ	Set of all the request events issued by remote users ‡

Table 3.2: Notation used to describe feedback measures computation on each peer.

The Satisfaction S_s of a (local) user as server (*i.e.*, as an "uploader"), is instead computed as follows. For each item $i \in C$ whose upload the local user has completed, let $S_s[i]$ be the sum of the local user's intentions towards the users who have downloaded item i from her, multiplied by the number of successful remote download events (upload events from the local user point of view) related to i. That is :

$$S_s[i] = \sum_{u \in RD_i} (((Is_C^u + 1)/2) \cdot \mid RD_i^u \mid) \tag{3.3}$$

Then S_s is the average computed by aggregating the values $S_s[i]$ of the latest items uploaded by the local user:

$$S_s = \frac{\sum_i S_s[i]}{\mid RD \mid} \tag{3.4}$$

The *Adequation* A_c of a (local) user as client (*i.e.* as a "downloader") is computed as follows. For each item $i \in C$ for which a request has been issued by the local user, let $A_c[i]$ be the average of the local user's intentions towards all the users who currently have item i. That is :

$$A_c[i] = \frac{\sum_{u \in H_i}((Ic_C^u + 1)/2)}{\mid H_i \mid} \tag{3.5}$$

Then A_c is the average computed by aggregating the values $A_c[i]$ of the latest requests issued by the local user:

$$A_c = \frac{\sum_i A_c[i]}{\mid LQ \mid} \tag{3.6}$$

The Adequation A_s of a (local) user as server (*i.e.* as an "uploader"), is instead computed as follows. For each item $i \in C$ whose upload has been requested to the local user, let $A_s[i]$ be the sum of the local user's intentions towards the users who have requested item i, multiplied for the number of request events related to i. That is:

$$A_s[i] = \sum_{u \in RQ_i} (((Is_C^u + 1)/2) \cdot \mid RQ_i^u \mid) \tag{3.7}$$

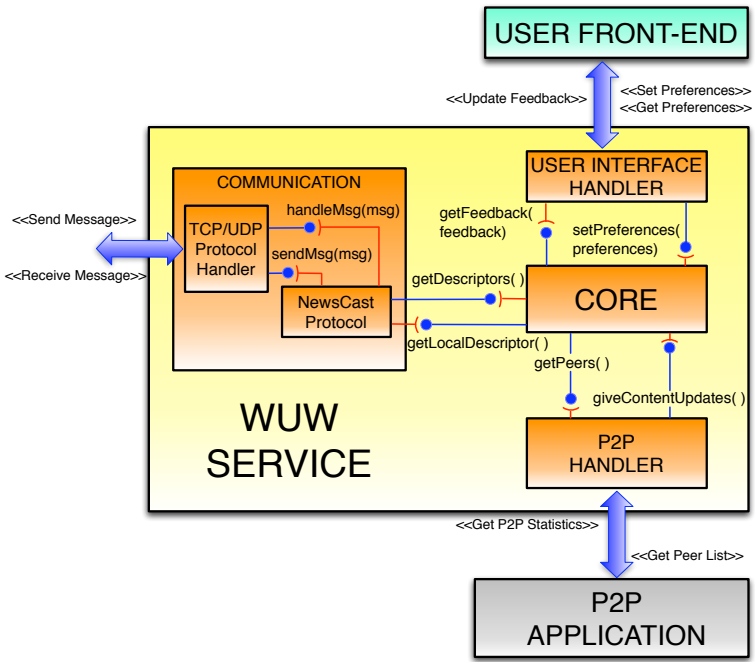


Figure 3.6: WUW Architecture

Then A_s is the average computed by aggregating the values $A_s[i]$ of the latest requested items:

$$A_s = \frac{\sum_i A_s[i]}{\mid RQ \mid} \tag{3.8}$$

Finally, at any given time, the *System Evaluation* of a (local) user as client (respectively: server) is calculated as the ratio S_c/A_c (respectively: S_s/A_s).

3.3.2 Implementing WUW

As proof-of-concept, we implemented WUW on top of BitTorrent [23, 24], an efficient P2P protocol for content distribution. Our experiments were conducted on the French testbed Grid5000 [20].

Service design and implementation WUW functionalities are organized in different modules, Figure 3.6 shows the WUW architecture. It acts as a middleware between the local instance of a P2P application and the overlay management system of the P2P network. The P2P application communicates with WUW to know about other peers in the overlay and WUW communicates with the overlay coordinator (*e.g.*, a tracker, a DHT protocol, or other, depending on the P2P system being used) to get information about the state of the overlay. To be able to influence the list of peers used by the local P2P application, WUW intercepts the communications between the local peer and the overlay coordinator and modifies the peer list by only including the peers in its current LBP. WUW is implemented as a multithread application thus any information exchange among the modules is asynchronous and concurrent.

In the next, each WUW module is described.

The User Interface Handler module The primary source of information for WUW is the local user. The User Interface Handler (UI Handler) module takes care of getting input from the user (her profile and preferences) and outputs the computed feedback measures. This module is independent from other WUW modules, so it is possible to implement different user interfaces at will.

The Communication module The Communication module implements basic functions to facilitate communications, over TCP or UDP, among WUW instances at different peers.

Moreover, an epidemic protocol is provided to disseminate among the peers: user intentions, user public profiles and additional data related to content dissemination, *e.g.* the number of content items at each peer. The epidemic dissemination of up-to-date information makes it possible for any local user to know what are the recently computed intentions of remote users towards her and what is the state of the diffusion of a given content.

The overlay used by the epidemic protocol is different from the content distribution overlay and it is locally maintained at each peer with negligible overhead.

The P2P Handler module This module is the only part of WUW that is aware of the specific P2P application being used by the local user. The interfaces provided by this module allow other modules to transparently exchange information with a specific content distribution protocol.

The P2P Handler module contains the logic that defines content items and their parts in a way that is consistent with the P2P application.

It is in charge of getting information from and giving information to the rest of the P2P system. This includes the communication with the P2P overlay manager, concerning the global list of peers sharing a given content. For each content being shared and for each neighbor in the overlay, the P2P Handler module retrieves data about any distinct download or upload event from/to a given neighbor, related to a given item. Finally, through this module WUW sends to the P2P application the periodically renewed LBP for a given content.

The Core module. The main functionalities of WUW are placed in this module, which is the orchestrator of the service. User intentions and feedback measures are computed in a timely way. Every n seconds, a routine is started which performs the following steps:

1. Gets updates about the remote users from the Communication module.
2. Gets the latest information about the activity of the local P2P application from the P2P Handler module.
3. Gets the latest changes in the users preferences from the UI Handler module.
4. Updates the local state with the collected information.
5. Computes the feedback measures related to the latest local activity, considering the current intentions for all the neighbors and all the contents.
6. Makes the feedback measures available to the local user via the UI Handler module.

7. Computes the intention values to be associated to every neighbor for every content, according to the local strategy.

8. Builds a global ranking of all neighbors, considering the associated intentions for all the contents.

9. Creates a *LBP* for each content currently shared by the local user.

10. Sends the *LBPs* to the P2P application, via the P2P Handler module.

Our WUW prototype is implemented in Java and requires JVM 1.7 or higher. The source code is available under GPL license on Github[5].

The current implementation features a P2P Handler module that is able to interact with BitTorrent MainLine 3.1.9 (in the following: BT). We provide an instrumentation class in MainLine that periodically collects the required information from the working memory of the BT process and sends it to WUW via a local socket.

The Communication module hosts the NewsCast epidemic protocol [58] to maintain the overlay used to disseminate user profiles, intentions and content related information.

The UI Handler module is currently a facility to handle the service configuration via XML files.

Experimental results. In our experiments we deployed WUW on the Grid5000 platform. The goal is to know the potential impact of WUW on the time to complete the download of a content and the effectivennes of WUW in improving users' satisfaction.

We consider a P2P network of 300 peers. Each peer runs an instance of the WUW service and an instrumented instance of BT. Only one target content is considered, a file of 1.1 GB. A standard BT tracker is also deployed as overlay manager. 30 peers are seeders (they have the full content since the beginning) and 270 peers are leechers. The download and upload bandwidth of each peer is set to 1024 and 512 kbps correspondingly. The number of maximal connections by peer is 15. At each peer, WUW's epidemic protocol sends information about the known peers to one randomly chosen peer every 3 seconds, while the feedback measures and the peer lists are updated every 8 seconds.

We implemented a simple strategy that considers public profiles represented by a single integer value in [0..4]. In our experiments, we introduce a balance among the type of users (same number of users of each profile) and users prefere other user with the same profile or close to them. The closest the users' profiles, the bigger the intentions. Past failed exchanges between peers influence negatively their intentions.

Our experiments each peer knows (thanks to the tracker) a list of 40 random peers. Such list can be updated every 100 seconds. We consider two different scenarios:

- *Scenario 1* : standard BT execution. WUW gives to BT 15 peers randomly chosen to exchange content. In this scenario, WUW only computes intentions and monitores user satisfaction and adequation.

- *Scenario 2* : WUW in action. The first time WUW gives to BT 15 peers, among which 5 are the best according to its locally computed ranking, and the remaining

[5] http://github.com/marbiaz/WhatUsersWant

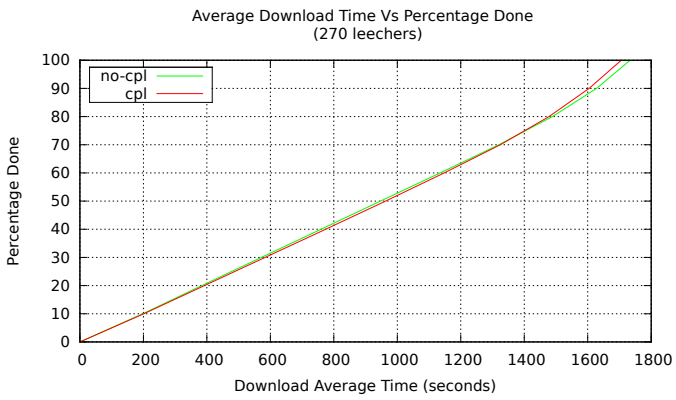


Figure 3.7: Average download time.

10 are chosen at random. All the other times, WUW gives to BT the best 15 peers in its updated ranking.

For both scenarios we measure: time to complete the download, average satisfaction and average adequation in the overlay. For each scenario 20 runs are executed. We then take the averages of the measured values.

Fig. 3.7 shows the average time taken by the peers in the overlay to complete the download of the target file. The parametrization of our strategy makes it possible to obtain almost identical performance in the two scenarios. Tinkering with local overlays usually worsen overall performance, *i.e.*, dropping connections and creating new ones is extremely expensive. Nonetheless, this result shows that it is actually possible to obtain good performance and improve users' satisfaction if there is the possibility to tune the overlay in a way that accounts for the relations among the number of peers, their average amount of bandwidth and the size of their local neighborhoods.

Fig. 3.8 to 3.10 show aggregate values (minimum, 0.25 percentile, median, 0.75 percentile and maximum) of WUW feedback measures in the overlay at different time snapshots. The two experimental scenarios are thus compared with respect to the same measure at the corresponding time.

Figure 3.8 clearly shows the positive impact of WUW on user satisfaction as client. All relevant aggregate values are better in Scenario 2 than in Scenario 1. The flooring effect that is visible at $satisfaction = 0.5$ is due to the definition of our strategy.

Figure 3.9 demonstrates a similar effectiveness with respect to the satisfaction of users as servers. In this case, though, the values measured in Scenario 2 are lower at the beginning. From the standpoint of uploaders, a very small overlay, not optimized by WUW, is of limited satisfaction. The values rapidly improve when the BT instances receive more and better peers (after 100 and 200 seconds).

Adequation as server (not shown) globally follows a pattern that is very similar to the one illustrated so far. Nevertheless, adequation as client, shown in Figure 3.10, does not improve in Scenario 2. Adequation as client is better for Scenario 1. We see here that the adequation reflects the extreme ease of the downloading task, determined by our settings and resulting in a faultless and over-provisioned network of peers. That is, if a content (or at least the part of it that is currently requested) is readily available from a large number of peers in the overlay, the profiles of the users that have it are, on average, not

significantly better for any particular user. In our setting, at a given time the same items are available from users whose position in the local ranking at each peer is quite different. Thus the adequation as client of each user, measured on the intentions towards those who can provide these items, cannot improve that much by tuning local neighborhoods.

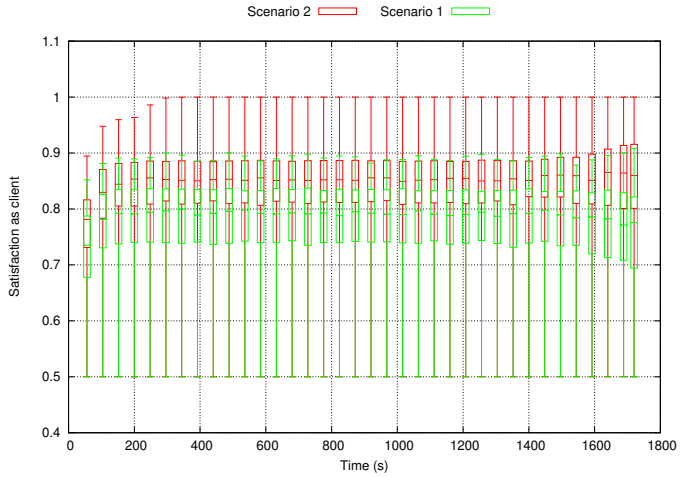


Figure 3.8: Satisfaction as client.

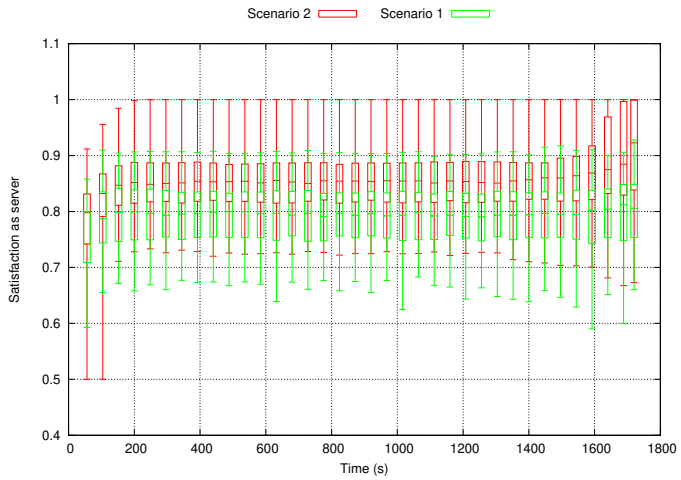


Figure 3.9: Satisfaction as server.

3.4 Conclusion

In this chapter, I presented my research contributions to the P2P context concerning data privacy and user satisfaction.

I presented a complete solution (Primod and Priserv) for data privacy in P2P systems that supports the notion of purpose of HDB. PriMod, a privacy model for P2P systems, integrates the purposes notion as mainspring. Purposes are omnipresent in several process of sensitive data management. Data owners specify, through personal privacy policies, the access purpose for their data. Data publication attaches the allowed access purposes.

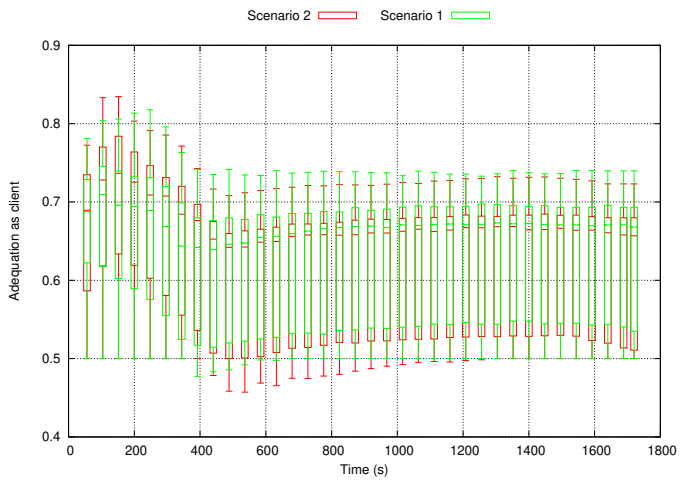


Figure 3.10: Adequation as client.

Data requesters specify the access purpose in their requests, thus they are committed to their intended and expressed use of data.

PriServ is a privacy service that implements PriMod. The PriServ prototype combines purpose and operation-based access control, trust techniques, cryptography techniques, and digital checksums. A privacy-preserving data sharing application for online social networks illustrates this approach.

Several improvements can be made to PriMod and PriServ. The purpose-based index should be anonymized to avoid servers to know the partial view of the index they store. A more semantically rich query language may also be proposed. But above all, auditing solutions should be proposed to verify compliance of data use with the specified privacy preferences. This is still an open and challenging issue.

Concerning user satisfaction, I presented WUW (What Users Want), a framework defining concepts to achieve, in a fully decentralized way, The goal is to take into account user preferences at each peer in order to shape distribution overlays accordingly and to provide a quantified feedback to users, which are then able to evaluate the impact of their choices on their satisfaction and adequation to the system. We detailed the procedures through which WUW makes it possible to evaluate users, characterized by heterogeneous profiles, with respect to personal, undisclosed preferences. We described the design of a P2P service, which implements our framework and is oriented to unstructured P2P overlays. Finally, we presented and discussed experimental results, obtained with a prototype implementation of the WUW service, deployed on a distributed grid platform. They picture our first steps in the exploration of the various issues raised by the relations between performance and overlay tuning.

Our future work will evolve along three main directions. The first is related to extending the support to different P2P applications. The second concerns the experimental study of ways to tune local neighborhoods towards greater user satisfaction, while affecting performance in a way that can be estimated or controlled. The third is the characterization of different interesting use case scenarios, *e.g.*, scenarios (a) in which satisfaction and adequation enlighten differently the state of user contribution (low satisfaction, high adequation), (b) where users dynamically change their preferences in order to ameliorate their satisfaction and adequation during the distribution of a given content,

(c) with multiple contents and where the intersection of the $LBPs$ (Locally Best Peer-list) is not empty, (d) with different strategies per user, (e) with large number of peers and occurrence of churn.

Chapter 4

Deducing BGPs from logs of Linked Data providers

Contents

4.1 Introduction **57**
4.2 FETA: a reverse function **59**
4.2.1 Motivating example and problem statement 59
4.2.2 FETA algorithms 62
4.2.3 Experiments 65
4.3 SWEEP, BGP deduction from a streaming log of TPF servers **69**
4.3.1 Motivation 70
4.3.2 SWEEP in a nutshell 71
4.3.3 Implementation and demonstration 72
4.4 Conclusion **72**

4.1 Introduction

The Semantic Web builds on top of the Web by *semantically describing* its resources (Web pages, XML documents, music, movies, *etc.*). Semantic descriptions are statements about Web data, written in terms of relations provided by an ontology which is a formal specification of an application domain. RDF (Resource Description Framework)[1] is a graph-based data model for managing data through simple ontologies. The RDF data model allows writting labeled graphs using triples (*subject, predicate, object*). A subject can be a URI or a blank node (unnamed URI). A property or predicate can be a URI. An object can be a URI, blank node, or literal (string). A triple can be seen as a pair of entities connected by a named relationship. OWL (Web Ontology Language)[2] and RDFS (RDF Schema)[3] are Semantic Web languages designed to represent rich and complex knowledge

[1] https://www.w3.org/RDF/
[2] https://www.w3.org/OWL/
[3] https://www.w3.org/TR/rdf-schema/

about things, groups of things, and relations between things. They are computational logic-based languages such that knowledge can be exploited by computer programs, *e.g.*, to verify the consistency of that knowledge or to make implicit knowledge explicit.

The *Web of data*, or Linked Data (LD) results from applying these technologies. It interlinks massive amounts of datasets across the Web in multiple domains like life science, government, social networking, media and publications. The Web of data is queried using SPARQL, a SQL-like langage[4]. At the core of SPARQL are Basic Graph Pattern (BGP) queries, *i.e.*, conjunctive queries. A BGP is a set of *triple patterns* modeling their conjunction. A triple pattern is like an RDF triple but variables can exist in the subject, the predicate or the object. Query evaluation consists in matching BGPs against the explicit triples of a graph.

RDF data can be accessible as data dumps or through live queryable interfaces like SPARQL endpoints or Triple Pattern Fragments (TPF) [114]. SPARQL endpoints are RESTful services that accept queries over HTTP written in SPARQL and adhering to the SPARQL protocol[5]. TPF instead, is a low-cost server interface that only accepts single triple patterns over HTTP.

Live queryable interfaces make possible distributed query processing over the Web of data. Given a SPARQL query and a virtually defined federation of SPARQL endpoints, a federated query engine performs the following tasks [68, 95]. (i) *query decomposition*: normalizes, rewrites and simplifies queries; (ii) *data localization*: performs source selection among a defined federation and rewrites the query into a distributed query (*i.e.*, set of subqueries annotated with sources that can evaluate them); (iii) *global query optimization*: optimizes the distributed query by rewriting an equivalent distributed query using various heuristics like minimizing intermediate results, minimizing number of calls to endpoints, *etc.*; (iv) *distributed query execution*: executes the optimized query plan with physical operators available in federated query engines.

Query engines [2, 15, 37, 44, 99, 107, 114] allow data consumers to execute SPARQL queries over a decentralized federation of servers maintained by LD providers. However, due to the way distributed query processing is done on the Web of data, data providers are not aware of users' queries; they just observe subqueries they receive on their servers. Thus, they have no idea about the usage of data they provide, *i.e.*, when and which datasets are joined with their own datasets. The knowledge of what queries do with datasets is essential for tuning servers, justify return of investment or better organize collaboration among providers.

A simple solution for this problem is to consider that data consumers publish their queries. However, public queries cannot be considered as representative of real data usage because they may represent a small portion of really executed queries. Only logs give evidences about real execution of queries.

Thus, in this chapter, I address the following problem: if LD providers share their logs, can they infer the BGP of queries executed over their federation? Many works have focused on web log mining, but none has addressed reversing BGPs from a federated query log. I focus on this problem in two query processing contexts: in a federation of SPARQL endpoints [2, 107] and in Triple Pattern Fragments [114].

[4] https://www.w3.org/TR/rdf-sparql-query/
[5] https://www.w3.org/TR/rdf-sparql-protocol/

The chapter is organized as follows. Section 4.2 introduces FETA, a motivating example, the proposed algorithms, and our experimental study. Section 4.3 presents SWEEP, a motivating example, our approach and a demonstration. Section 4.4 concludes.

Contributions of this chapter were published in [27, 90, 91, 92, 93].

4.2 FETA: a reverse function

In this section I introduce FETA, a Federated quEry TrAcking system that computes BGPs from a federated log produced by SPARQL endpoints. Based on subqueries contained in a log, FETA deduces triple patterns and joins among triple patterns with a good precision and recall. Our main contributions are:

1. the definition of the problem of reversing BGPs from a federated log,
2. the FETA algorithm to reverse BGPs from federated logs,
3. an experimental study using federated queries of the benchmark FedBench [106]. From execution traces of these queries, FETA deduces BGPs under two scenarios, queries executed in isolation and in concurrence.

4.2.1 Motivating example and problem statement

In Figure 4.1, two data consumers, C_1 and C_2, execute concurrently federated queries $CD3$ and $CD4$ of FedBench [106]. They may use the Anapsid or FedX federated query engines over a federation of SPARQL endpoints composed of *LMDB, DBpedia InstanceTypes, DBpedia InfoBox* and *NYTimes*. CD3 asks for presidents of United States, their party and pages of the New York Times talking about them. Whereas CD4 asks for pages of the New York Times talking about actors apearing in the Tarzan movie. Notice that tp_4 and tp_5 in both queries have the same predicate with different variables in subjects and objects.

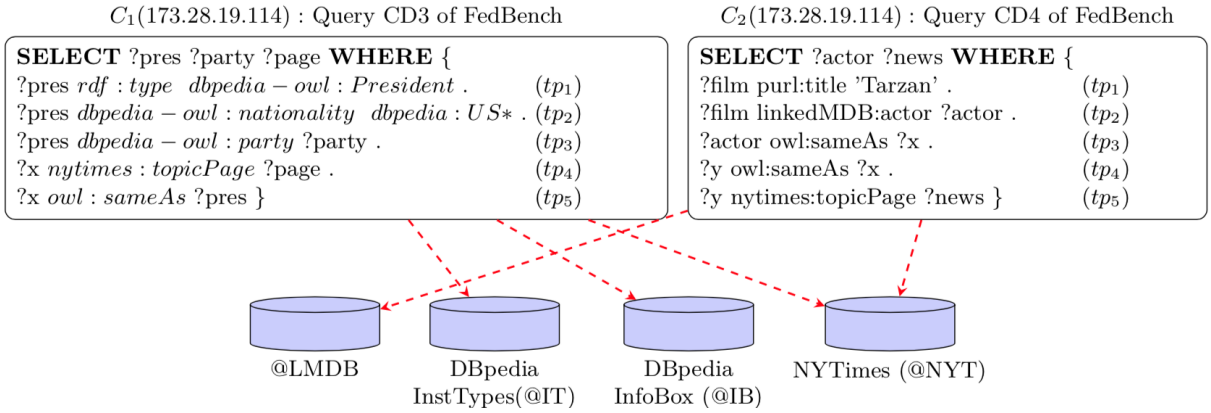


Figure 4.1: Concurrent execution of federated queries CD3 and CD4 of FedBench over a federation of SPARQL endpoints.

Federated query engines, decompose SPARQL queries into a sequence of subqueries as partially presented in Tables 4.1 and 4.2 for query CD3 using Anapsid and FedX, respectively. Even-numered lines are the subqueries received by the SPARQL endpoints

and odded-numered ones correspond to answers sent for the queries. As this federation of SPARQL endpoints receive only subqueries corresponding to physical operators implementing the query plan, the original queries e.g., CD3 and CD4 remain unknown to the corresponding data providers. For example, $CD3$ can be decomposed into $\{tp_1^{@IT}.(tp_2.tp_3)^{@IB}.(tp_4.tp_5)^{@NYT}\}$. So, NYT data provider just observes tp_4 and tp_5 and does not know these triple patterns are joined with tp_1 from IT and tp_2, tp_3 from IB. Consequently, data providers do not know the real usage of data they provide.

In this work, we address the following research question: *if data providers share theirs logs, can they reverse engineer BGPs and sources that evaluated each triple pattern?* On the previous example, we aim to extract two BGPs: one corresponding to $CD3$ $\{tp_1^{@IT}.(tp_2.tp_3)^{@IB}.(tp_4.tp_5)^{@NYT}\}$ and another to $CD4$ $\{(tp_1.tp_2.tp_3)^{@LMDB}.(tp_4.tp_5)^{@NYT}\}$.

	IP	Time	Subquery/Answer
1	173...	11:24:19 @IT	SELECT **?pres** WHERE { ?pres rdf:type dbpedia-owl:President }
2	173...	11:24:23 @IT	{{"pres", "http:.../Ernesto_Samper" } {"pres", "http:.../Shimon_Peres" } {"pres", "http:.../Barack_Obama" },...}
3	173...	11:24:21 @IB	SELECT **?party ?pres** WHERE { ?pres dbpedia-owl:nationality dbpedia:United_States . ?pres dbpedia-owl:party ?party }
4	173...	11:24:24 @IB	{{{"party","http:.../Democratic_Party_%28United_States%29" }{"pres", "***http:.../Barack_Obama***" }}, {{"party","http:.../Democratic_Party_%28United_States%29" }{"pres", "***http:.../Johnny_Anders***" }}, {{"party","http:.../Republican_Party_%28US%29" }{"pres", "***http:.../Judith_Flanagan_Kennedy***" }},...}
5	173...	11:24:25 @NYT	SELECT **?pres** ?x ?page WHERE { ?x nytimes:topicPage ?page . ?x owl:sameAs ?pres . FILTER ((***?pres=<http:.../Barack_Obama>***) || (***?pres=<http:.../Johnny_Anders>***) || (***?pres=<http:.../Judith_Flanagan_Kennedy>***),...) || }} LIMIT 10000 OFFSET 0
6	173...	11:24:27 @NYT	{{"pres","http:../Barack_Obama" }{"x","http://data.nytimes.com/47452218948077706853" }{"page", "http://topics.nytimes.com/.../barack_obama/index.html"}}

Table 4.1: Partial federated log of CD3 executed by Anapsid ($E_{Anapsid}(CD3)$) where can be observed $\{tp_1^{@IT}.(tp_2.tp_3)^{@IB}.(tp_4.tp_5)^{@NYT}\}$.

We consider that an execution of a federated query FQ_i produces a partially ordered sequence of subqueries SQ_i represented by $E(FQ_i) = [SQ_1, ..., SQ_n]$. Each subquery is processed by one endpoint of the federation at a given time represented by a timestamp. We suppose that endpoint's clocks are synchronized, i.e., timestamps of logs can be compared safely. Timestamps of subqueries in a federated log are partially ordered because two endpoints can receive queries at same time. Query execution with a particular federated query engine, qe_i, is represented by $E_{qe_i}(FQ_i)$. So Tables 4.1 and 4.2 present a federated log with a subset of subqueries of $E_{Anapsid}(CD3)$ and $E_{FedX}(CD3)$ correspondingly. We represent concurrent execution of n federated queries by $E(FQ_1 \parallel ... \parallel FQ_n)) = [SQ_1, ..., SQ_n]$. In our example of Figure 4.1, data providers will observe $E(CD3 \parallel CD4)$.

Definition 11 (reverse function)**.** *Given a federated log corresponding to the execution of one federated query $E(FQ_i)$, find a function $f(E(FQ_i))$ producing a set of BGPs $\{BGP_1, ..., BGP_n\}$, where each triple pattern is annotated with the endpoints that evaluated it, such that $f(E(FQ_i))$ approximates ($\approx$) the BGPs existing in the original federated query. Thus, if we consider that $BGP(FQ_i)$ returns the set of BGPs of FQ_i: $f(E(FQ_i)) \approx BGP(FQ_i)$.*

We consider two BGPs are similar ($\approx$) if they contain same triple patterns and thus same joins. So the quality of f can be evaluated with precision and recall of triple patterns

	IP	Time	Subquery/Answer
1	173...	17:04:08 @IB	SELECT `?pres ?party` WHERE {?pres owl:nationality <http://dbpedia.org/.../United_States> . ?pres owl:party ?party }
2	173...	17:04:10 @IB	{{{ "pres", "***http:.../Barack_Obama***"} { "party", "http:.../Democratic_Party_%28United_States%29" }}, {{ "pres", "***http:.../Johnny_Anders***" } { "party", "http:.../Independent_%28politics%29" }}, {{ "pres", "***http:.../Judith_Flanagan_Kennedy***" } { "party", "http:.../Republican_Party_%28US%29" }},...}
3	173...	17:04:11 @IT	SELECT `?o_0 ?o_1 ?o_2` WHERE { {***<http:.../Barack_Obama>*** rdf:type ?o_0. FILTER (?o_0 = <http://dbpedia.org/ontology/President>) } UNION {***<http:.../Johnny_Anders>*** rdf:type ?o_1. FILTER (?o_1 = <http://dbpedia.org/ontology/President>) } UNION {***<http:.../Judith_Flanagan_Kennedy>*** rdf:type ?o_2. FILTER (?o_2 = <http://dbpedia.org/ontology/President>) },... }
4	173...	17:04:12 @IT	{{ "o_0", "http://dbpedia.org/ontology/President" } { "o_1", "" } { "o_2", "" },...}
5	173...	17:04:13 @NYT	SELECT `?x` WHERE { ?x owl:sameAs ***<http:.../Barack_Obama>***. }
6	173...	17:04:14 @NYT	{{ "x", ***"http://data.nytimes.com/47452218948077706853"***}}
7	173...	17:04:15 @NYT	SELECT `?page` WHERE { ***<http://data.nytimes.com/47452218948077706853>*** nytimes:topicPage ?page }
8	173...	17:04:16 @NYT	{{ "page", "http://topics.nytimes.com/.../barack_obama/index.html" }}

Table 4.2: Partial federated log of CD3 executed by FedX ($E_{FedX}(CD3)$) where can be observed $\{(tp_2.tp_3)^{@IB}.tp_1^{@IT}.tp_5^{@NYT}.tp_4^{@NYT}\}$.

and joins returned by f against those existing in original queries. For example, if $f(E(Q_1))$ returns one BGP with two triple patterns $\{\{(?x\ p1\ ?y)^{@NYT}.\ (?y\ p2\ ?z)^{@IT}\}\}$, while real query execution was two BGPs $\{\{(?x\ p1\ ?y)^{@NYT}\}, \{(?y\ p2\ ?z)^{@IT}\}\}$ then precision and recall in terms of triple patterns is perfect while precision and recall in terms of joins is 0 and ∞ respectively. If $f(E(CD3))$ produces $\{tp_1^{@IT}.(tp_2.tp_3)^{@IB}.(tp_4.tp_5)^{@NYT}\}$, then precision and recall are perfect according to BGPs present in CD3.

In our motivating example, if C_1 and C_2 have different IP addresses, then it is possible to split $E(CD3 \parallel CD4)$ into $E(CD3)$ and $E(CD4)$ and apply the reverse function. However, in the worst case, C_1 and C_2 have the same IP address. In this case, we expect that $f(E(CD3 \parallel CD4)) \approx f(E(CD3) \cup f(E(CD4)$.

Definition 12 (resistance to concurrency)**.** *The reverse function f should guarantee that BGPs obtained from execution traces of* isolated *federated queries, approximate ($\approx$) results obtained from execution traces of* concurrent *federated queries :* $f(E(FQ_1)) \cup ... \cup (f(E(FQ_n)) \approx f(E(FQ_1 \parallel ... \parallel FQ_n))$.

Federated query engines implement joins through physical operators, from which, the most implemented are *symmetric hash* and *nested-loop* joins. Both can be implemented as pipelined operators and produce results as early as possible. Symmetric hash join [117] maintains one hash table on each input and at the same time it uses each triple from either input dataset to probe the hash table of the other dataset. Nested-loop join [40, 81], in a double iteration, uses each triple of the outer dataset to search matching triples in the inner. For nested-loops to be effective, the outer is the smaller and the inner is the largest input. In the pipelined implementation, of the outer dataset are progressively used to probe matching triples in the inner, without waiting all outer triples to be extracted locally at the client.

@ep1
s1 p1 o1
s2 p1 o2
s3 p1 o2

(a) EP1

@ep2
s2 p2 o3
s3 p2 o4

(b) EP2

gap	reversed BGPs
∞	{ (?x p1 ?y)$^{@ep1}$. (?x p2 ?y)$^{@ep2}$. (?z p1 o2)$^{@ep1}$. (?z p2 ?y)$^{@ep2}$ }
1	{ ?x p1 ?y }$^{@ep1}$ { ?z p1 o2 } $^{@ep1}$ { s2 p2 ?y }$^{@ep2}$ { s3 p2 ?y }$^{@ep2}$
2	{ ?x p1 ?y }$^{@ep1}$ { (?z p1 o2)$^{@ep1}$. (?z p2 ?y)$^{@ep2}$ }

(c) deducted BGPs

time	subquery	bindings
$1^{@ep1}$	$sq1$={?x p1 ?y}	$\mu1$={{x, s1}{y, o1},{x, s2}{y, o2},{x, s3}{y, o2}}
$4^{@ep1}$	$sq2$={?z p1 o2}	$\mu2$={{z, s2},{z, s3}}
$6^{@ep2}$	$sq3$={s2 p2 ?y}	$\mu3$={{y, o3}}
$7^{@ep2}$	$sq4$={s3 p2 ?y}	$\mu4$={{y, o4}}

(d) federated log

Figure 4.2: Reversing BGPs from a federated log.

Finding a reverse function f is challenging. *It requires to interlink constants or variables from different subqueries.* In the general case, linking constant or variables from different subqueries is ambiguous as explained in Figure 4.2. Figures 4.2a and 4.2b present two endpoints, each one providing some triples. Figure 4.2d has the federated log corresponding to the execution of the queries

$Q_1 =$ SELECT ?z ?y WHERE {?z p1 o2 . ?z p2 ?y} and

$Q_2 =$ SELECT ?x ?y WHERE {?x p1 ?y}.

Figure 4.2c shows reversing results according to different values for a "gap" described below.

From a syntactic analysis of the log, there are 3 possible joins on variable $?y$. Joins may be possible if $?y$ mappings concern same concepts (ontologically), which is the case here. But analyzing more deeply, if we try a join on mappings on $?y$, we obtain an empty result set. So, there are no joins on $?y$. If we analyze the subjects of $sq3$ and $sq4$, they have been returned in mappings of previous subqueries; first in a mapping of $?x$ then in a mapping of $?z$. So there is a possibility of two nested-loops; $sq3, sq4$ with $sq1$ but also with $sq2$. First line of Figure 4.2c shows the reversed BGP where the two last triple patterns represent the inner part of these nested-loops and where $s2, s3$ were replaced by variables $?x$ and $?z$ respectively.

As a real log can be huge, it is not possible to take into account all log entries in the analysis of each subquery, so it is necessary to establish a maximal *gap of time*. Depending on this gap, on verifications made, and concurrent traces, reversed BGPs are different. In our example, if the gap is 1, only $sq3$ and $sq4$ can be analyzed together. As the join on $?y$ gives no results, a join is discarded. If the gap is 2, then the reversed BGPs are the expected ones.

4.2.2 FETA algorithms

We propose FETA as a system of heuristics to implement the reverse function f. FETA analyzes a sequence of subqueries to obtain BGPs given a user-defined *gap of time* which

is the time interval, between two subqueries in the log, necessary to say that it is possible they are part of the same federated query.

Algorithm 1: FETA's global algorithm

input : $\mathcal{Q}, \mathcal{A}, gap$
output : $\mathcal{G}$

1 $\mathbb{G} \leftarrow$ GraphConstruction($\mathcal{Q}, \mathcal{A}, gap$)
2 $\mathcal{G} \leftarrow$ NestedLoopDetection($\mathbb{G}, gap$)
3 $\mathcal{G} \leftarrow$ GraphRefinement($\mathcal{G}, gap$)

FETA finds $\mathcal{G}$, the set of BGPs processed by Φ, given: (i) a federation Φ of endpoints sharing their logs, (ii) a federated log $E(FQ_1 \parallel ... \parallel FQ_n)$, containing traces of received queries $\mathcal{Q}$ and returned answers $\mathcal{A}$, and (iii) a gap of time.

Before constructing $\mathcal{G}$, FETA uses a graph $\mathbb{G} = \{g\}$ composed of a set of graphs where $g = \langle V, E, IPaddress \rangle$ is an undirected connected graph, where:

- $V = \{q_i | q_i \in \mathcal{Q}\}$ is a set of distinct nodes, such that q_i is an annotated query like $q_i = \langle q_i, \{timestamp\}, \{endpoint\} \rangle$; q_i may contain a set of endpoints and a set of timestamps for one query, this is due to query aggregation/merging explained next.
- E is a set of distinct pairs (q_i, q_j), annotated with w, the number of common variables of q_i and q_j.
- $IPaddress$ is the client query engine.

FETA works in three phases as shown in Algorithm 1. First phase executes two main functions, $LogPreparation(\mathcal{Q}, \mathcal{A})$ and $CommonJoinCondition(q, \mathbb{G}, gap)$.

LogPreparation prepares and cleans the input log. ASK queries are suppressed and identical subqueries or subqueries differing only in their offset values are aggregated in one single query. Timestamp of such aggregated query becomes an interval. Same queries are sent twice to the same endpoint to be sure obtaining an answer, and to different endpoints to have complete answers. Similar queries with different offsets are sent to avoid reaching the endpoint's limit response.

Common join condition is a heuristic that incrementally constructs $\mathbb{G}$, by joining queries q depending on the given gap and having common projected variables or triple patterns with common IRI or literal on their subjects or objects. In general, subqueries are joined on their common projected variables. However, we consider also IRIs and literals, even if they can produce some false positives.

In our example of Figure 4.2, with a gap of 2, three graphs are constructed: $\mathbb{G} = \{\{sq1\}, \{sq2\}, \{(sq3, sq4)\}\}$.[6]

Nested-loop detection In the second phase, the NestedLoopDetection shown in Algorithm 2, analyzes existing graphs g to detect nodes being part of nested-loops. This step will significantly reduce the size of each g because nested-loops can be executed with hundreds of subqueries. The goal is to identify subqueries participating in a nested-loop

[6]To simplify, all annotations to sq_i are omitted.

and obtain two joined triple patterns from n subqueries. During this phase, the initial graph $\mathbb{G}$, where nodes are subqueries, is analyzed to produce $\mathcal{G}$, a graph where nodes are triple patterns.

Algorithm 2: NestedLoopDetection($\mathbb{G}, gap$)

input : $\mathbb{G}, gap$
output : $\mathcal{G}$

1 $CP \leftarrow \emptyset$
2 **forall** $g \in \mathcal{G}$ **do**
3 $CP \leftarrow CP$+MergingSubqueries(g) `// Merges subqueries having particular characteristics to obtain one inner subquery, then extracts and returns candidate triple patterns`
4 $\mathcal{G} \leftarrow$ Preprocessing($\mathbb{G}$) `// From` $\mathbb{G}$ `that is a graph of subqueries, it is obtained a graph of triple patterns` $\mathcal{G}$
5 **forall** $cp \in CP$ **do**
6 Association($cp, \mathcal{G}, \mathcal{A}$) `// Makes` μ^{-1} `to deduce a corresponding triple pattern tp. If tp is an inner tp, it is linked to an existing outer tp and other joins are searched with CommonJoinCondition`

Nested-loops vary in the way variable bindings of the outer dataset are pushed in subqueries sent to the inner dataset, but also in the syntax of these subqueries. If we observe Table 4.1, Lines 3-5, and Table 4.2, Lines 1-3, we can identify two different nested-loops. To detect them, this heuristic has two steps: (i) *merging subqueries*, that aggregates produced subqueries to push bindings of the outer dataset towards the inner dataset, into one big aggregated subquery (that we call inner subquery) and (ii) *Association* that associates the subquery that pushes the outer dataset (that we call outer subquery) towards the inner subquery. This association is made using the following function of inverse mapping.

We assume pairwise disjoint infinite sets $\mathcal{B}$, $\mathcal{L}$, $\mathcal{I}$ (blank nodes, literals, and IRIs respectively). We assume additionally an infinite set $\mathcal{S}$ of variables. A mapping μ is a partial, non surjective and non injective, function that maps variables to RDF terms $\mu : \mathcal{S} \rightarrow \mathcal{BLI}$. See [96] for more explanations.

Definition 13 (Inverse mapping)**.** *We define μ^{-1}, the inverse mapping as a partial, non surjective and non injective, function $\mu^{-1} : \mathcal{BLI} \rightarrow \mathcal{S}$ where $\mu^{-1} := \{(val, s) \mid val \in \mathcal{BLI}, s \in \mathcal{S}\}$ such that $(s, val) \in \mu$. $\mathcal{B}$ is considered for generalization reasons even if blank nodes cannot be used for joins between 2 different datasets.*

This heuristic currently considers the nested-loop with filter option `nlfo` (used by Anapsid 2.7), the nested-loop with bound joins `nlbj`, and nested-loop with exclusive groups `nleg` (used by FedX 3.0).

Following our example of Figure 4.2, the merging step will merge $sq3$ and $sq4$ because they seem to be part of a nested-loop.[7] The association step will make the inverse mapping of subjects of $sq3$ and $sq4$ searching in the space of mappings answered before

[7] We have several heuristics to decide if a set of queries are part of a nested-loop, here $sq3$ and $sq4$ differ only in their subjects and were executed at the same time.

them. If the gap is 2, μ^{-1} of $\{s2, s3\}$ gives z. Then variable found will replace $s2$ and $s3$ in the corresponding triple pattern and each outer subquery (here $sq2$) is linked to the (merged) inner subquery. Finally, according to the gap, other joins with the inner subquery are searched in all g of the same IP address. Thus, this phase produces $\mathcal{G} = \{\{?x\ p1\ ?y\}, \{?z\ p1\ o2\ .\ ?z\ p2\ ?y\}\}$.

Nested-loop detection is the most challenging heuristic of FETA because μ^{-1} may return more than one variable. In our example, if the gap is ≥ 5, μ^{-1} gives z and x. This depends on the execution context, i.e., concurrent federated queries, and the execution analysis, i.e., the considered gap of time. Thus, some times, identifying the variable that appeared in the original query is uncertain.

Graph Refinement The GraphRefinement phase, shown in Algorithm 3, verifies that 1) edges of each $g \in \mathcal{G}$ that were not produced by an exclusive groups or a nested-loop, are on same concepts for their common projected variables and 2) their join has a not null result set. From this verification, symmetric hash joins are identified.

Algorithm 3: GraphRefinement($\mathcal{G}, gap$)

input : $\mathcal{G}, gap$
output : $\mathcal{G}$

1 **forall** $g \in \mathcal{G}$ **do**
2 $\quad$ $\mathcal{G}$ is verified to confirm or discard existing joins that were not produced by a nested loop
3 $\quad$ **forall** *$e \in g$ such that e is a pairwise join that was not produced by a nested loop join* **do**
4 $\quad\quad$ $SameConcept/SameAs(e, \mathcal{A})$
5 $\quad\quad$ $NotNullJoin(e, \mathcal{A})$

This heuristic produces false positives because it infers all possible joins that may be made at the query engine. If a star-shape set of triple patterns exists, all possible combinations of joins will be deduced instead of the subset of joins chosen by the query engine. The consequence for FETA is that it privileges recall to the detriment of precision. In our example, this phase produces no joins.

4.2.3 Experiments

We evaluate FETA by reusing the queries and the setup of FedBench [106][8]. We use the collection of Cross Domain (CD) and Life Science (LS), each one has 7 federated queries. CD queries concern datasets DBpedia[9], NY Times, LinkedMDB, Jamendo, Geonames and SW Dog Food. LS queries use datasets DBpedia, KEGG, Drugbank and CheBi. We setup in total 19 SPARQL endpoints using Virtuoso OpenLink[10] 6.1.7.

[8] To the best of our knowledge, a public set of real federated queries executed over the Linked Data does not exists.

[9] DBpedia is distributed in 12 data subsets (http://fedbench.fluidops.net/resource/Datasets), in our setup, DBpedia Ontology dataset is duplicated in all endpoints, so we install 11 endpoints for DBpedia instead of 12.

[10] http://virtuoso.openlinksw.com/

Query	Anapsid	FedX	Query	Anapsid	FedX
CD1	14	48	LS1	2	2
CD2	4	4	LS2	44	124
CD3	142	162	LS3	872	4706
CD4	4	104	LS4	10	6
CD5	4	48	LS5	792	916
CD6	16	562	LS6	3252	20878
CD7	52	604	LS7	240	970
Total	236	1532	Total	5212	27602

Table 4.3: Number of SELECT subqueries received for queries of Cross Domain (CD) and Life Science (LS) produced by Anapsid and FedX.

We executed federated queries with Anapsid 2.7 and FedX 3.0. We configured Anapsid to use Star Shape Grouping Multi-Endpoints (SSGM) heuristic and we disabled the cache for FedX. We capture http requests and answers from endpoints with justniffer 0.5.12[11]. FETA is implemented in Java 1.7 and is available at https://github.com/coumbaya/feta.

Table 4.3 presents the number of subqueries received by endpoints, produced by FedX and Anapsid. The number of subqueries can be up to 20878 (for FedX, query LS6) and there is a significant difference between FedX and Anapsid (for LS6, Anapsid produced 3252 subqueries, 6 times less than FedX).

Execution time, of FETA analysis, of almost all CD queries was less that one second. Longest time was for LS6 (from FedX traces): 4 minutes and 31 seconds. You can see details in https://github.com/coumbaya/feta/blob/master/experiments_with_fedbench.md#executiontime.

The goals of the experiments are twofold: (i) to evaluate the precision and recall of FETA with federated queries executed in isolation and (ii) to evaluate the precision and recall of FETA with federated queries executed concurrently under a worst case scenario: when BGPs of different federated queries cannot be distiguished because they come from the same IP address.

FETA deductions of federated queries in isolation We executed CD and LS collections of queries in isolation, using Anapsid and FedX. We captured 28 sequences of subqueries, each one was used as input for FETA. Figures 4.3 to 4.6 present precision and recall of FETA deductions in terms of triple patterns and joins by federated query and by federated query engine.

In average, we obtained 94,64% of precision and 94,64% of recall of triple patterns deduction. We obtained 79,40% of precision and 87,80% of recall for joins detected by FETA.

From Anapsid traces, the sets of triple patterns, i.e., BGPs, deduced correspond to CD and LS queries except for Union queries, i.e., CD1, LS1 and LS2. For CD1, presented in Figure 4.7a, FETA gives one BGP instead of two because of the joinable common variables and the common IRI used in their BGPs. This query has two BGPs but a join is possible between them. As the Union of each query is made locally at the query engine, FETA deduces a symmetric hash join: $\mathsf{FETA}(E_{Anapsid}(CD1)) = \{(tp_2.tp_3)^{@NYT}.tp_1^{@DBpedia}\}$. The deduction was similar for LS2. For LS1, presented in Figure 4.7b, FETA deduces only the

[11] http://justniffer.sourceforge.net/

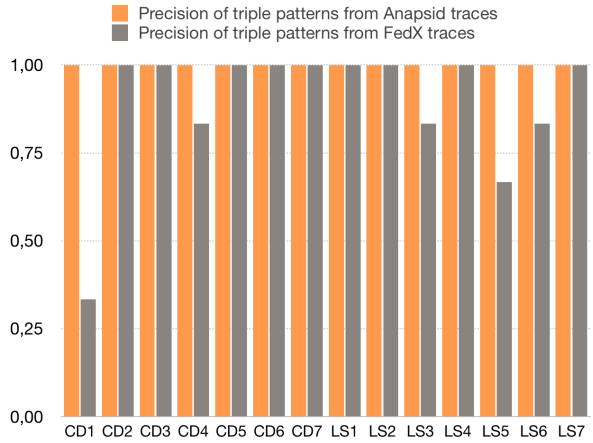


Figure 4.3: Precision of triple patterns.

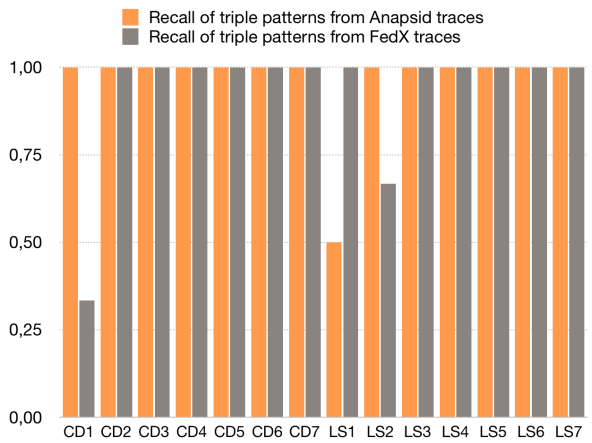


Figure 4.4: Recall of triple patterns.

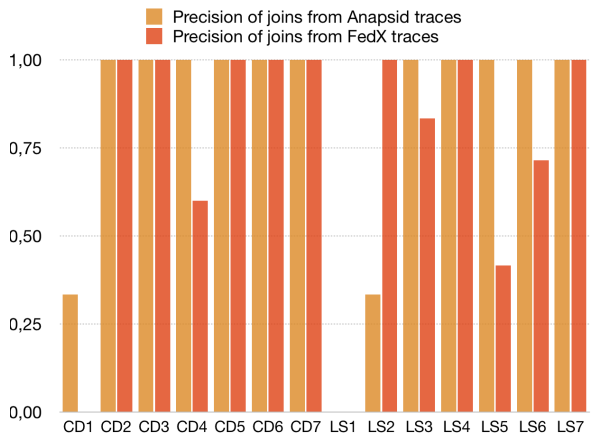


Figure 4.5: Precision of joins.

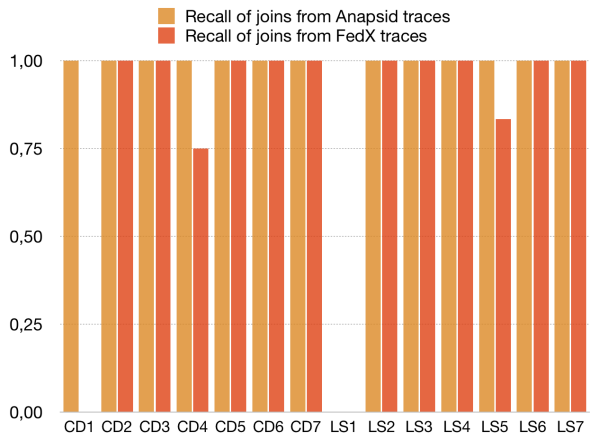


Figure 4.6: Recall of joins.

SELECT ?predicate ?object **WHERE** {{
dbpedia:Barack_Obama ?predicate ?object (tp_1)
} **UNION** {
?subject owl:sameAs dbpedia:Barack_Obama. (tp_2)
?subject ?predicate ?object }} (tp_3)

(a) CD1

SELECT ?drug ?melt **WHERE** {{
?drug *drugbank* : *meltingPoint* ?melt (tp_1)
} **UNION** {
?drug dbpedia-owl-drug:meltingPoint ?melt}} (tp_2)

(b) LS1

Figure 4.7: Two Union queries.

first BGP because Anapsid does not send a subquery for the second BGP of the Union ($tp2$). From its source selection process, Anapsid knows there is no endpoint that can evaluate $tp2$ and only $tp1$ is send to Drugbank.

From FedX traces, deduced BGPs correspond for CD2, CD3, CD5, CD6, CD7, LS4 and LS7. For LS1 FETA finds one BGP instead of two (but unlike Anapsid, all triple patterns are well deduced). All other problems of deduction come from the nested-loop detection of FETA. For CD1 and LS2 FETA fails to find some triple patterns. We illustrate what happen on CD1. Instead of finding the object of $tp2$ that is an IRI, it finds the variable ?*object*. The reason is that this IRI is contained in the mapping of $tp3$ that is used in a nested-loop with $tp1$. Concerning CD4, LS3, LS5, and LS6, FETA finds two possible variables for a component of a triple pattern (a subject or an object). That is because, in the inverse

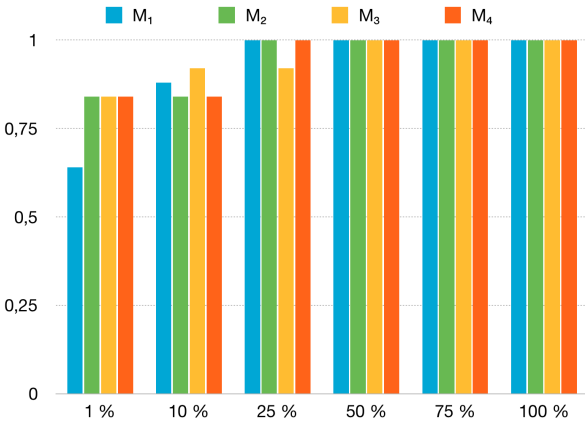


Figure 4.8: Recall of joins from Anapsid traces for MX by gap.

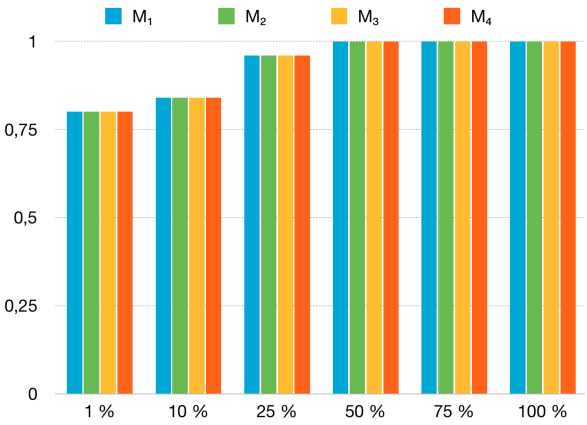


Figure 4.9: Recall of joins from FedX traces for MX by gap.

mapping process, the set of mappings of two different hidden variables are equal, one is contained into another, or there is an intersection between them. We illustrate this case with CD4 (see Figure 4.12). FETA finds that two variables may correspond to the subject of an inner join query, $?y$ and $?actor$. That is because the set of mappings of $?y$ corresponds to a subset of the mappings of $?actor$. As FETA cannot decide which variable is the good one, it produces two triple patterns, the good one with $?y$, and $tp5'$ with $?actor$[12]: $\mathsf{FETA}(E_{FedX}(CD4)) = \{(tp_1.tp_2)^{@LMDB}.(tp_3.tp_4)^{@Geonames,etc.}.tp_5^{@NYT}.|tp_5'^{@NYT}\}$.

In general, because the nested-loop detection of FETA, supplementary triple patterns will be deduced from FedX traces thus making precision decreases. But as right triple patterns are in general well deduced, recall is good.

Thus, FETA succeed in deducing 11 out of 14 exact BGPs from Anapsid traces, and 7 out of 14 from FedX traces. Globally FETA finds 18/28 exact BGPs, i.e., 64%. If we include Union queries where all triple patterns are deduced, FETA finds (18+3)/28 queries, i.e., 75% of FedBench queries.

Does FETA results resist to concurrency? To analyze FETA behavior with concurrent federated queries, we implemented a tool to shuffle logs of queries, produced in isolation, according to different parameters. So, we are able to produce different significative representations of $E(FQ_1 \parallel ... \parallel FQ_n)$ of our queries. Produced traces vary in (i) the order of queries, (ii) the number of subqueries, of the same federated query, appearing continuously (blocks of 1 to 16 subqueries), and (iii) the delay between each subquery (from 1 to 16 units of time).

An important parameter of FETA is the maximum gap of time between two subqueries to consider them as potentially part of the same federated query. In our experiments, gap varies from 1% to 100% of the total time of each mix. We measure precision and recall of deductions made by FETA, making vary the gap, from traces of federated queries in isolation against our mixes of traces of concurrent queries.

If FETA can distinguish triple patterns of concurrent and different federated queries, precision and recall by join are perfect provided that the gap of time is big enough. We analyzed a set of chosen federated queries having distinguishable triple patterns that we

[12] $tp3.tp4$ were sent to: Geonames, NYT, Jamendo, SWDF, LMDB, DBpediaNYT, DBpediaLGD.

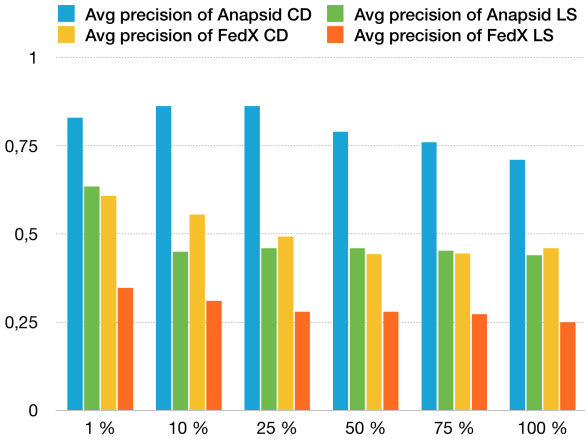


Figure 4.10: Average of precision of joins of the four mixes by gap.

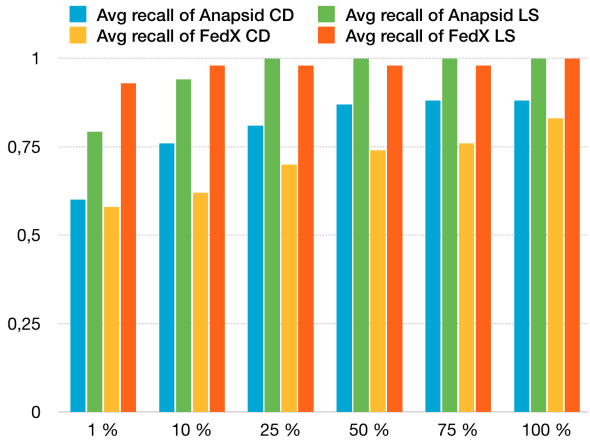


Figure 4.11: Average of recall of joins of the four mixes by gap.

named MX: CD3, CD4, CD5, CD6, LS2 and LS3. We produced 4 different mixes of these queries ($M_1, ..., M_4$) that were analyzed by FETA under 6 different gaps (1%, 10%, etc.) producing 6 groups of deductions. We obtained 100% of precision of joins from traces of Anapsid and FedX since the smallest gap. Concerning recall, Figures 4.8 and 4.9 show recall of joins from Anapsid and FedX traces respectively. We get 100% of recall with a gap of 50% from traces of both query engines.

If triple patterns of concurrent queries are the same or syntaxically similar it is hard for FETA to obtain good precision and recall. We produced four different and concurrent mixes by family of queries of FedBench (4 for CD and 4 for LS). Each set of mixes were analyzed by query engine and by gap.

Figure 4.10 shows the average of precision for each set of mixes, i.e., each bar concerns 4 mixes. From this figure, it is evident to see that for FETA it is easier to analyze traces from Anapsid than from FedX and that CD queries are more distiguishable than LS ones. That is because triple patterns of LS queries vary less than those of CD queries, thus it is less evident to separate LS queries from their mixed traces. Furthermore, the bigger the gap the smaller the precision. That is because more false joins are detected thus reducing precision.

Figure 4.11 shows the average of recall. In general, recall of LS is bigger than recall of CD. That is because LS queries generate lots of symmetric hash joins covering the good ones. Unlike precision, the bigger the gap, the bigger the recall. That is because more joins are detected thus the possibility of finding the good ones is bigger. Results detailed by mix are available at `https://github.com/coumbaya/feta/blob/master/experiments_with_fedbench.md`

4.3 SWEEP, BGP deduction from a streaming log of TPF servers

The Triple Pattern Fragments (TPF) interface demonstrates how it is possible to publish Linked Data at low-cost while preserving data availability [114]. But, data providers hosting TPF servers are not able to analyze the SPARQL queries they execute because

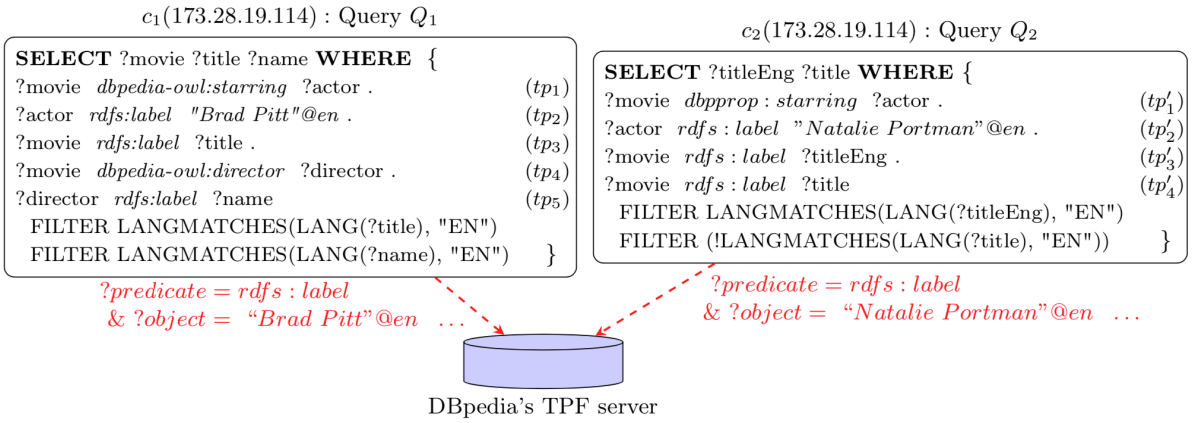


Figure 4.12: Concurrent execution of queries Q_1 and Q_2.

they only receive and evaluate subqueries with one triple pattern. Understanding the executed SPARQL queries is important for data providers for prefetching, benchmarking, auditing, *etc.* In this section I introduce SWEEP, a streaming web service that deduces Basic Graph Patterns (BGPs) of SPARQL queries from a TPF server log. I show that SWEEP is capable of extracting BGPs of SPARQL queries evaluated by a DBpedia's TPF server.

4.3.1 Motivation

Understanding the executed SPARQL queries is fundamental for data providers. Mining logs of SPARQL endpoints allows to detect recurrent patterns in queries for prefetching [77], benchmarking [87], auditing [89], *etc.* It provides the type of queries issued, the complexity and the used resources [82, 97]. It allows also to distinguish between man or machine made queries [100, 103]. Such analysis cannot be done on logs of TPF servers because they only contain information about single triple patterns. A BGP of a SPARQL query, that is a set of conjunctive graph patterns, is scattered over the log.

[113] reported statistics from the logs of the DBpedia's TPF server. However, statistics only concern single triple pattern queries and not BGPs. In the previous section, I proposed an algorithm to extract BGPs of *federated SPARQL queries* from logs of a *federation of SPARQL endpoints*. Here, I address a similar scientific problem but in the context of a single TPF server.

SWEEP is a streaming web service that is able to extract BGPs from logs of TPF servers in real-time. From the stream of single triple pattern queries of a TPF server, SWEEP is capable of extracting BGPs. This allows data providers running TPF servers to better know how their data are used.

In Figure 4.12, two clients, c_1 and c_2, execute concurrently queries Q_1 and Q_2 over the DBpedia's TPF server. Q_1 asks for movies starring Brad Pitt and Q_2 for movies starring Natalie Portman.[13] Both queries have one BGP composed of several triple patterns (tp_n).

The TPF client decomposes the SPARQL queries into a sequence of triple pattern queries partially presented in Table 4.4. Odd-numbered lines represent received triple pattern queries and even-numbered ones represent sent triples after evaluation on the RDF

[13]These queries come from http://client.linkeddatafragments.org/.

	IP	Time	Asked triple pattern/TPF
1	172...	11:24:19	?predicate=rdfs:label & ?object="Brad Pitt"@en
2	172...	11:24:23	**dbpedia:Brad_Pitt** rdfs:label "Brad Pitt"@en ,
3	172...	11:24:24	?predicate=dbpedia-owl:starring & ?object=**dbpedia:Brad_Pitt**
4	172...	11:24:27	**dbpedia:A_River_Runs_Through_It_(film)** dbpedia-owl:starring dbpedia:Brad_Pitt dbpedia:Troy_(film) dbpedia-owl:starring dbpedia:Brad_Pitt ...
5	172...	11:24:28	?subject=**dbpedia:A_River_Runs_Through_It_(film)** &?predicate=rdfs:label

Table 4.4: Excerpt of a DBpedia's TPF server log for query Q_1.

graph. Lines 1 and 3, correspond to triple pattern queries for tp_2 and tp_1 of Q_1.[14] We can observe that the object in Line 3, comes from a mapping seen in Line 2. This *injection* of a mapping obtained from a previous triple pattern query, is clearly a *bind join* from tp_2 towards tp_1.

As the TPF server only sees triple pattern queries, the original queries are unknown to the data provider. Thus, our research question is: *Can we extract BGPs from a TPF server log?*

The main challenge is to distinguish *similar queries*, that is queries whose triple patterns are the same for the TPF server as tp_1 vs tp'_1. In our example, we aim to extract two BGPs from the TPF server log, one corresponding to Q_1, BGP[1]= $\{tp_1.tp_2.tp_3.tp_4.tp_5\}$ and another corresponding to Q_2, BGP[2]= $\{tp'_1.tp'_2.tp'_3.tp'_4\}$.

4.3.2 SWEEP in a nutshell

SWEEP uses a TPF server log, as the one of Table 4.4, composed of an unlimited ordered sequence of execution traces organized by IP-address. It considers a fixed-size window sliding over the TPF server log. Window size can depend on the memory available for the streamed log or on the average of known values used as timeout by TPF clients.

We consider a set G of deduced BPGs. Each time a triple pattern query (tpq_i) arrives, SWEEP creates a new $BPG_j \in G$ or updates an existing one.

Suppose G is empty and SWEEP receives tpq_1 =`{?s p2 toto}` where `?s` produces 2 mappings: $\{c1, c2\}$. As G is empty, SWEEP creates BGP_1 containing tpq_1 with the current time as timestamp, $BGP_1.ts = time()$.

Then, if tpq_2 =`{c1 p1 ?o}` arrives, as $c1$ appears in mappings of a $BGP_j \in G$, SWEEP detects a bind join. This implies updating BGP_1 with the join `{?s p2 toto . ?s p1 ?o}`. If tpq_3 = `{c2 p1 ?o}` arrives, as it is already represented in BGP_1, nothing is done.

If BGP_1 is out the window, *i.e.*, $time() - BGP_1.ts > window$, then it must no longer be updated; it is delivered and removed from the stream.

We run SWEEP with queries proposed by the TPF web client (`http://client.linkeddatafragments.org/`). From 21 queries executed, we obtained 100% of precision and 87% of recall of deduced BGPs when compared to the BGPs of corresponding original queries. SWEEP succeeds in this case because these queries are note very *similar*. Different precision and recall would be produced with a more challenging set of queries.

[14] TPF clients always rename variables as *"subject"* or *"object"*, regardless of how they are named in the original query.

INFORMATION

Gap (hh:mm:ss)	Evaluated Queries	BGP	TPQ	Avg Precision	Avg Recall
0:01:30	3 / 3	9	270	1.000	1.000

DEDUCED BGPS

(20 more recents)

	ip	time	bgp	Original query	Precision	Recall
9	127.0.0.1	2017-07-27 18:33:50.460040	?jo0 <http://www.w3.org/2000/01/rdf-schema#label> "Belgium"@en . ?jo1 <http://dbpedia.org/ontology /locationCountry> ?jo0 . ?se18777_7484 <http://dbpedia.org /ontology/developer> ?jo1 .	qsim-WS-127.0.0.1-84_1 PREFIX dbpedia-owl: <http://dbpedia.org/ontology/> PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#> SELECT ?software ?company WHERE { ?software dbpedia-owl:developer ?company. ?company dbpedia-owl:locationCountry [rdfs:label "Belgium"@en]. }	1.000	1.000
8	127.0.0.1	2017-07-27 18:33:35.535898	?js0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dbpedia.org /ontology/Book> . ?js0 <http://dbpedia.org/ontology /author> ?oe18782_7064 .	qsim-WS-127.0.0.1-83_1 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> PREFIX dbpedia-owl: <http://dbpedia.org /ontology/> SELECT DISTINCT ?book ?author WHERE { ?book rdf:type dbpedia-owl:Book; dbpedia-owl:author ?author. } LIMIT 100	1.000	1.000
7	127.0.0.1	2017-07-27 18:33:15.094616	?js0 <http://dbpedia.org/property /cityServed> <http://dbpedia.org /resource/Italy> . ?js0 <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://dbpedia.org /ontology/Airport> .	qsim-WS-127.0.0.1-82_1 PREFIX dbpedia-owl: <http://dbpedia.org/ontology/> PREFIX dbpprop: <http://dbpedia.org/property/> SELECT DISTINCT ?entity WHERE { ?entity a dbpedia-owl:Airport; dbpprop:cityServed dbpedia:Italy. }	1.000	1.000
6	127.0.0.1	2017-07-27 18:33:51.133406	?se18779_7052 <http://dbpedia.org /ontology/developer> ?oe18779_7052 .	No query assigned		

Figure 4.13: SWEEP dashboard.

4.3.3 Implementation and demonstration

Figure 4.13 presents the dashboard of SWEEP available at http://sweep.priloo.univ-nantes.fr. It shows the most recent deduced BGPs and original client queries when they are available. Our TPF client, http://tpf-client-sweep.priloo.univ-nantes.fr, sends the original client query to SWEEP to be able to calculate precision and recall.

If you want to test SWEEP with another TPF client, you must specify the address of the DBpedia's TPF server we have setup: http://tpf-server-sweep.priloo.univ-nantes.fr. In this case, SWEEP will deduce BGPs but will not be able to calculate precision and recall.

We used, the versions of JavaScript for Node.js of the TPF server and client. The source code is available at https://github.com/edesmontils/SWEEP.

4.4 Conclusion

Federated query tracking allows data providers to know how their datasets are used. In this chapter I proposed FETA, a federated query tracking approach that reverses federated Basic Graph Patterns (BGP) from a shared log maintained by data providers. FETA links and unlinks variables present in different subqueries of the federated log by applying a set of heuristics to decrypt behaviour of physical join operators.

Even in a worst case scenario, FETA extracts BGPs that contain original BGPs of federated queries executed with Anapsid and FedX. Extracted BGPs, annotated with

endpoints, give valuable information to data providers about which triples are joined, when and by whom.

We think FETA opens several interesting perspectives. First, heuristics can be improved in many ways by better using semantics of predicates and answers. Second, we can improve FETA to make it agnostic to the federated query engine.

Third, FETA can be used to generate a transactional log of BGPs from a temporal log of subqueries. Analyzing frequency of BPGs in a transactional log should allow to discriminate false positive deductions of FETA.

SWEEP demonstrates how it is possible to deduce the BGPs executed by a TPF server. This allows data providers to have a better understanding of the usage of their data.

With SWEEP it would be possible to detect whether clients are executing federated queries over multiple datasets hosted by one TPF server. And if multiple data providers agree on streaming their logs to a shared SWEEP service, they would be able to detect federated queries executed over multiple TPF servers.

Chapter 5

Combination and classification of privacy policies

Contents

5.1 Introduction **75**
5.2 PrODUCE policies composition based on data usage context **76**
5.2.1 The PriLoo ontology 77
5.2.2 Motivating example: geriatric center use case 78
5.2.3 Composition process 79
5.3 CaLi: a lattice-based model for license classifications **86**
5.3.1 Formal model description 87
5.3.2 A CaLi ordering for Creative Commons 91
5.3.3 Implementation of CaLi orderings 93
5.4 Conclusion **95**

5.1 Introduction

Web applications facilitate combining resources (linked data, web services, source code, documents, *etc.*) to create new ones. To facilitate reuse, resource producers should systematically associate licenses with resources before sharing or publishing them [108]. Licenses specify precisely the conditions of reuse of resources, *i.e.*, what actions are *permitted*, *obliged* and *prohibited* when using the resource. Very well known examples of licenses are the family of Creative Commons licences[1]. Privacy policies, more specifically, describe licenses but also the context of resource usage like involved entities (persons, institutions, *etc.*), dates, particular data, *etc.*

For a resource producer, choosing the appropriate license for a combined resource or choosing the appropriate licensed resources for a combination is a difficult process. It involves choosing a license compliant with all the licenses of combined resources as well as

[1] https://creativecommons.org/licenses/

analysing the reusability of the resulting resource through the compatibility of its license. The risk is either, to choose a license too restrictive making the resource difficult to reuse, or to choose a not enough restrictive license that will not sufficiently protect the resource.

Relations of compatibility, compliance and restrictiveness on licenses could be very useful in a wide range of applications. We consider simplified definitions of compliance and compatibility inspired by works like [35, 39, 65, 116]: *a license l_j is compliant with a license l_i if a resource licensed under l_i can be licensed under l_j without violating l_i.* If a license l_j is compliant with l_i then we consider that l_i is compatible with l_j and that resources licensed under l_i are reusable with resources licensed under l_j. Usually but not always, when l_i is less restrictive than l_j then l_i is compatible with l_j.

In my work, I am interested in facilitating users to choose the rigth license to their resources. On the one hand, I address the problem of policies' composition, that is, from a set of policies how to compose a policy compatible with all of them. On the other hand, I am interested in generalizing the positionning of licenses in terms of compatibility, in particular, how to automatically position a license over a set of licenses in terms of compatibility and compliance?

In this chapter, I present two contributions. Section 5.2 presents a mechanism, based on semantic web technologies, to compose privacy policies. The originality of this approach is that the composition rules are based on the data usage context and deduce implicit terms. Section 5.3 presentas a model for license orderings that uses restrictiveness relations and constraints among licenses to define compatibility and compliance.

Contributions of this chapter were published in [83, 85, 86, 110]

5.2 PrODUCE policies composition based on data usage context

The semantic web allows to express data in a way that facilitates data sharing and data analysis. On the one hand, data owners can share their data through endpoints which process queries. Privacy policies are often attached to these data. These policies describe how to use data, what is permitted, obliged or prohibited. Nevertheless, everything is not expressed and implicit aspects should be considered about data usage taking into account contextual aspects. On the other hand, data consumers access endpoints to process data using a query engine. This latter can be a federated query engine able to process queries, orchestrating simultaneous access to multiple endpoints.

The issue we deal with here is, how to compute the usage policy of combined data, result of a federated query. Challenging questions are (i) how to define a usage policy by composition of multiple usage policies? (ii) how to take into account usage context aspects like usage location, usage purposes but also predefined stances of users (optimistic, pessimistic)? (iii) how to compose privacy policies that are not defined with the same set of policy terms or how to manage a term created specifically for a policy or for a context?

In this section, an approach for composition of usage policies, based on semantic web technologies is proposed. Besides defining usage policies, this solution takes into account implicit or general aspects of the data usage context during policies composition.

5.2.1 The PriLoo ontology

The ontology Privacy Lookout[2] (PriLoo) is used to represent policies. An abstraction of the PriLoo ontology is shown in Figure 5.1. It supports traditional models, such as *permissions*, *prohibitions* and *obligations*, represented as properties and organized between a Policy of Usage Context (PUC) and a License. A PUC has a License which can obligate or prohibit *LegalTerms* (*i.e.*, *by*, *sa*, *history*, *notice*, *etc.*). Licenses can also permit or prohibit *Operations* (*read*, *write*, *distribute*, *etc.*).

PUC describes the usage policy under different contextual aspects. It describes (i) *implicitProperties* in terms of *ImplicitStatus*, two values are allowed, *all-but-prohibited*, to prohibit all implicit terms and *all-but-permits-or-obliges*, to permit or to oblige implicit terms; (ii) *Purposes* (*i.e.*, *commercial*, *medical*, *tracking*, *scientific*, *etc.*) that are considered in the usage context because they are business activities; (iii) other properties like the grantee, the grantor, concerned resources, the valid period of time, the usage locality, *etc.*

LegalTerms, *Operations* and *Purposes* are *terms* that can be structured according to a hierarchical tree using the inheritance relationship. For instance, *LegalTerm* "moral rights preserve" inherits of "rights preserve", *consultation* purpose inherits of *medical* purpose).[3]

Several standard licenses have been defined in PriLoo like CC-By or Beerware.[4] In order to simplify licenses, PriLoo allows to define a license as part of a family of licenses. Thus, licenses which have common descriptions are grouped into families, for example *CreativeCommons* or *PublicDomain*. These licenses can be included in PUCs.

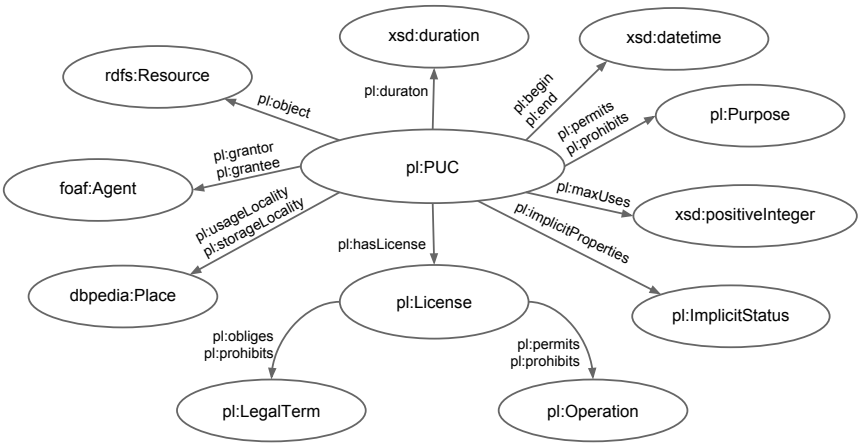


Figure 5.1: Abstraction of the PriLoo ontology.

Figure 5.2 shows two examples of privacy policies written in the RDF/N3 syntax. *Policy 1* is defined by a resident of a care institution, to protect access to his/her personal information such as temperature or blood pressure, contained in the file *Resident1PersonalData.n3*. In addition, the PUC of this policy permits access for *scientific* and *medical* purposes but the *tracking* purpose is prohibited. The grantee is a geriatrician. The licence of this

[2] http://www.desmontils.net/PriLoo/current/pl-ontology.n3.

[3] See http://www.desmontils.net/PriLoo/current/pl-legal-terms.n3 to obtain the legal terms, operations and purposes defined in PriLoo.

[4] See http://www.desmontils.net/PriLoo/current/pl-licenses.n3 for the list of standard licenses expressed with PriLoo.

policy allows the *read* operation. *Policy 2* is defined by Mary Thomson. PUC states that data about daily activities, contained in the file *Digitalresources.n3*, can be accessed for *scientific* purposes by a research center (specified in *pl:grantee*). The license of this policy is CC BY that belongs to the family *CreativeCommons* and the *write* operation is permitted.

```
1 :License1 a pl:License ;
2 pl:permits operation:read .

3 :PUC_elder1 a pl:PUC ;
4 pl:permits purpose:scientific, purpose:medical ;
5 pl:prohibits purpose:tracking ;
6 pl:object <Resident1PersonalData.n3> ;
7 pl:hasLicense :License1 ;
8 pl:duration "P0Y0M2D"^ ^ xsd:duration ;
9 pl:maxUses "3"^ ^ xsd:integer ;
10 pl:grantee
<http://www.clinicasantaclarita.com/
11 Dr_Clemente_Humberto_Zuniga_Gil.html>,
<http://serenaseniorcare.com/>,
<http://www.cicese.edu.mx/> ;
12 pl:grantor <Resident1.n3> ;
13 pl:usageLocality
<http://dbpedia.org/resource/Mexico>,
<http://dbpedia.org/resource/USA> ;
14 pl:storageLocality
<http://dbpedia.org/resource/Mexico> ;
15 pl:global-preference pl:pessimistic .
```

a) Policy 1

```
1 :License2 a pl:License ;
2 pl:memberOfTheFamily lic:CreativeCommons ;
3 pl:obliges legalTerm:by ;
4 pl:permits operation:write .

5 :PUC_researchCenter1 a pl:PUC ;
6 pl:object <Digitalresources.n3> ;
7 pl:hasLicense :License2 ;
8 pl:permits purpose:scientific ;
9 pl:prohibits purpose:sales, purpose:commercial
;
10 pl:duration "P0Y0M2D"^^xsd:duration ;
11 pl:grantee <http://www.cicese.edu.mx/> ;
12 pl:grantor <MaryThomson.n3> ;
13 pl:usageLocality
<http://dbpedia.org/resource/
Mexico>;
14 pl:storageLocality
<http://dbpedia.org/resource/
USA> .
```

b) Policy 2

Figure 5.2: Two examples of privacy policies.

Other contextual aspects are defined like location of storage (*e.g.*, "Mexico" in line 14 of Figure 5.2a), usage locality (*e.g.*, "Mexico" in line 13 of Figure 5.2b), the period of time of usage, number of permitted usages (*e.g.*, 3 uses in line 9 of Figure 5.2a).

5.2.2 Motivating example: geriatric center use case

The scenarios considered concern daily activities of a geriatric center [109] where many older adults are living together. Produced personal data are stored in a distributed system where each older adult has his/her own storage system and personal policies available through an endpoint. We consider a single federated query engine as a web service available to physicians, caregivers, scientist, *etc.* Our module PrODUCE (Privacy Policies cOmposition with Data Usage ContExt) performs a context-aware composition process using ontology-based rules (see Figure 5.3).

Scenario 1.

Every two weeks a physician visits the geriatric center to check residents. She searches for the relevant data of each consulted resident, so she uses the information system to get the data. The relevant information is related with vitals signs (temperature, blood pressure, pulse, weight, etc.*), meals, medicaments, but also, anomalies or comments from caregivers. In this occasion a physician requires data about blood pressure of all older adults who have hypertension.*

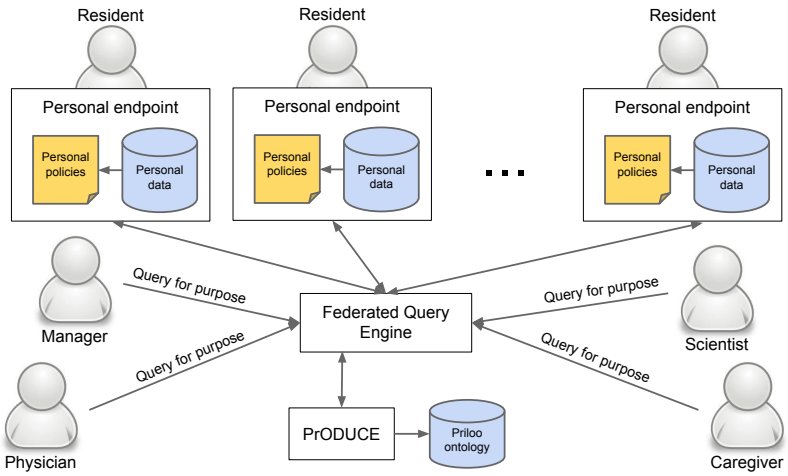


Figure 5.3: Federated query process in a geriatric center.

In this scenario three policies from different residents are considered: *Policy 1* (Figure 5.2), *Policy 3* and *Policy 4* (Figure 5.4). The physician needs to obtain the older adults having blood pressure > 120, which is considered as an indicator of hypertension. The purpose of the physician's query is *medical* which is included in the query.

Scenario 2.

The geriatric center collaborates with other institutions for scientific research purposes. Scientists investigate about specific topics related with the caring process. This time a scientist performs an evaluation of a group of elders taking a particular drug. For this, she queries regularly the blood pressure of the group and collects related data of every older adult of the group.

For this scenario, three policies from different residents are also considered: *License1*, *License4* and *License5*. Now, the purpose of this query is *scientific*.

In both scenarios, users want to query data about older adults. Each resident has his/her own personal policy and the users their specific query's purposes. So the need to merge every aspect of concerned policies emerged. The composition process is presented in next section.

5.2.3 Composition process

The proposed composition process, generates a policy from a set of policies, see Figure 5.5. Firstly, input policies are extended with terms used by the PUC and the license. Then implicit terms are added according to context rules or explicit default terms. Secondly, basic composition rules are applied. Finally, inconsistencies are identified and solved. When the composition is not possible a *FALSE* answer is returned. The rationale of every stage is presented next, and illustrated, step by step, with scenarios from previous section.

```
1 :License3 a pl:License ;
2 pl:permits operation:read, operation:distribute .

3 :PUC_elder2 a pl:PUC ;
4 pl:permits purpose:medical ;
5 pl:prohibits purpose:commercial, purpose:scientific ;
6 pl:object <Resident2PersonalData.n3> ;
7 pl:hasLicense :License3 ;
8 pl:duration "P0Y0M2D"^ ^ xsd:duration ;
9 pl:maxUses "3"^ ^ xsd:integer ;
10 pl:global-preference pl:pessimistic ;
11 pl:grantee <http://serenaseniorcare.com/> ;
12 pl:grantor <Resident2.n3> ;
13 pl:usageLocality <http://dbpedia.org/resource/Mexico> ;
14 pl:storageLocality <http://dbpedia.org/resource/Mexico> .
```

a) Policy 3

```
1 :License4 a pl:License ;
2 pl:permits operation:sharing, operation:publishing, operation:distribute, operation:read ;
3 pl:obliges tlegalTerm:by .

4 :PUC_elder3 a pl:PUC ;
5 pl:permits purpose:scientific, purpose:medical, purpose:wellbeing, purpose:consultation, pur-
pose:commercial ;
6 pl:object <Resident3PersonalData.n3> ;
7 pl:hasLicense :License4 ;
8 pl:duration "P0Y0M2D"^ ^ xsd:duration ;
9 pl:maxUses "3"^ ^ xsd:integer ;
10 pl:global-preference pl:optimistic ;
11 pl:grantee <http://www.clinicasantaclarita.com/Dr_Clemente_Humberto_Zuniga_Gil.html>,
<http://serenaseniorcare.com/> ;
12 pl:grantor <Resident3.n3> ;
13 pl:usageLocality <http://dbpedia.org/resource/Mexico>, <http://dbpedia.org/resource/USA>
;
14 pl:storageLocality <http://dbpedia.org/resource/Mexico>,
<http://dbpedia.org/resource/USA> .
```

b) Policy 4

```
1 :License5 a pl:License ;
2 pl:permits term:read, term:distribution .

3 :PUC_elder2 a pl:PUC ;
4 pl:permits term:scientific, term:tracking ;
5 pl:prohibits term:commercial, term:medical ;
6 pl:object <Resident2PersonalData.n3> ;
7 pl:hasLicense :License5 ;
8 pl:duration "P0Y0M2D"^ ^ xsd:duration ;
9 pl:maxUses "3"^ ^ xsd:integer ;
10 pl:global-preference pl:optimistic ;
11 pl:grantee <http://serenaseniorcare.com/> ;
12 pl:grantor <Resident2.n3> ;
13 pl:usageLocality <http://dbpedia.org/resource/Mexico> ;
14 pl:storageLocality <http://dbpedia.org/resource/Mexico> .
```

c) Policy 5

Figure 5.4: Policies for scenarios 1 and 2.

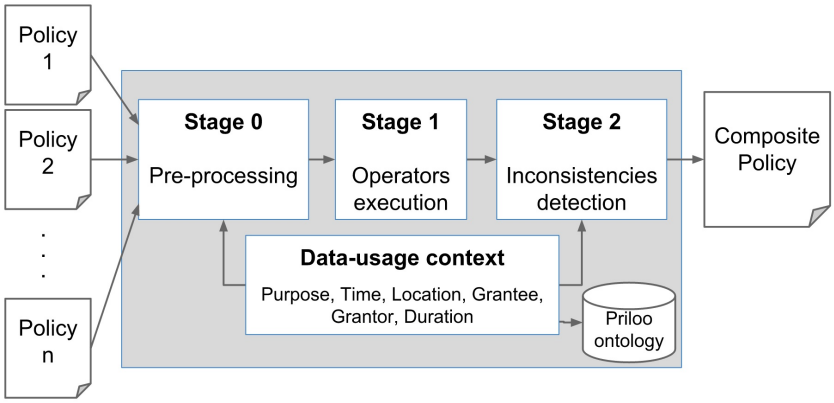


Figure 5.5: Composition process.

Stage 0. Pre-processing. In this stage, the policies are analyzed and, if necessary, all additional terms of the ontology (which are concerned by the policy) are incorporated. This rule-based process, takes into account not only terms existing at the PUC definition but also all terms existing at the composition time (perhaps terms existing in another policy). Consequently, rules that exploit relationships with data-usage context and, sometimes, that define implicit terms management are proposed to complete policies. Table 5.1 presents examples of rules (written in Jena[5]) used in this stage. Three sets of rules were defined:

1. "business rules" depend on licenses (Table 5.1a). For instance, these rules are used to add a purpose to a PUC or to add a legal term, associated to a given purpose, to a license.

2. "propagation rules" take into account inheritance of terms according to a property (Table 5.1b). For instance, when a term inherits from another term and the last one is permitted, then the first one is permitted too (unless the PUC prohibits it).

3. "implicit management rules" manage implicit terms (Table 5.1c). If a term is not specified in a PUC, the propagation rules (explained before) do not conclude and the property to manage them (*pl:implicitProperties*) does not exist, then, implicit management rules deduce, from the context (purposes, licenses, *etc.*), if unspecified terms are permitted, obliged or prohibited.

Pre-processing stage for Scenario 1.

The input policies are pre-processed and all of them allow the *medical* purpose which is the main query's purpose of the physician. The expanded policies include terms added based on the *medical* purpose after applying the business rules. For instance, in the extended *Policy 3*, shown in Figure 5.6a, *consultation* and *tracking* purposes are added because they inherit from the *medical* purpose. Moreover, due to the *medical* purpose and after applying the implicit management rules (*all-but-prohibited* because in *medical* context all

[5]Java API for RDF in https://jena.apache.org/.

Table 5.1: Ontology-based rules considering data usage context.

	Ontology-based rules	Description
a	[addTermsFromContext: (?up pl:hasLicense ?l), (?up pl:permits term:scientific), noValue(?l pl:PolicyProperty term:constraintDerivative) -> (?l pl:obliges term:constraintDerivative)]	For a context which contains the purpose "scientific", this business rule adds to the licence the obligation of "constraintDerivative"'.
b	[addHierachicalPurposes: (?up pl:hasLicense ?lic), (?up pl:prohibits ?t), (?s term:inherits ?t), noValue(?lic ?p ?s) -> (?up pl:prohibits ?s)]	This propagation rule applies inheritance to a term : when a term is prohibited, all terms more specific are also prohibited unless this term is already used.
c	[addImplicitTerms: (?up pl:permits term:medical), -> noValue(?p pl:implicitProperties ?q) (?up pl:implicitProperties pl:all-but-prohibited)]	In this implicit management rule, when the PUC permits a medical use, all terms which are not used by the license are prohibited.

data is very important and must be treated as confidential), all not explicitly specified terms are added as *prohibits*, as well as prohibited purposes in the PUC.

Pre-processing stage for Scenario 2.

One objective pursued by scientists is to publish their research results. Therefore, as a part of the context the *publishing* purpose inside the *permits* model is included in all the policies containing the *scientific* purpose. These not explicitly specified terms are included in the pre-processing stage, the property *all-but-permit-oblige* is used to perform this action. This property also allows to include all not prohibited terms as obligations inside the *obliges* model shown in *Policy 5* (Figure 5.6b).

Stage 1. Operators execution. In this stage, the terms of all models (*permits, prohibits and obliges*) are analyzed. The operators shown in Table 5.2 are applied depending on the model: AND operator for permissions and OR operator for prohibitions and obligations. AND operator includes terms that appear in all the policies inside the *permits* model. In the *prohibits* and *obliges* models, the OR operator includes all the terms that appear in at least one policy. [6]

Operators execution stage for Scenario 1.

Here, the AND/OR operators are applied to combine the three policies. In this case, only the *permits* (*i.e.*, *read* and *medical*) that appeared in all the policies are added to the composite policy, all *prohibitions* and *obligations* terms are also added to the composite policy if they appear in at least one policy.

Operators execution stage for Scenario 2.

In this case, the *permits* purposes are *scientific* and *distribute*. The latter as a result of the policy expansion in the pre-processing stage. The purpose *medical* is *prohibit* therefore it

[6] Due to lack of space, the resulting policies of the scenarios are not presented.

```
1 :License3 a pl:License ;
2 pl:permits operation:distribute , operation:read ;
3 pl:prohibits term:rightsPreserve , term:by , term:rename , opera-
tion:publishing , term:history , term:origin , operation:unlimitedDisclosure
, term:PublicDomainPreserve , term:waiver , term:using , term:fairDealing
, term:holdLiable , term:lesserCopyLeft , operation:sharing ,
term:limitedCommercial , term:moralRightsPreserve , term:copyrightNotice
, term:derivative , operation:copy , term:warranty , operation:write ,
term:freeSourceCode , term:otherRightsPreserve , term:sa , term:notice ,
term:constraintDerivative .

4 :PUC_elder2 a pl:PUC ;
5 pl:begin "2014-02-03T00:00:00.000+01:00" ;
6 pl:getPurposeFrom :License3 ;
7 pl:global-preference pl:pessimistic ;
8 pl:grantee <http://serenaseniorcare.com/> ;
9 pl:grantor <Resident2.n3> ;
10 pl:hasLicense :License2 ;
11 pl:implicitProperties pl:all-but-permitted-or-obliged , pl:all-but-
prohibited ;
12 pl:object ;
13 pl:permits purpose:consultation , purpose:tracking , purpose:medical ;
14 pl:prohibits purpose:scientific , purpose:care , purpose:sales , pur-
pose:privateUse , purpose:commercial , purpose:gift , purpose:wellbeing ,
purpose:management ;
15 pl:storageLocality <http://dbpedia.org/resource/Mexico> ;
16 pl:usageLocality <http://dbpedia.org/resource/Mexico> ;
17 pl:duration "P0Y0M2D"^ ^ xsd:duration ;
18 pl:maxUses 3 .
```

a) Policy 3 extended, Scenario 1

```
1 :License5 a pl:License ;
2 pl:obliges term:fairDealing , term:constraintDerivative ,
term:waiver , term:otherRightsPreserve , term:copyrightNotice ,
term:warranty , term:history , term:sa , term:notice , term:holdLiable ,
term:lesserCopyLeft , term:by , term:origin , term:PublicDomainPreserve ,
term:moralRightsPreserve , term:limitedCommercial , term:freeSourceCode ,
term:rightsPreserve ;
3 pl:permits operation:read , term:sharing , operation:rename , opera-
tion:distribute , operation:using , operation:unlimitedDisclosure , opera-
tion:derivative , operation:copy , operation:write , operation:publishing .

4 :PUC_elder2 a pl:PUC ;
5 pl:begin "2014-02-03T00:00:00.000+01:00" ;
6 pl:duration "P0Y0M2D"^ ^ xsd:duration ;
7 pl:getPurposeFrom :License5 ;
8 pl:global-preference pl:optimistic ;
9 pl:grantee <http://serenaseniorcare.com/> ;
10 pl:grantor <Resident2.n3> ;
11 pl:hasLicense :License3 ;
12 pl:implicitProperties pl:all-but-permitted-or-obliged ;
13 pl:object ;
14 pl:permits purpose:management , purpose:scientific , purpose:tracking ,
purpose:privateUse , term:care , term:wellbeing ;
15 pl:prohibits purpose:sales , purpose:commercial , purpose:medical ,
purpose:gift , purpose:consultation ;
16 pl:storageLocality <http://dbpedia.org/resource/Mexico> ;
17 pl:usageLocality <http://dbpedia.org/resource/Mexico> ;
18 pl:maxUses 3 .
```

b) Policy 5 extended, Scenario 2

Figure 5.6: Extended policies for both scenarios.

remains as is. The term *by* is *obliges* and all the rest of the terms added in the previous stage remain (*i.e.*, *moralRights*, *origin*, *etc.*).

Model	Operator	Description
Permits	AND	A term is permitted if it appears in all policies.
Prohibits and Obliges	OR	A term is prohibited/obligated if it appears in at least in one policy.

Table 5.2: Operators used for policies composition.

Stage 2. Inconsistencies detection. Generated policy in the previous stage is checked to search for inconsistencies. In this verification, we consider that original terms appearing in the policy have the highest priority, terms added by business rules have a medium priority, and terms added by implicit management and propagation rules have the lowest priority. Taking into account the previous priorities, next steps were applied:

- if one permitted term, with the same priority, is prohibited in at least one policy, then it will not be included in the final policy;
- if two terms are not compatible then we choose one of them based on the requester purpose;
- the term with the highest priority will always be included in the final policy.

Inconsistencies detection stage for Scenario 1.

The final stage eliminates the remaining inconsistencies, *i.e.*, the term *by* and *constraint-Derivative* as *obliges* are the original terms. This means they have the highest priority, then they are eliminated from the *prohibits* model (Figure 5.7a).

Inconsistencies detection stage for Scenario 2.

Some of the terms of the resulted policy, after the previous stage, appeared as *obliges* and *prohibits* what generates inconsistencies. Only the purpose *scientific* and, terms *publishing* and *read* were as *permits*. For the composite policy (Figure 5.7b), the inconsistencies found among terms inside *obliges* and *prohibits* models are suppressed considering the priority of the terms (*i.e.*, *by*, *notice*).

As can be seen, each purpose contributes to each policy by adding different terms. Also, the stance of the owner is considered, as well as the user purpose (*i.e.*, purposes expressed in the query by the scientist - *scientific* purpose - and the physician - *medical* purpose).

It is possible the composition gives an empty policy if the composition process does not succeed. When the composition process ends, PrODUCE sends its results to the query engine. If the process does not succeed, the result is *FALSE*. Otherwise, the produced policy is returned. The query engine then returns FALSE or processes the user query. If the query is possible, the composed policy is attached to the query result and the *pl* properties are used to specify the concerned resource (the query result as a *pl:object*).

```
1 :resultedMedicalPolicy a pl:License ;
2 pl:obligues term:by , term:constraintDerivative ;
3 pl:permits operation:read ;
4 pl:prohibits operation:rename , term:PublicDomainPreserve , term:waiver ,
term:fairDealing , term:otherRightsPreserve , term:holdLiable , operation:using ,
term:copyrightNotice , term:warranty , operation:distribute , operation:derivative , term:sa
, term:rightsPreserve , operation:copy , term:lesserCopyLeft , operation:sharing , term:by
, term:unlimitedDisclosure , term:limitedCommercial , term:history , operation:write ,
operation:publishing , laterm:moralRightsPreserve , term:freeSourceCode , term:origin ,
term:notice .

5 :PUC_scenario1 a pl:PUC ;
6 pl:permits purpose:medical , purpose:consultation ;
7 pl:prohibits purpose:care , purpose:tracking , purpose:management , purpose:sales , pur-
pose:privateUse , purpose:commercial , purpose:gift , purpose:scientific , purpose:wellbeing
.
8 pl:duration "P0Y0M2D"^ ^ xsd:duration ;
9 pl:grantee <http://serenaseniorcare.com/> ;
10 pl:grantor <Residen1.n3>, <Residen2.n3>, <Residen3.n3> ;
11 pl:hasLicense resultedMedicalPolicy ;
12 pl:object <CompositePersonalData.n3> ;
13 pl:storageLocality <http://dbpedia.org/resource/Mexico> ;
14 pl:usageLocality <http://dbpedia.org/resource/Mexico> ;
15 pl:maxUses 3 .
```

a) Scenario 1

```
1 :resultedScientificPolicy a pl:License ;
2 pl:obliges term:moralRightsPreserve , term:by , term:notice , term:lesserCopyLeft
, term:holdLiable , term:fairDealing , term:origin , term:rightsPreserve ,
term:PublicDomainPreserve , term:warranty , term:copyrightNotice , term:waiver , term:sa
, term:constraintDerivative , term:otherRightsPreserve , term:history , term:freeSourceCode
, term:limitedCommercial ;
3 pl:permits operation:publishing , operation:read ;
4 pl:prohibits operation:rename , term:PublicDomainPreserve , term:waiver ,
term:fairDealing , term:otherRightsPreserve , term:holdLiable , operation:using ,
term:copyrightNotice , term:warranty , operation:distribute , operation:derivative , term:sa
, term:rightsPreserve , operation:copy , term:lesserCopyLeft , operation:sharing , term:by
, term:unlimitedDisclosure , term:limitedCommercial , term:history , operation:write ,
term:moralRightsPreserve , term:freeSourceCode , term:origin , term:notice .

5 :PUC_scenario2 a pl:PUC ;
6 pl:permits purpose:scientific ;
7 pl:prohibits purpose:consultation , purpose:care , purpose:tracking , purpose:management
, purpose:sales , purpose:privateUse , purpose:commercial , purpose:gift , purpose:medical ,
purpose:wellbeing .
8 pl:duration "P0Y0M2D"^ ^ xsd:duration ;
9 pl:grantee <http://cicese.edu.mx/> ;
10 pl:grantor <Residen1.n3>, <Residen2.n3>, <Residen3.n3> ;
11 pl:hasLicense resultedScientificPolicy ;
12 pl:object <CompositePersonalData.n3> ;
13 pl:storageLocality <http://dbpedia.org/resource/Mexico> ;
14 pl:usageLocality <http://dbpedia.org/resource/Mexico> ;
15 pl:maxUses 3 .
```

b) Scenario 2

Figure 5.7: Composite policies of both scenarios.

5.3 CaLi: a lattice-based model for license classifications

Usually but not always, when l_i is less restrictive than l_j then l_i is compatible with l_j. For instance, see Fig. 5.8 that shows an excerpt of three Creative Commons (CC)[7] licenses described in RDF and using the ODRL vocabulary[8]. Notice that there exists a restrictiveness order among these licenses, (a) is less restrictive than (b) and (b) is less restrictive than (c). By transitivity (a) is less restrictive than (c). Notice also that (a) is compatible with (b) and (c), but (b) is not compatible with (c). This is due to the semantics of the prohibited action *DerivativeWorks* that forbids the distribution of a derivation (remix, transform or build upon) of the protected resource under a different license. Thus, depending on the semantics of their actions, a restrictiveness relation between two licenses does not imply a compatibility relation.

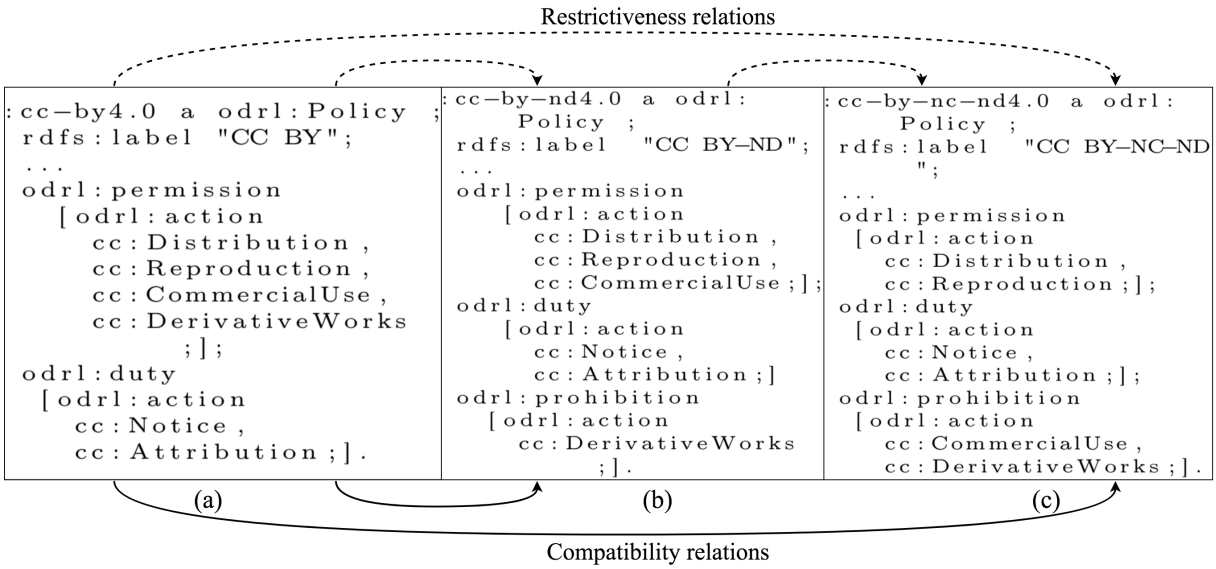


Figure 5.8: Three Creative Commons licenses described in RDF.

This section presented a composition process that produces a policy that is compliant to a set of policies. Next section generalizes the compatibility and compliance relation among whatever set of licenses (*i.e.*, only the part of policies that defines permissions, obligations and prohibitions).

The approach we propose to partially order licenses in terms of compatibility and compliance passes through a restrictiveness relation. In a license, actions can be distributed in what we call *status*, e.g., permissions, obligations and prohibitions. To decide if a license l_i is less restrictive than l_j, it is necessary to know if an action in a status is considered as less restrictive than the same action in another status. In the introductory example (Fig. 5.8), we consider that permissions are less restrictive than obligations, which are less restrictive than prohibitions, i.e., $Permission \leqslant Duty \leqslant Prohibition$.

We remark that if two licenses have a restrictiveness relation then it is possible that they have a compatibility relation too. The restrictiveness relation between the licenses

[7] https://creativecommons.org/

[8] The term *duty* is used for obligations https://www.w3.org/TR/odrl-model/

can be automatically obtained according to the status of actions without taking into account the semantics of the actions. Thus, based on lattice-ordered sets [26], we define a restrictiveness relation among licenses.

To identify the compatibility among licenses, we refine the restrictiveness relation with constraints. The goal is to take into account the semantics of actions. Constraints also distinguish valid licenses from non-valid ones. We consider a license l_i as non-valid if a resource can not be licensed under l_i, e.g., a license that simultaneously permits the *Derive* action[9] and prohibits *DerivativeWorks*[10].

This approach is based on:

1. a set of *actions* (e.g., *read, modify, distribute*, etc.);
2. a *restrictiveness lattice of status* that defines (i) all possible status of an action in a license (i.e., permission, obligation, prohibition, recommendation, undefined, etc.) and (ii) the restrictiveness relation among status; a *restrictiveness lattice of licenses* is obtained from a combination of 1 and 2;
3. a set of *compatibility constraints* to identify if a restrictiveness relation between two licenses is also a compatibility relation; and
4. a set of *license constraints* to identify non-valid licenses.

Next section introduces formally the CaLi model and Section 5.3.1 introduces a simple example of a CaLi ordering.

5.3.1 Formal model description

We first define a *restrictiveness lattice of status*. We use a lattice structure because it is necessary, for every pair of status, to know which status is less (or more) restrictive than both.

Definition 14 (Restrictiveness lattice of status $\mathcal{LS}$)**.** *A* restrictiveness lattice of status *is a lattice* $\mathcal{LS} = (S, \leqslant_S)$ *that defines all possible* status S *for a license and the relation* $\leqslant_S$ *as the* restrictiveness relation *over* S*. For two status* s_i*,* s_j*, if* $s_i \leqslant_S s_j$ *then* s_i *is less restrictive than* s_j*.*

Different $\mathcal{LS}$s can be defined according to the application domain. Fig. 5.9a shows the diagram of a $\mathcal{LS}$ inspired by file systems where actions can be either prohibited or permitted. With this lattice, prohibiting to read a file is more restrictive than permitting to read it. Fig. 5.9b illustrates a $\mathcal{LS}$ for CC licenses where actions are either permitted, required (Duty) or prohibited. Fig. 5.9c shows a $\mathcal{LS}$ inspired by the ODRL vocabulary. In ODRL, actions can be either permitted, obliged, prohibited or not specified (i.e., undefined). In this lattice, the undefined status is the least restrictive and the prohibited one the most restrictive. Fig. 5.9d shows a $\mathcal{LS}$ where a recommended or permitted action is less restrictive than the same action when it is permitted and recommended.

Now we formally define a license based on the status of its actions.

[9] https://www.w3.org/TR/odrl-vocab/#term-derive
[10] https://www.w3.org/TR/odrl-vocab/#term-DerivativeWorks

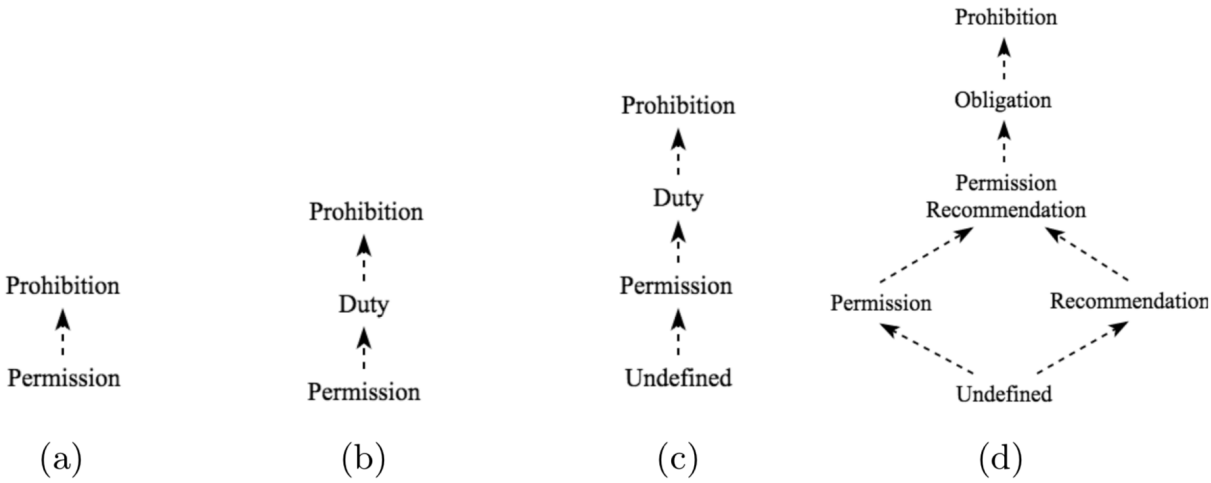


Figure 5.9: Examples of restrictiveness lattices of status ($\mathcal{LS}$). Dashed arrows represent restrictiveness, e.g., in (a) Permission is less restrictive than Prohibition

Definition 15 (License)**.** *Let $\mathcal{A}$ be a set of actions and $\mathcal{LS} = (S, \leqslant_S)$ be a restrictiveness lattice of status. A* license *is a function $l : \mathcal{A} \to S$. We denote by $\mathcal{L}_{\mathcal{A},\mathcal{LS}}$ the set of all licenses.*

For example, consider $\mathcal{A} = \{read, modify, distribute\}$, $\mathcal{LS}$ the lattice of Fig. 5.9c and two licenses: l_i which permits *read* and *distribute* but where *modify* is undefined and l_j where *modify* is also undefined but which permits *read* and prohibits *distribute*. We define l_i and l_j as follows:

$\forall a \in \mathcal{A}$:

$$l_i(a) = \begin{cases} \text{Undefined} & \text{if } a \in \{modify\}; \\ \text{Permission} & \text{if } a \in \{read, distribute\}. \end{cases}$$

$$l_j(a) = \begin{cases} \text{Undefined} & \text{if } a \in \{modify\}; \\ \text{Permission} & \text{if } a \in \{read\}; \\ \text{Prohibition} & \text{if } a \in \{distribute\}. \end{cases}$$

A restrictiveness lattice of status and a set of licenses make possible to partially order licenses in a restrictiveness lattice of licenses.

Definition 16 (Restrictiveness relation over licenses)**.** *Let A be a set of actions and $\mathcal{LS} = (S, \leqslant_S)$ be a restrictiveness lattice of status associated to the join and meet operators $\vee_S$ and $\wedge_S$, and $l_i, l_j \in \mathcal{L}_{\mathcal{A},\mathcal{LS}}$ be two licenses. We say that l_i is less restrictive than l_j, denoted $l_i \leqslant_{\mathcal{R}} l_j$, if for all actions $a \in \mathcal{A}$, the status of a in l_i is less restrictive than the status of a in l_j. That is, $l_i \leqslant_{\mathcal{R}} l_j$ if $\forall a \in \mathcal{A}, l_i(a) \leqslant_S l_j(a)$.*

Moreover, we define the two operators $\vee$ and $\wedge$ as follows. For all actions $a \in \mathcal{A}$, the status of a in $l_i \vee l_j$ (resp. $l_i \wedge l_j$) is the join (resp. meet) of the status of a in l_i and the status of a in l_j. That is, $(l_i \vee l_j)(a) = l_i(a) \vee_S l_j(a)$ and $(l_i \wedge l_j)(a) = l_i(a) \wedge_S l_j(a)$.

For example, consider $\mathcal{LS}$ the lattice of Fig. 5.9c, and licenses l_i and l_j defined previously; $l_i \leqslant_{\mathcal{R}} l_j$ because $l_i(read) \leqslant_S l_j(read), l_i(modify) \leqslant_S l_j(modify)$ and $l_i(distribute) \leqslant_S l_j(distribute)$. In this example, $l_i \vee l_j = l_j$ because $\forall a \in \mathcal{A}$, $(l_i \vee l_j)(a) = l_j(a)$, e.g., $(l_i \vee l_j)(distribute) = l_j(distribute) = Prohibition$. If for an action, it is not possible to say

which license is the most restrictive then the compared licenses are not comparable by the restrictiveness relation.

Remark 1. *The pair* $(\mathcal{L}_{\mathcal{A},\mathcal{LS}}, \leqslant_{\mathcal{R}})$ *is a restrictiveness lattice of licenses, whose* $\vee$ *and* $\wedge$ *are respectively the join and meet operators.*

In other words, for two licenses l_i and l_j, $l_i \vee l_j$ (resp. $l_i \wedge l_j$) is the least (resp. most) restrictive license that is more (resp. less) restrictive than both l_i and l_j.

Remark 2. *For an action* $a \in \mathcal{A}$*, we call* $(\mathcal{L}_{\{a\},\mathcal{LS}}, \leqslant_{\mathcal{R}})$ *the* action lattice *of* a*. Remark that* $(\mathcal{L}_{\mathcal{A},\mathcal{LS}}, \leqslant_{\mathcal{R}})$ *and* $\prod_{a\in\mathcal{A}}(\mathcal{L}_{\{a\},\mathcal{LS}}, \leqslant_{\mathcal{R}})$ *are isomorphic. That is, a restrictiveness lattice of licenses can be generated through the coordinatewise product [26] of all its* action lattices*. The total number of licenses in this lattice is* $|\mathcal{LS}|^{|\mathcal{A}|}$*.*

For example, consider $\mathcal{A} = \{read, modify\}$, $\mathcal{LS}$ the lattice of Fig. 5.9a, $(\mathcal{L}_{\mathcal{A},\mathcal{LS}}, \leqslant_{\mathcal{R}})$ is isomorphic to $(\mathcal{L}_{\{read\},\mathcal{LS}}, \leqslant_{\mathcal{R}}) \times (\mathcal{L}_{\{modify\},\mathcal{LS}}, \leqslant_{\mathcal{R}})$. Figure 5.10a,b,c illustrates the product of these action lattices and the produced restrictiveness lattice of licenses.

To identify the compatibility relation among licenses and to distinguish valid licenses from non-valid ones it is necessary to take into account the semantics of actions. Thus, we apply two types of constraints to the restrictiveness lattice of licenses: license constraints and compatibility constraints.

Definition 17 (License constraint)**.** *Let* $\mathcal{L}_{\mathcal{A},\mathcal{LS}}$ *be a set of licenses. A license constraint is a function* $\omega_{\mathcal{L}} : \mathcal{L}_{\mathcal{A},\mathcal{LS}} \rightarrow$ *Boolean which identifies if a license is valid or not.*

For example, the license constraint $\omega_{\mathcal{L}_1}$ considers a license $l_i \in \mathcal{L}_{\mathcal{A},\mathcal{LS}}$ non-valid if *read* is prohibited but modification is permitted (i.e., a *modify* action implies a *read* action):

$$\omega_{\mathcal{L}_1}(l_i) = \begin{cases} False & \text{if } l_i(read) = \text{Prohibition and } l_i(modify) = \text{Permission;} \\ True & \text{otherwise.} \end{cases}$$

Definition 18 (Compatibility constraint)**.** *Let* $(\mathcal{L}_{\mathcal{A},\mathcal{LS}}, \leqslant_{\mathcal{R}})$ *be a restrictiveness lattice of licenses. A compatibility constraint is a function* $\omega_{\rightarrow} : \mathcal{L}_{\mathcal{A},\mathcal{LS}} \times \mathcal{L}_{\mathcal{A},\mathcal{LS}} \rightarrow$ *Boolean which constraints the restrictiveness relation* $\leqslant_{\mathcal{R}}$ *to identify compatibility relations among licenses.*

For example, consider that a license prohibits the action *modify*. In the spirit of *Derivative Work*, we consider that the distribution of the modified resource under a different license is prohibited. Thus, the compatibility constraint $\omega_{\rightarrow_1}$, considers that a restrictiveness relation $l_i \leqslant_{\mathcal{R}} l_j$ can be also a compatibility relation if l_i does not prohibit *modify*. This constraint is described as:

For $l_i, l_j \in \mathcal{L}_{\mathcal{A},\mathcal{LS}}$,

$$\omega_{\rightarrow_1}(l_i, l_j) = \begin{cases} False & \text{if } l_i(modify) = \text{Prohibition;} \\ True & \text{otherwise.} \end{cases}$$

Now we are able to define a CaLi ordering from a restrictiveness lattice of licenses and constraints defined before.

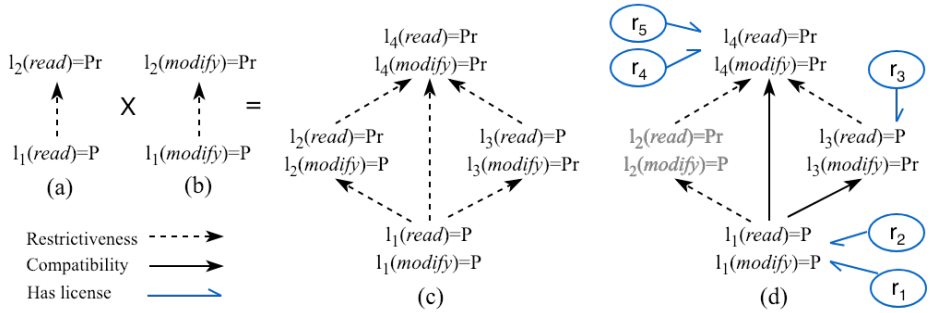


Figure 5.10: (a) and (b) are the action latices $(\mathcal{L}_{\{read\},\mathcal{LS}}, \leqslant_{\mathcal{R}})$ and $(\mathcal{L}_{\{modify\},\mathcal{LS}}, \leqslant_{\mathcal{R}})$, where $\mathcal{A} = \{read, modify\}$ and $\mathcal{LS}$ is the lattice of Fig. 5.9a (Pr=Prohibition and P=Permission). The product of these action lattices gives the restrictiveness lattice of licenses (c) $(\mathcal{L}_{\mathcal{A},\mathcal{LS}}, \leqslant_{\mathcal{R}})$ (reflexive relations are not represented). (d) is the CaLi ordering $\langle \mathcal{A}, \mathcal{LS}, \{\omega_{\mathcal{L}_1}\}, \{\omega_{\rightarrow_1}\}\rangle$.

Definition 19 (CaLi ordering)**.** *A CaLi ordering is a tuple* $\langle \mathcal{A}, \mathcal{LS}, C_{\mathcal{L}}, C_{\rightarrow}\rangle$ *such that* $\mathcal{A}$ *and* $\mathcal{LS}$ *form a restrictiveness lattice of licenses* $(\mathcal{L}_{\mathcal{A},\mathcal{LS}}, \leqslant_{\mathcal{R}})$*,* $C_{\mathcal{L}}$ *is a set of license constraints and* $C_{\rightarrow}$ *is a set of compatibility constraints. For two licenses* $l_i \leqslant_{\mathcal{R}} l_j \in \mathcal{L}_{\mathcal{A},\mathcal{LS}}$*, we say that* l_i is compatible with l_j*, denoted by* $l_i \rightarrow l_j$*, if* $\forall \omega_{\mathcal{L}} \in C_{\mathcal{L}}, \omega_{\mathcal{L}}(l_i) = \omega_{\mathcal{L}}(l_j) = True$ *and* $\forall \omega_{\rightarrow} \in C_{\rightarrow}, \omega_{\rightarrow}(l_i, l_j) = True$*.*

Remark 3. *We define the* compliance relation *as the opposite of the* compatibility relation*. For two licenses* l_i, l_j*, if* $l_i \rightarrow l_j$ *then* l_j is compliant with l_i*.*

A CaLi ordering is able to answer our research question, *given a license* l_i*, how to automatically position* l_i *over a set of licenses in terms of compatibility and compliance?* It allows to evaluate the potential reuse of a resource depending on its license. Knowing the compatibility of a license allows to know to which extent the protected resource is reusable. On the other hand, knowing the compliance of a license allows to know to which extent other licensed resources can be reused. Next section shows an example of CaLi ordering.

Example 1

Consider a CaLi ordering $\langle \mathcal{A}, \mathcal{LS}, \{\omega_{\mathcal{L}_1}\}, \{\omega_{\rightarrow_1}\}\rangle$ such that:

- $\mathcal{A}$ is the set of actions $\{read,\ modify\}$,
- $\mathcal{LS}$ is a restrictiveness lattice of status where an action can be either permitted or prohibited, and $Permission \leqslant_S Prohibition$ (cf Fig. 5.9a),
- $\omega_{\mathcal{L}_1}$ is the license constraint introduced in the example of Def. 17, and
- $\omega_{\rightarrow_1}$ is the compatibility constraint introduced in the example of Def. 18.

Fig. 5.10d shows a visual representation of this CaLi ordering. Licenses in grey are identified as non-valid by $\omega_{\mathcal{L}_1}$. They are part of the ordering but cannot protect resources. Dashed arrows represent restrictiveness relations $\leqslant_{\mathcal{R}}$. Black arrows represent restrictiveness relations that are also compatibility relations.

Consider a set of resources $R = \{r_1, r_2, r_3, r_4, r_5\}$. $\rightharpoonup$ is the *has license* relation such that $\{r_1, r_2\} \rightharpoonup l_1; r_3 \rightharpoonup l_3; \{r_4, r_5\} \rightharpoonup l_4$. Thanks to our CaLi ordering, next questions can be answered.

- *Which licensed resources can be reused in a resource that has as license l_3?* Those resource whose licenses are compatible with l_3: r_1 and r_2 that have license l_1 which precedes l_3, as well as r_3 that has the license l_3 itself.
- *Which licensed resources can reuse a resource that has as license l_1?* Those resource whose licenses are compliant with l_1: r_3, r_4 and r_5 that have licenses l_3 and l_4 which follow l_1, as well as r_1 and r_2 that have the license l_3 itself.

Resulting licenses can be returned ordered in a graph of compatibility.

We illustrated CaLi with a simple restrictiveness lattice of status, next section introduces a more realistic CaLi ordering inspired by licenses of Creative Commons.

5.3.2 A CaLi ordering for Creative Commons

Creative Commons proposes 7 licenses that are legally verified, free of charge, easy-to-understand and widely used when publishing resources on the Web. These licenses use 7 actions that can be permitted, required or prohibited. In this CaLi example, we search to model a complete compatibility ordering of all possible valid licenses using these 7 actions.

Description of a CC ordering based on CaLi

Consider $CC_$CaLi, a CaLi ordering $\langle \mathcal{A}, \mathcal{LS}, C_{\mathcal{L}}, C_{\rightarrow} \rangle$ such that:

- $\mathcal{A}$ is the set of actions *{cc:Distribution, cc:Reproduction, cc:DerivativeWorks, cc:CommercialUse, cc:Notice, cc:Attribution, cc:ShareAlike}*,
- $\mathcal{LS}$ is the restrictiveness lattice of status depicted in 5.9b[11], and
- $C_{\mathcal{L}}, C_{\rightarrow}$ are the sets of constraints defined next.

$C_{\mathcal{L}} = \{\omega_{\mathcal{L}_2}, \omega_{\mathcal{L}_3}\}$ allows to invalidate a license (1) when *cc:CommercialUse* is required and (2) when *cc:ShareAlike* is prohibited:

$$\omega_{\mathcal{L}_2}(l_i) = \begin{cases} False & \text{if } l_i(\textit{cc:CommercialUse}) = \text{Duty;} \\ True & \text{otherwise.} \end{cases}$$

$$\omega_{\mathcal{L}_3}(l_i) = \begin{cases} False & \text{if } l_i(\textit{cc:ShareAlike}) = \text{Prohibition;} \\ True & \text{otherwise.} \end{cases}$$

$C_{\rightarrow} = \{\omega_{\rightarrow_2}, \omega_{\rightarrow_3}\}$ allows to identify (1) when *cc:ShareAlike* is required and (2) when *cc:DerivativeWorks* is prohibited. That is because *cc:ShareAlike* requires that the distribution of derivative works be under the same license only, and *cc:DerivativeWorks*, when prohibited, does not allow the distribution of a derivative resource, regardless of the license.

$$\omega_{\rightarrow_2}(l_i, l_j) = \begin{cases} False & \text{if } l_i(\textit{cc:ShareAlike}) = \text{Duty;} \\ True & \text{otherwise.} \end{cases}$$

[11] To simplify, we consider that a requirement is a duty.

$$\omega_{\rightarrow_3}(l_i, l_j) = \begin{cases} False & \text{if } l_i(cc{:}DerivativeWorks) = \text{Prohibition;} \\ True & \text{otherwise.} \end{cases}$$

Other constraints could be defined to be closer to the CC schema[12] but for the purposes of this compatibility ordering these constraints are enough.

Analysis of $CC_$CaLi

The size of the restrictiveness lattice of licenses is 3^7 but the number of valid licenses of $CC_$CaLi is 972 due to $C_{\mathcal{L}}$. That is, 5 actions in whatever status and 2 actions (*cc:CommercialUse* and *cc:ShareAlike*) in only 2 status: $3^5 * 2^2$.

The following $CC_$CaLi licenses are like the official CC licenses.

$$CCBY(a) = \begin{cases} \text{Permission} & \text{if } a \in \{cc{:}Distribution,\ cc{:}Reproduction \\ & \quad cc{:}DerivativeWorks,\ cc{:}CommercialUse \\ & \quad cc{:}ShareAlike\}; \\ \text{Duty} & \text{if } a \in \{cc{:}Notice,\ cc{:}Attribution\}. \end{cases}$$

$$CCBYNC(a) = \begin{cases} \text{Permission} & \text{if } a \in \{cc{:}Distribution,\ cc{:}Reproduction \\ & \quad cc{:}DerivativeWorks,\ cc{:}ShareAlike\}; \\ \text{Duty} & \text{if } a \in \{cc{:}Notice,\ cc{:}Attribution\}; \\ \text{Prohibition} & \text{if } a \in \{cc{:}CommercialUse\}. \end{cases}$$

The following $CC_$CaLi licenses are not part of the official CC licenses. License $CC\ l_1$ is like CC BY-NC but without the obligation to give credit to the copyright holder/author of the resource. $CC\ l_2$ is like CC BY but with the prohibition of making multiple copies of the resource. License $CC\ l_3$ allows only exact copies of the original resource to be distributed. $CC\ l_4$ is like $CC\ l_3$ with the prohibition of commercial use.

$$CC\ l_1(a) = \begin{cases} \text{Permission} & \text{if } a \in \{cc{:}Distribution,\ cc{:}Reproduction, \\ & \quad cc{:}DerivativeWorks,\ cc{:}ShareAlike, \\ & \quad cc{:}Notice,\ cc{:}Attribution\}; \\ \text{Prohibition} & \text{if } a \in \{cc{:}CommercialUse\}. \end{cases}$$

$$CC\ l_2(a) = \begin{cases} \text{Permission} & \text{if } a \in \{cc{:}Distribution,\ cc{:}DerivativeWorks, \\ & \quad cc{:}CommercialUse,\ cc{:}ShareAlike\}; \\ \text{Duty} & \text{if } a \in \{cc{:}Notice,\ cc{:}Attribution\}; \\ \text{Prohibition} & \text{if } a \in \{cc{:}Reproduction\}. \end{cases}$$

$$CC\ l_3(a) = \begin{cases} \text{Permission} & \text{if } a \in \{cc{:}Distribution,\ cc{:}ShareAlike,\ cc{:}CommercialUse\}; \\ \text{Duty} & \text{if } a \in \{cc{:}Notice,\ cc{:}Attribution,\ cc{:}Reproduction\}; \\ \text{Prohibition} & \text{if } a \in \{cc{:}DerivativeWorks\}. \end{cases}$$

$$CC\ l_4(a) = \begin{cases} \text{Permission} & \text{if } a \in \{cc{:}Distribution,\ cc{:}ShareAlike\}; \\ \text{Duty} & \text{if } a \in \{cc{:}Notice,\ cc{:}Attribution,\ cc{:}Reproduction\}; \\ \text{Prohibition} & \text{if } a \in \{cc{:}DerivativeWorks,\ cc{:}CommercialUse\}. \end{cases}$$

In $CC_$CaLi, the minimum is the license where all actions are permitted (i.e., CC Zero) and the maximum is the license where all actions are prohibited.

[12] https://creativecommons.org/ns

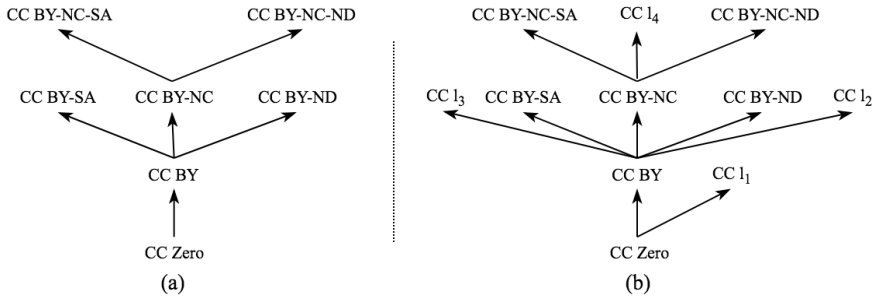


Figure 5.11: Compatibility subgraphs of $CC_$CaLi: (a) contains the 7 official CC licenses and (b) contains $CC\ l_1$ to $CC\ l_4$ in addition to the 7 official CC licenses.

Fig. 5.11 shows two subgraphs of $CC_$CaLi with only the compatibility relations. Fig. 5.11a shows only the 7 official CC licenses and Fig. 5.11b includes also $CC\ l_1$ to $CC\ l_4$. These graphs can be generated using the CaLi implementation (cf Section 5.3.3). Thanks to $\omega_{\rightarrow_2}$, the restrictiveness relation between CC BY-SA and CC BY-NC-SA is not identified as a compatibility relation and thanks to $\omega_{\rightarrow_3}$, the restrictiveness relation between CC BY-ND and CC BY-NC-ND is not identified as a compatibility relation. We recall that a license that prohibits *cc:DerivativeWorks* is not compatible even with itself.

The compatibility relations of Fig. 5.11a are conform to the ones obtained from the Web2rights tool. This example shows the usability of CaLi with a real set of licenses.

5.3.3 Implementation of CaLi orderings

The goal of this section is twofold, to analyse the algorithm we implemented to produce CaLi orderings and to illustrate the usability of CaLi through a prototype of a license-based search engine.

Experimental validation

The size growth of CaLi orderings is exponential, i.e., $|\mathcal{LS}|^{|\mathcal{A}|}$. Nevertheless, it is not necessary to explicitly build a CaLi ordering to use it. Sorting algorithms like insertion sort can be used to produce subgraphs of a CaLi ordering.

We implemented an algorithm that can sort any set of licenses using the $\mathcal{LS}$ of Fig. 5.9c in $\sum_{i=0}^{n-1} i = \frac{n(n-1)}{2}$ comparisons of restrictiveness (approx. $n^2/2$), n being the number of licenses to sort, i.e., $O(n^2)$. The goal is to be able to insert a license in a graph in linear time $O(n)$ without sorting again the graph.

We use a heuristic, based on the restrictiveness of the new license, to chose between two strategies, *1)* to insert a license traversing the graph from the minimum or *2)* from the maximum. To do this, our algorithm calculates the relative position of the new license (node) from the number of actions that it obliges and prohibits. The median depth (number of levels) of the existing graph is calculated from the median of the number of prohibited and obliged actions of existing licenses. Depending on these numbers, a strategy is chosen to find the place of the new license in the graph.

Results shown in Fig. 5.12 demonstrate that our algorithm sorts a set of licenses with at most $n^2/2$ comparisons. We used 20 subsets of licenses of different sizes from the

$CC_$CaLi ordering. Size of subsets was incremented by 100 up to 2187 licenses. Each subset was created and sorted 3 times randomly. The curve was produced with the average of the number of comparisons to sort each subset.

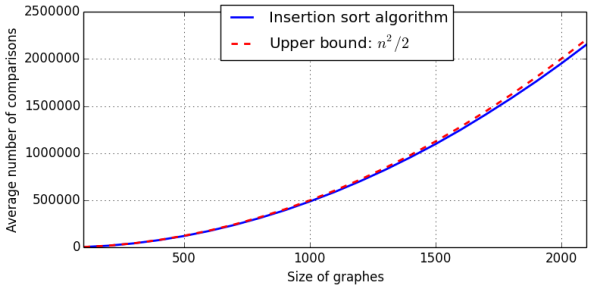


Figure 5.12: Performance of the implemented insertion sort algorithm in number of comparisons of restrictiveness with incremental size of subsets of licenses.

A comparison of restrictiveness takes on average 6 milliseconds[13], thus to insert a license in a 2000 licenses graph takes an average of 12 seconds. Building a whole graph is time consuming (a 2000 licenses graph takes on average 8 hours to sort) but this time can be reduced with further optimisations of the process to compare the restrictiveness of two licenses. The implementation in Python of our algorithm and details of our experiments are available on GitHub[14].

A search engine based on an ODRL CaLi ordering

We implemented a prototype of a search engine that allows to find linked data[15] and source code repositories[16] based on the compatibility or the compliance of their licenses. We use licenses described with the ODRL vocabulary. ODRL proposes properties to define semantic dependencies among actions[17] that we translate as CaLi constraints. *Included In* is defined as "An Action transitively asserts that another Action encompasses its operational semantics". *Implies* is defined as "An Action asserts that another Action is not prohibited to enable its operational semantics". Thereby we consider that if an action a_i is included in another action a_j then a_i implies a_j. For example, *CommercialUse* is included in *use*, therefore we consider that *CommercialUse* implies *use*. That means that if *CommercialUse* is permitted then *use* should be permitted too. To preserve this dependency we implemented the constraint $\omega_{\mathcal{L}_4}$.

$$\omega_{\mathcal{L}_4}(l_i) = \begin{cases} False & \text{if } a_i \text{ odrl:includedIn } a_j \\ & \text{AND } (l_i(a_i) = \text{Permitted OR } l_i(a_i) = \text{Obliged}) \\ & \text{AND } l_i(a_j) = \text{Prohibited;} \\ True & \text{otherwise.} \end{cases}$$

We use $ODRL_$CaLi, a CaLi ordering $\langle \mathcal{A}, \mathcal{LS}, C_{\mathcal{L}}, C_{\rightarrow} \rangle$ such that:

[13] With a 160xIntel(R) Xeon(R) CPU E7-8870 v4 2.10GHz 1,5 Tb RAM.
[14] https://github.com/benjimor/CaLi-Search-Engine
[15] http://cali.priloo.univ-nantes.fr/ld/
[16] http://cali.priloo.univ-nantes.fr/rep/
[17] https://www.w3.org/TR/odrl-vocab/#actionConcepts

- $\mathcal{A}$ is the set of 72 actions of ODRL,
- $\mathcal{LS}$ is the restrictiveness lattice of status of Fig. 5.9c,
- $C_{\mathcal{L}} = \{\omega_{\mathcal{L}_2}, \omega_{\mathcal{L}_3}, \omega_{\mathcal{L}_4}\}$, and
- $C_{\rightarrow} = \{\omega_{\rightarrow_2}, \omega_{\rightarrow_3}\}$.

The size of this ordering is 4^{72} and it is not possible to build it. This search engine illustrates the usability of $ODRL_$CaLi through two subgraphs. On the one side, there is a subgraph with the most used licenses in DataHub[18] and OpenDataSoft. Licenses in this graph are linked to some RDF datasets such that it is possible to find datasets whose licenses are compatible (or compliant) with a particular license. On the other side, there is a subgraph with the most used licenses in GitHub. Here, licenses are linked to some GitHub repositories and it is possible to find repositories whose licenses are compatible (or compliant) with a particular license.

5.4 Conclusion

PrODUCE, the policies composition process I presented in this chapter, uses basic operators and ontology-based rules that consider data usage context.

Implicit terms, based on the usage context, extend usage policies leading to additional inconsistencies during the composition process. However, most of these inconsistencies can be eliminated with contextual rules that may incorporate priorities. This approach is very flexible because, new aspects of data usage context can be easily included by extending the PriLoo ontology and defining, accordingly, the set of rules necessary for the composition.

Future works include the definition of rules for other contextual aspects as the laws of the usage and storage locations of concerned data. Another research direction is to analyze how to construct a feedback when the policies combination is not possible and return it to the user, instead of a false result.

I proposed a lattice-based model to define compatibility and compliance relations among licenses. Our approach is based on a restrictiveness relation that is refined with constraints to take into account the semantics of actions existing in licenses. We have shown the feasibility of our approach through two CaLi orderings, one using the Creative Commons vocabulary and the second using ODRL. We experimented the production of CaLi orderings with the implementation of an insertion sort algorithm whose cost is $n^2/2$. We implemented a prototype of a license-based search engine that highlights the feasibility and usefulness of our approach. Our compatibility model does not intent to provide a legal advice but it allows to exclude those licenses that would contravene a particular license.

A perspective of this work is to take into account other aspects of licenses related to usage contexts like jurisdiction, dates of reuse, *etc.* Another perspective is to analyse how two compatibility orderings can be compared. That is, given two CaLi orderings, if there is an alignment between their vocabularies and their restrictiveness lattices of status are homomorphic then find a function to pass from a CaLi ordering to another.

[18] https://old.datahub.io/

Chapter 6

On-going work and perspectives

In this document, I provide some relevant research activities that I have developed since my recruitement as Assistant Professor at the Nantes University. During this period, from 2005 to 2019, I co-supervised 4 PhD students (one in progress), 2 post-docs and several Master projects. I participated in several international, national and regional projects and published more than 50 articles in national and international conferences and journals.

Since 2007, the interest on the Web of data knows a significant and regular growth. Having data as linked data, in RDF, enables huge amounts of datasets to be interconnected on the Web, creating new and innovative applications. In the continuation of my on-going research, I will keep working in the Linked Data domain because it opens interesting research perspectives. I am interested in two complementary research questions (i) how to promote the integration of existing datasets and Web APIs (public and private) into the Linked Data, so that many datasets are virtually integrated and (ii) how to make the Linked Data an environment in which the virtual integration of widely distributed datasets respects the licenses or policies of all datasets involved in processing a query. Next sections give some details about these perspectives.

Promoting integration of existing datasets into the Linked Data

One way of integrating structured datasets (column-based, JSON, *etc.*) into the Linked Data is through *RDF mappings*. An RDF mapping consists in a set of rules that define the transformation of an initial dataset into an RDF dataset. Traditionally, RDF mappings are used to materialize structured datasets into RDF data. But, to avoid expensive investments in terms of storage and time, some data providers use RDF mappings to integrate virtually and on-demand, non-RDF data as Linked Data [80, 84].

In this context, I will tackle two issues, how to asses the quality of RDF mappings and how to exploit existing mappings to integrate new structured datasets to the Linked Data.

Assessing the quality of RDF mappings

Making a good mapping for a dataset is a challenging task. It requires to answer several questions, for instance: (i) what are the resources described in the dataset? (ii) what are the attributes of these resources? (iii) what is the set of attributes that can uniquely

identify each resource? (iv) what are the relations among resources? (v) which ontologies are pertinent to describe these resources, as well as the relations and the attributes of these resources?

RDF mappings can be written in different languages such as RML [29], SPARQL-Generate [72], or YARRRML [45]. There exist also semi-automated frameworks such as Karma [41], RMLEditor [46] or Juma [64]. Multiple mappings with different qualities for the same initial dataset can be defined. Therefore, how can we know which one is better than another? What are the characteristics that indicate to which extent a mapping is good?

Several works have been proposed to asses the quality of RDF datasets. [119] surveys the state of the art in this domain. To our knowledge very few works focus on mapping quality assessment. [28] proposes a framework that assess and refine RDF mappings but this framework focuses only on logical errors that can be corrected automatically. Although, there are many other important features to assess a mapping like maintainability and human readability aspects. Thus, I aim at proposing a framework to assess the quality of mappings based on an initial semi-structured dataset and a corresponding mapping.

Exploiting RDF mappings to integrate new structured datasets

Data providers who are reluctant to integrate their structured datasets to the Linked Data, want to know to which extent their datasets might benefit from existing semantic datasets. That is, with which datasets of the Linked Data their dataset can be combined (*i.e.*, queried), knowing that the dataset to be integrated is not yet materialized in RDF and that only an RDF mapping can be defined. In particular, a list of ranked datasets according to their degree of conjunction/disjunction with a particular dataset would be very valuable.

In the state of the art, [38] provides an approach for generating federated SPARQL queries based on RDF datasets. This approach allows flexible parameterization of realistic conjunctive benchmark queries. Query parameterization includes structure, complexity, and cardinality constraints. Similarly, [101] proposes a query generator which can generate a variety of federated SPARQL queries over a given set of RDF datasets. Their goal is to facilitate the process of benchmarking for SPARQL query federation systems. Generated queries are conjunctive and use the OPTIONAL and FILTER keywords. [42] proposes a solution for generating federated SPARQL queries from query logs executed in the past. The generated queries are not limited to conjunctive queries, but also queries using UNION (disjunction), OPTIONAL, FILTER, *etc.*

These solutions are not appropriate for our problem. On the one hand, the particular dataset that a provider wants to integrate with the Linked Data is not materialized in RDF and a log of SPARQL queries already executed does not exists. On the other hand, a ranked list of RDF datasets according to their degree of conjunction/disjunction with a particular dataset is not provided.

My work aims to provide an approach that exploits RDF mappings instead of RDF datasets or query logs. An RDF mapping can be seen as an RDF summary of the dataset it describes. An important number of RDF mappings exist and they can provide useful information to evaluate the conjunction/disjunction capabilities among virtually integrated datasets with a limited overhead. Thus, I aim at proposing an approach that based on

RDF mappings, produces a ranked list of datasets that can be queried with a particular structured dataset.

Defining SPARQL query engines that respect licenses

A key problem when publishing datasets that can be easily combined (*i.e.*, queried) with unpredictable datasets is how to be compliant of all licenses or policies of all datasets involved in a query. When two or more licensed datasets participate in the evaluation of a federated query, the result set must be protected by a license compliant with licenses of involved datasets. When combining licensed datasets it is not always possible to find a license compliant with all involved licenses, as I have shown with the CaLi model [86]. That means that even if a user can query individually each dataset of a federation, it should not be possible to execute some federated queries combining some datasets because the incompatibility of their licenses.

A solution to the compatibility of licenses is to negotiate licenses with data providers. But this negotiation takes time and is not always possible. Another solution is to discard datasets of conflicting licenses during the source selection process. This solution may lead to an empty query result. To tackle this problem, I propose to use query relaxation techniques. That is, to use relaxation rules to relax triple patterns in order to match triples of other datasets.

The number of possible rewritten queries can be important. It depends on the number of triple patterns and the relaxation possibilities of each triple pattern. There exist some approches that use similarity measures to find relevant relaxed queries [30, 31, 32, 33, 47, 49, 50]. In a distributed environment, the challenge is to limit the communication cost during the relaxation process, *i.e.*, how to limit the communication cost of a source selection process to identify the most relevant relaxed queries that produce non empty query results. In my future work, I aim at proposing a federated query engine that always returns a query result protected by a license compliant with each license of the involved datasets.

List of Figures

2.1 Graphical view of SocioPath as a UML class diagram. 7
2.2 Use case 1: isolated PC. 9
2.3 Use case 2: GoogleDocs. 14
2.4 Relations of access and control in the use case 2 (GoogleDocs). 15
2.5 Degree of dependence on persons' sets. 16
2.6 The system for the activity "John edits letter.doc" as a DAG of use case 1. . . . 18
2.7 LINA's WDAG for the activity "a user accesses a file on the SVN". 22
2.8 Graph transformation. 26
2.9 Value of the probability expectation for 50 persons using the three methods **mTNA**, **Copy** and **Split**. 27
2.10 Difference between the opinion's probability expectation of a graph using **mTNA**, **Copy**, and **Split** when $u = 0$ and the trust value resulting from using SocioTrust. 27

3.1 Privacy policy (PP) model 37
3.2 Global architecture 40
3.3 Comparison of the three algorithms of trust level searching 44
3.4 Stabilization of the cost of trust level searching 45
3.5 WUW Strategy. 47
3.6 WUW Architecture 50
3.7 Average download time. 53
3.8 Satisfaction as client. 54
3.9 Satisfaction as server. 54
3.10 Adequation as client. 55

4.1 Concurrent execution of federated queries CD3 and CD4 of FedBench over a federation of SPARQL endpoints. 59
4.2 Reversing BGPs from a federated log. 62
4.3 Precision of triple patterns. 67
4.4 Recall of triple patterns. 67
4.5 Precision of joins. 67
4.6 Recall of joins. 67
4.7 Two Union queries. 67
4.8 Recall of joins from Anapsid traces for MX by gap. 68
4.9 Recall of joins from FedX traces for MX by gap. 68
4.10 Average of precision of joins of the four mixes by gap. 69
4.11 Average of recall of joins of the four mixes by gap. 69

4.12 Concurrent execution of queries Q_1 and Q_2. 70
4.13 SWEEP dashboard. 72

5.8 Three Creative Commons licenses described in RDF. 86
5.9 Examples of restrictiveness lattices of status ($\mathcal{LS}$). Dashed arrows represent restrictiveness, e.g., in (a) Permission is less restrictive than Prohibition 88
5.10 (a) and (b) are the action latices ($\mathcal{L}_{\{read\},\mathcal{LS}}, \leqslant_{\mathcal{R}}$) and ($\mathcal{L}_{\{modify\},\mathcal{LS}}, \leqslant_{\mathcal{R}}$), where $\mathcal{A} = \{read, modify\}$ and $\mathcal{LS}$ is the lattice of Fig. 5.9a (Pr=Prohibition and P=Permission). The product of these action lattices gives the restrictiveness lattice of licenses (c) ($\mathcal{L}_{\mathcal{A},\mathcal{LS}}, \leqslant_{\mathcal{R}}$) (reflexive relations are not represented). (d) is the CaLi ordering $\langle \mathcal{A}, \mathcal{LS}, \{\omega_{\mathcal{L}_1}\}, \{\omega_{\rightarrow_1}\}\rangle$. 90
5.11 Compatibility subgraphs of CC_CaLi: (a) contains the 7 official CC licenses and (b) contains $CC\ l_1$ to $CC\ l_4$ in addition to the 7 official CC licenses. . . . 93
5.12 Performance of the implemented insertion sort algorithm in number of comparisons of restrictiveness with incremental size of subsets of licenses. 94

List of Tables

2.1 Glossary of notations (1). 8
2.2 Deduced access and control relations. 10
2.3 Sets of persons John depends on (use case 2 - GoogleDocs). 16
2.4 Different systems and their trust values. 21
2.5 User's trust value in the system SVN in LINA. 23
2.6 Users' opinions in the system for the activity "a user access a file on the SVN" of our research laboratory. 30

3.1 Purpose-based data reference table (PBDRT) of doctor Dj. 38
3.2 Notation used to describe feedback measures computation on each peer. 49

4.1 Partial federated log of CD3 executed by Anapsid ($E_{Anapsid}(CD3)$) where can be observed $\{tp_1^{@IT}.(tp_2.tp_3)^{@IB}.(tp_4.tp_5)^{@NYT}\}$. 60
4.2 Partial federated log of CD3 executed by FedX ($E_{FedX}(CD3)$) where can be observed $\{(tp_2.tp_3)^{@IB}.tp_1^{@IT}.tp_5^{@NYT}.tp_4^{@NYT}\}$. 61
4.3 Number of SELECT subqueries received for queries of Cross Domain (CD) and Life Science (LS) produced by Anapsid and FedX. 66
4.4 Excerpt of a DBpedia's TPF server log for query Q_1. 71

5.2 Operators used for policies composition. 84

Bibliography

[1] *1.0 P3P Purposes of Data Collection Elements. http://p3pwriter.com/LRN_041.asp.*

[2] M. Acosta, M. Vidal, T. Lampo, J. Castillo, and E. Ruckhaus. "ANAPSID: An Adaptive Query Processing Engine for SPARQL Endpoints". In: *International Semantic Web Conference (ISWC), Part I*. Bonn, Germany, Oct. 2011. DOI: 10.1007/978-3-642-25073-6_2.

[3] R. Agrawal, P. Bird, T. Grandison, J. Kiernan, S. Logan, and W. Rjaibi. "Extending Relational Database Systems to Automatically Enforce Privacy Policies". In: *IEEE Conference on Data Engineering (ICDE)*. Tokyo, Japan, Apr. 2005.

[4] R. Agrawal, J. Kiernan, R. Srikant, and Y. Xu. "Hippocratic Databases". In: *Very Large Databases (VLDB)*. Hong Kong, China, Aug. 2002.

[5] I. Agudo, C. Fernandez-Gago, and J. Lopez. "A Model for Trust Metrics Analysis". In: *Proceedings of the 5th International Conference on Trust, Privacy and Security in Digital Business (TrustBus)*. Turin, Italy, Sept. 2008, pp. 28–37.

[6] R. Akbarinia, V. Martins, E. Pacitti, and P. Valduriez. "Design and Implementation of APPA". In: *Global Data Management (Eds. R. Baldoni, G. Cortese, F. Davide), IOS Press* (2006).

[7] N. Alhadad. "Bridging the Gap between Social and Digital Worlds: System Modeling and Trust Evaluation". Theses. Université de Nantes, June 2014. URL: https://tel.archives-ouvertes.fr/tel-01112455.

[8] N. Alhadad, Y. Busnel, P. Serrano-Alvarado, and P. Lamarre. "Trust Evaluation of a System for an Activity with Subjective Logic". In: *TrustBus*. Vol. 8647. LNCS. Munich, Germany: Springer, Sept. 2014. DOI: DOI=10.1007/978-3-319-09770-1_5. URL: https://hal.archives-ouvertes.fr/hal-01059198.

[9] N. Alhadad, P. Lamarre, Y. Busnel, P. Serrano Alvarado, and M. Biazzini. "SocioPath: In Whom You Trust?" In: *Bases de Données Avancées (BDA)*. Short paper. Rabat, Morocco, Nov. 2011. URL: https://hal.archives-ouvertes.fr/hal-00638753.

[10] N. Alhadad, P. Lamarre, Y. Busnel, P. Serrano-Alvarado, M. Biazzini, and C. Sibertin-Blanc. "SocioPath: Bridging the Gap between Digital and Social Worlds". In: *23rd International Conference on Database and Expert Systems Applications (DEXA)*. Vol. 7446. LNCS. Short paper. Vienna, Austria: Springer, Sept. 2012. URL: https://hal.archives-ouvertes.fr/hal-00725098.

[11] N. Alhadad, P. Lamarre, P. Serrano-Alvarado, and Y. Busnel. "Trust Approach Based on User's Activities". In: *Atelier Protection de la Vie Privée (APVP)*. Ile de Groix, France, June 2012. URL: https://hal.archives-ouvertes.fr/hal-00755038.

[12] N. Alhadad, P. Lamarre, P. Serrano-Alvarado, Y. Busnel, and M. Biazzini. "SocioPath: In Whom You Trust?" In: *Atelier Protection de la Vie Privée / Géolocalisation et Vie Privée (APVP)*. Soreze, France, June 2011. URL: https://hal.archives-ouvertes.fr/hal-01362325.

[13] N. Alhadad, P. Serrano-Alvarado, Y. Busnel, and P. Lamarre. "System Modeling and Trust Evaluation of Distributed Systems". In: *Transactions on Large-Scale Data and Knowledge-Centered Systems, ISSN: 1869-1994*. LNCS 22.9430 (Sept. 2015). URL: https://hal.archives-ouvertes.fr/hal-01166896.

[14] N. Alhadad, P. Serrano-Alvarado, Y. Busnel, and P. Lamarre. "Trust Evaluation of a System for an Activity". In: *TrustBus*. Vol. 8058. LNCS. Prague, Czech Republic: Springer, Aug. 2013. URL: https://hal.archives-ouvertes.fr/hal-00853679.

[15] C. Basca and A. Bernstein. "Avalanche: Putting the Spirit of the Web back into Semantic Web Querying". In: *International Semantic Web Conference (ISWC)*. Shanghai, China, Nov. 2010. URL: http://ceur-ws.org/Vol-658/paper527.pdf.

[16] M. Biazzini, R. Carvajal-Gomez, A. Perez-Espinosa, P. Serrano-Alvarado, P. Lamarre, and E. Perez Cortes. *WUW (What Users Want): A Service to Enhance Users' Satisfaction in Content-Based Peer-to-Peer Networks*. Tech. rep. 20 pages. LINA-University of Nantes, Oct. 2012. URL: https://hal.archives-ouvertes.fr/hal-00744775.

[17] M. Biazzini, P. Serrano-Alvarado, and R. Carvajal-Gomez. "Towards improving user satisfaction in decentralized P2P networks". In: *COLLABORATECOM*. CORE C. Austin, Texas, United States: EAI, Oct. 2013. URL: https://hal.archives-ouvertes.fr/hal-00871672.

[18] R. Buyya, M. Pathan, and A. Vakali. *Content Delivery Networks*. Lecture Notes in Electrical Engineering. Springer, 2008. ISBN: 9783540778868. URL: http://books.google.co.jp/books?id=p2C2cZkTrmsC.

[19] J.-W. Byun and N. Li: "Purpose Based Access Control for Privacy Protection in Relational Database Systems". In: *Very Large Databases (VLDB) Journal* 17.4 (2008).

[20] F. Cappello, E. Caron, M. Dayde, F. Desprez, Y. Jegou, P. Primet, E. Jeannot, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, B. Quetier, and O. Richard. "Grid'5000: A Large Scale and Highly Reconfigurable Grid Experimental Testbed". In: *Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing*. GRID '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 99–106. ISBN: 0-7803-9492-5. DOI: 10.1109/GRID.2005.1542730. URL: http://dx.doi.org/10.1109/GRID.2005.1542730.

[21] R. Carvajal-Gómez and P. Serrano-Alvarado. "Evaluating WUW, a Service to Enhance users' Satisfaction in Content-Based Peer-to-Peer Networks". In: *Grid 5000 Winter School*. Nantes, France, Dec. 2012. URL: https://hal.archives-ouvertes.fr/hal-01362309.

[22] I. Clarke, S. G. Miller, T. W. Hong, O. Sandberg, and B. Wiley. "Protecting Free Expression Online with Freenet". In: *IEEE Internet Computing* 6.1 (2002). ISSN: 1089-7801.

[23] B. Cohen. "Incentives Build Robustness in BitTorrent". In: *Workshop on Economics of Peer-to-Peer Systems*. Berkley, CA, USA, June 2003.

[24] B. Cohen. *The BitTorrent Protocol Specification*. Feb. 2008. URL: http://www.bittorrent.org/.

[25] L. Cranor, M. Langheinrich, M. Marchiori, M. Presler-Marshall, and J. Reagle. *The Platform for Privacy Preferences 1.0 (P3P1.0) Specification*. 2002.

[26] B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order*. Cambridge university press, 2002.

[27] E. Desmontils, P. Serrano Alvarado, and P. Molli. "SWEEP: a Streaming Web Service to Deduce Basic Graph Patterns from Triple Pattern Fragments". In: *16th International Semantic Web Conference (ISWC)*. Demo paper. Vienna, Austria: Springer, Oct. 2017. URL: https://hal.archives-ouvertes.fr/hal-01583513.

[28] A. Dimou, D. Kontokostas, M. Freudenberg, R. Verborgh, J. Lehmann, E. Mannens, S. Hellmann, and R. Van de Walle. "Assessing and Refining Pappings to RDF to Improve Dataset Quality". In: *International Semantic Web Conference (ISWC)*. Bethlehem, Pennsylvania, USA, Oct. 2015.

[29] A. Dimou, M. Vander Sande, P. Colpaert, R. Verborgh, E. Mannens, and R. Van de Walle. "RML: A Generic Language for Integrated RDF Mappings of Heterogeneous Data". In: *Workshop on Linked Data on the Web (LDOW) collocated with WWW*. Lyon, France, Apr. 2014.

[30] P. Dolog, H. Stuckenschmidt, H. Wache, and J. Diederich. "Relaxing RDF Queries Based on User and Domain Preferences". In: *Journal of Intelligent Information Systems* 33 (2009).

[31] S. Ferré. "Answers Partitioning and Lazy Joins for Efficient Query Relaxation and Application to Similarity Search". In: *Extended Semantic Web Conference (ESWC)*. Heraklion, Crete, Greece, June 2018.

[32] G. Fokou, S. Jean, A. Hadjali, and M. Baron. "RDF Query Relaxation Strategies Based on Failure Causes". In: *Extended Semantic Web Conference (ESWC)*. Heraklion, Crete, Greece, May 2016.

[33] R. Frosini, A. Calì, A. Poulovassilis, and P. T. Wood. "Flexible Query Processing for SPARQL". In: *Journal of Semantic Web* 8 (2017).

[34] D. Gambetta. "Trust: Making and Breaking Cooperative Relations". In: vol. 13. Department of Sociology, University of Oxford, 2000. Chap. Can we Trust Trust, pp. 213–237.

[35] G. Gangadharan, M. Weiss, V. D'Andrea, and R. Iannella. "Service License Composition and Compatibility Analysis". In: *International Conference on Service-Oriented Computing (ICSOC)*. 2007.

[36] J. Golbeck and J. A. Hendler. "Inferring Binary Trust Relationships in Web-Based Social Networks". In: *ACM Transactions on Internet Technology* 6.4 (2006), pp. 497–529.

[37] O. Görlitz and S. Staab. "SPLENDID: SPARQL Endpoint Federation Exploiting VOID Descriptions". In: *International Workshop on Consuming Linked Data (COLD)*. Bonn, Germany, Oct. 2011. URL: http://ceur-ws.org/Vol-782/GoerlitzAndStaab_COLD2011.pdf.

[38] O. Görlitz, M. Thimm, and S. Staab. "SPLODGE: Systematic Generation of SPARQL Benchmark Queries for Linked Open Data". In: *International Semantic Web Conference (ISWC)*. Springer. Boston, MA, USA, Nov. 2012.

[39] G. Governatori, A. Rotolo, S. Villata, and F. Gandon. "One License to Compose Them All. A Deontic Logic Approach to Data Licensing on the Web of Data". In: *International Semantic Web Conference (ISWC)*. Sydney, NSW, Australia, Oct. 2013.

[40] G. Graefe. "Query Evaluation Techniques for Large Databases". In: *ACM Comput. Surv.* 25.2 (1993), pp. 73–170. DOI: 10.1145/152610.152611. URL: http://doi.acm.org/10.1145/152610.152611.

[41] S. Gupta, P. Szekely, C. A. Knoblock, A. Goel, M. Taheriyan, and M. Muslea. "Karma: A System for Mapping Structured Sources Into the Semantic Web". In: *Extended Semantic Web Conference (ESWC), Poster&Demo*. Heraklion, Crete, Greece, May 2012.

[42] F. Hacques, H. Skaf-Molli, P. Molli, and S. E. Hassad. "PFed: Recommending Plausible Federated SPARQL Queries". In: *International Conference on Database and Expert Systems Applications (DEXA)*. Springer. Linz, Austria, Aug. 2019.

[43] R. Harrison. *TOGAF Version 8.1*. New York: Van Haren Publishing, 2007.

[44] O. Hartig, C. Bizer, and J. C. Freytag. "Executing SPARQL Queries over the Web of Linked Data". In: *International Semantic Web Conference (ISWC)*. Chantilly, VA, USA, Oct. 2009. DOI: 10.1007/978-3-642-04930-9_19. URL: http://dx.doi.org/10.1007/978-3-642-04930-9_19.

[45] P. Heyvaert, B. De Meester, A. Dimou, and R. Verborgh. "Declarative Rules for Linked Data Generation at Your Fingertips!" In: *Extended Semantic Web Conference (ESWC), Poster&Demo*. Heraklion, Crete, Greece, June 2018.

[46] P. Heyvaert, A. Dimou, A.-L. Herregodts, R. Verborgh, D. Schuurman, E. Mannens, and R. Van de Walle. "RMLEditor: a Graph-Based Mapping Editor for Linked Data Mappings". In: *Extended Semantic Web Conference (ESWC)*. Heraklion, Crete, Greece, May 2016.

[47] A. Hogan, M. Mellotte, G. Powell, and D. Stampouli. "Towards Fuzzy Query-Relaxation for RDF". In: *Extended Semantic Web Conference (ESWC)*. Heraklion, Crete, Greece, May 2012.

[48] F. Howell and R. McNab. "Simjava: a Discrete Event Simulation Library for Java". In: *International Conference on Web-Based Modeling and Simulation*. San Diego, CA, USA, Jan. 1998.

[49] H. Huang, C. Liu, and X. Zhou. "Approximating Query Answering on RDF Databases". In: *Journal of World Wide Web* 15 (2012).

[50] C. A. Hurtado, A. Poulovassilis, and P. T. Wood. "Query Relaxation in RDF". In: *Journal on Data Semantics X* (2008).

[51] M. Jawad. "Data Privacy in P2P Systems". Theses. Université de Nantes, June 2011. URL: https://tel.archives-ouvertes.fr/tel-00638721.

[52] M. Jawad, P. Serrano-Alvarado, and P. Valduriez. "Design of PriServ, A Privacy Service for DHTs". In: *International Workshop on Privacy and Anonymity in the Information Society (PAIS)*. Nantes, France, Mar. 2008, pp. 21–25. URL: https://hal.inria.fr/inria-00374320.

[53] M. Jawad, P. Serrano-Alvarado, and P. Valduriez. "Protecting Data Privacy in Structured P2P Networks". In: *Second International Conference on Data Management in Grid and P2P Systems (Globe)*. Vol. 5697. LNCS. Linz, Austria: Springer, Sept. 2009, pp. 85–98. URL: https://hal.inria.fr/inria-00414050.

[54] M. Jawad, P. Serrano-Alvarado, and P. Valduriez. "Supporting Data Privacy in P2P Systems". In: *Security and Privacy Preserving in Social Networks*. Ed. by R. Chbeir and B. A. Bouna. Lecture Notes in Social Networks. Chapter of 50 pages. Springer, Aug. 2013. ISBN: 978-3-7091-0893-2. URL: https://hal.archives-ouvertes.fr/hal-00807625.

[55] M. Jawad, P. Serrano-Alvarado, P. Valduriez, and S. Drapeau. "A Data Privacy Service for Structured P2P Systems". In: *Encuentro Nacional de Computación (ENC)*. Mexico, Mexico, Sept. 2009. URL: https://hal.archives-ouvertes.fr/hal-00419623.

[56] M. Jawad, P. Serrano-Alvarado, P. Valduriez, and S. Drapeau. "Ensuring Data Privacy in Structured P2P systems with PriServ". In: *Bases de Données Avancées (BDA)*. Namur, Belgium, Oct. 2009.

[57] M. Jawad, P. Serrano-Alvarado, P. Valduriez, and S. Drapeau. "Privacy Support for Sensitive Data Sharing in P2P Systems". In: *Bases de Données Avancées (BDA)*. Demonstration paper. Rabat, Morocco, Oct. 2011. URL: https://hal.archives-ouvertes.fr/hal-01153238.

[58] M. Jelasity, S. Voulgaris, R. Guerraoui, A.-M. Kermarrec, and M. van Steen. "Gossip-based Peer Sampling". In: *ACM Transactions on Computer Systems (TOCS)* 25.3 (Aug. 2007). ISSN: 0734-2071. DOI: 10.1145/1275517.1275520. URL: http://doi.acm.org/10.1145/1275517.1275520.

[59] A. Jøsang. "A Logic for Uncertain Probabilities". In: *Uncertainty, Fuzziness and Knowledge-Based Systems* 9.3 (2001), pp. 279–311.

[60] A. Jøsang and T. Bhuiyan. "Optimal Trust Network Analysis with Subjective Logic". In: *Proceeding of the 2nd International Conference on Emerging Security Information, Systems and Technologies (SECURWARE)*. 2008, pp. 179–184.

[61] A. Jøsang, R. Hayward, and S. Pope. "Trust Network Analysis with Subjective Logic". In: *Proceedings of the 29th Australasian Computer Science Conference (ACSC)*. 2006, pp. 85–94.

[62] A. Jøsang, R. Ismail, and C. Boyd. "A Survey of Trust and Reputation Systems for Online Service Provision". In: *Decision Support Systems* 43.2 (2007), pp. 618–644.

[63] A. Josey. *TOGAF Version 9: A Pocket Guide*. 2nd ed. Van Haren Publishing, 2009.

[64] A. C. Junior, C. Debruyne, and D. O'Sullivan. "An Editor that Uses a Block Metaphor for Representing Semantic Mappings in Linked Data". In: *Extended Semantic Web Conference (ESWC), Poster&Demo*. Heraklion, Crete, Greece, June 2018.

[65] G. M. Kapitsaki, F. Kramer, and N. D. Tselikas. "Automating the License Compatibility Process in Open Source Software With SPDX". In: *Journal of Systems and Software* 131 (2017).

[66] G. Karjoth, M. Schunter, and M. Waidner. "Platform for Enterprise Privacy Practices: Privacy-Enabled Management of Customer Data". In: *Workshop on Privacy Enhancing Technologies*. San Francisco, CA, USA, Apr. 2002.

[67] J. Kleinberg, C. H. Papadimitriou, and P. Raghavan. "On the Value of Private Information". In: *Theoretical Aspects of Rationality and Knowledge (TARK)*. Siena, Italy, June 2001.

[68] D. Kossmann. "The state of the art in distributed query processing". In: *ACM Computing Surveys (CSUR)* 32.4 (2000), pp. 422–469.

[69] J. Kubiatowicz, D. Bindel, Y. Chen, S. E. Czerwinski, P. R. Eaton, D. Geels, R. Gummadi, S. C. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Y. Zhao. "OceanStore: An Architecture for Global-Scale Persistent Storage." In: *Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. Cambridge, MA, USA, Nov. 2000.

[70] M. Langheinrich. *A P3P Preference Exchange Language (APPEL1.0) Specification*. 2001.

[71] K. LeFevre, R. Agrawal, V. Ercegovac, R. Ramakrishnan, Y. Xu, and D. J. DeWitt. "Limiting Disclosure in Hippocratic Databases." In: *Very Large Databases (VLDB)*. Toronto, Canada, Sept. 2004.

[72] M. Lefrançois, A. Zimmermann, and N. Bakerally. "A SPARQL Extension For Generating RDF From Heterogeneous Formats". In: *Extended Semantic Web Conference (ESWC)*. Portorož, Slovenia, May 2017.

[73] L. Li and Y. Wang. "A Subjective Probability Based Deductive Approach to Global Trust Evaluation in Composite Services". In: *Proceedings of the 9th IEEE International Conference on Web Services (ICWS)*. 2011, pp. 604–611. ISBN: 978-0-7695-4463-2. DOI: 10.1109/ICWS.2011.28. URL: http://dx.doi.org/10.1109/ICWS.2011.28.

[74] L. Li and Y. Wang. "Subjective Trust Inference in Composite Services". In: *Proceedings of the 24th Conference on Artificial Intelligence (AAAI)*. July 2010.

[75] *Liberty Alliance Project, Privacy Preference Expression Languages (PPELs). http://projectliberty.o file/Final_PPEL_White_Paper.pdf.*

[76] G. Liu, Y. Wang, M. Orgun, and E. Lim. "Finding the Optimal Social Trust Path for the Selection of Trustworthy Service Providers in Complex Social Networks". In: *IEEE Transactions on Services Computing* 6.2 (2011), pp. 152–167. ISSN: 1939-1374. DOI: 10.1109/TSC.2011.58.

[77] J. Lorey and F. Naumann. "Detecting SPARQL Query Templates for Data Prefetching". In: *ESWC Conference*. 2013. DOI: 10.1007/978-3-642-38288-8_9. URL: http://dx.doi.org/10.1007/978-3-642-38288-8_9.

[78] S. P. Marsh. "Formalising Trust as a Computational Concept". PhD thesis. Department of Mathematics and Computer Science, University of Stirling, 1994.

[79] D. H. Mcknight and N. L. Chervany. *The Meanings of Trust*. Tech. rep. University of Minnesota, Carlson School of Management, 1996.

[80] F. Michel, C. Faron Zucker, and J. Montagnat. "A Mapping-based Method to Query MongoDB Documents with SPARQL". In: *International Conference on Database and Expert Systems Applications (DEXA 2016)*. Porto, Portugal, Sept. 2016. URL: https://hal.archives-ouvertes.fr/hal-01330146.

[81] P. Mishra and M. H. Eich. "Join Processing in Relational Databases". In: *ACM Comput. Surv.* 24.1 (1992), pp. 63–113.

[82] K. Möller, M. Hausenblas, R. Cyganiak, G. Grimnes, and S. Handschuh. "Learning from Linked Open Data Usage: Patterns & Metrics". In: *WebSci10: Extending the Frontiers of Society On-Line*. 2010.

[83] B. Moreau, P. Serrano-Alvarado, and E. Desmontils. "CaLi: A Lattice-Based Model for License Classifications". In: *Bases de Données Avancées (BDA)*. Bucharest, Romania, Oct. 2018. URL: https://hal.archives-ouvertes.fr/hal-01816451/.

[84] B. Moreau, P. Serrano Alvarado, E. Desmontils, and D. Thoumas. "Querying non-RDF Datasets using Triple Patterns". In: *16th International Semantic Web Conference (ISWC)*. Demo paper. Vienna, Austria: Springer, Oct. 2017. URL: https://hal.archives-ouvertes.fr/hal-01583518.

[85] B. Moreau, P. Serrano-Alvarado, M. Perrin, and E. Desmontils. "A License-Based Search Engine". In: *16th Extended Semantic Web Conference (ESWC)*. Demo paper. Portoroz, Slovenia, June 2019. URL: https://hal.archives-ouvertes.fr/hal-02097027.

[86] B. Moreau, P. Serrano-Alvarado, M. Perrin, and E. Desmontils. "Modelling the Compatibility of Licenses". In: *16th Extended Semantic Web Conference (ESWC)*. Portorož, Slovenia, June 2019. URL: https://hal.archives-ouvertes.fr/hal-02069076.

[87] M. Morsey, J. Lehmann, S. Auer, and A.-C. N. Ngomo. "DBpedia SPARQL Benchmark–Performance Assessment with Real Queries on Real Data". In: *International semantic web conference (ISWC)*. Springer. Bonn, Germany, Oct. 2011.

[88] F. Moyano, M. C. Fernández-Gago, and J. Lopez. "A Conceptual Framework for Trust Models". In: *Proceedings of the 9th International Conference on Trust, Privacy and Security in Digital Business (TrustBus)*. 2012, pp. 93–104.

[89] S. U. Nabar, B. Marthi, K. Kenthapadi, N. Mishra, and R. Motwani. "Towards Robustness in Query Auditing". In: *Vary Large Databases (VLDB)*. Seoul, Korea, Sept. 2006.

[90] G. Nassopoulos. "Deducing Basic Graph Patterns from Logs of Linked Data Providers". Theses. Universite de Nantes, May 2017. URL: https://hal.archives-ouvertes.fr/tel-01536912.

[91] G. Nassopoulos, P. Serrano-Alvarado, P. Molli, and E. Desmontils. "FETA: Federated QuEry TrAcking for Linked Data". In: *27th International Conference on Database and Expert Systems Applications (DEXA)*. Vol. 9827. LNCS. Short paper, CORE B. Porto, Portugal: Springer, Sept. 2016. URL: https://hal.archives-ouvertes.fr/hal-01336386.

[92] G. Nassopoulos, P. Serrano-Alvarado, P. Molli, and E. Desmontils. *Extracting Basic Graph Patterns from Triple Pattern Fragment Logs*. Research Report. LS2N-University of Nantes, 2017. URL: https://arxiv.org/abs/1906.08574.

[93] G. Nassopoulos, P. Serrano-Alvarado, P. Molli, and E. Desmontils. *Tracking Federated Queries in the Linked Data*. Research Report. LINA-University of Nantes, Aug. 2015. URL: https://arxiv.org/abs/1508.06098.

[94] M. T. Ozsu and P. Valduriez. *Principles of Distributed Database Systems*. 3rd. Springer, 2011.

[95] M. T. Özsu and P. Valduriez. *Principles of Distributed Database Systems, Third Edition*. Springer, 2011. ISBN: 978-1-4419-8833-1. DOI: 10.1007/978-1-4419-8834-8. URL: http://dx.doi.org/10.1007/978-1-4419-8834-8.

[96] J. Pérez, M. Arenas, and C. Gutierrez. "Semantics and Complexity of SPARQL". In: *ACM Transactions on Database Systems (TODS)* 34.3 (2009). DOI: 10.1145/1567274.1567278. URL: http://doi.acm.org/10.1145/1567274.1567278.

[97] F. Picalausa and S. Vansummeren. "What are Real SPARQL Queries Like?" In: *SWIM Workshop*. 2011.

[98] J.-A. Quiané-Ruiz, P. Lamarre, and P. Valduriez. "A Self-Adaptable Query Allocation Framework for Distributed Information Systems". In: *International Journal on Very Large Data Bases (VLDBJ)* 18.3 (2009), pp. 649–674.

[99] B. Quilitz and U. Leser. "Querying Distributed RDF Data Sources with SPARQL". In: *European Semantic Web Conference (ESWC)*. Tenerife, Canary Islands, Spain, June 2008. ISBN: 3-540-68233-3, 978-3-540-68233-2. URL: http://dl.acm.org/citation.cfm?id=1789394.1789443.

[100] A. Raghuveer. "Characterizing Machine Agent Behavior through SPARQL Query Mining". In: *USEWOD Workshop*. 2012.

[101] N. A. Rakhmawati, M. Saleem, S. Lalithsena, and S. Decker. "Qfed: Query Set for Federated SPARQL Query Benchmark". In: *International Conference on Information Integration and Web-based Applications & Services*. ACM. 2014.

[102] M. Richardson, R. Agrawal, and P. Domingos. "Trust Management for the Semantic Web". In: *International Semantic Web Conference (ISWC)*. Sanibel Island, Florida, USA, Oct. 2003.

[103] L. Rietveld, R. Hoekstra, et al. "Man vs. Machine: Differences in SPARQL queries". In: *USEWOD Workshop*. 2014.

[104] S. Rouibia, M. Ghareed, B. Parrein, M. Biazzini, R. Carvajal-Gomez, A. Perez-Espinosa, and P. Serrano-Alvarado. "Towards a Hybrid Client/Server and P2P Architecture for Content Delivery over the Internet". In: *CFIP/NOTERE*. Bayonne, France, Oct. 2012. URL: https://hal.archives-ouvertes.fr/hal-00761483.

[105] A. Rowstron and G. House. "Storage Management and Caching in PAST, a Large-scale, Persistent Peer-to-Peer Storage Utility". In: *Symposium on Operating Systems Principles (SOSP)*. Banff, Alberta, Canada, Oct. 2001.

[106] M. Schmidt, O. Görlitz, P. Haase, G. Ladwig, A. Schwarte, and T. Tran. "FedBench: A Benchmark Suite for Federated Semantic Data Query Processing". In: *International Semantic Web Conference (ISWC)*. Bonn, Germany, Oct. 2011. URL: http://dx.doi.org/10.1007/978-3-642-25073-6_37.

[107] A. Schwarte, P. Haase, K. Hose, R. Schenkel, and M. Schmidt. "FedX: Optimization Techniques for Federated Query Processing on Linked Data". In: *International Semantic Web Conference (ISWC)*. Bonn, Germany, Oct. 2011. DOI: 10.1007/978-3-642-25073-6_38.

[108] O. Seneviratne, L. Kagal, and T. Berners-Lee. "Policy-Aware Content Reuse on the Web". In: *International Semantic Web Conference (ISWC)*. Chantilly, VA, USA, Oct. 2009.

[109] V. Soto-Mendoza, J. A. García-Macías, E. Chávez, A. I. Martínez-García, J. Favela, P. Serrano-Alvarado, and M. R. R. Zúñiga. "Design of a Predictive Scheduling System to Improve Assisted Living Services for Elders". In: *Transactions on Intelligent Systems and Technology (TIST)* 6.4 (2015).

[110] V. Soto-Mendoza, P. Serrano-Alvarado, E. Desmontils, and J. A. Garcia-Macias. "Policies Composition Based on Data Usage Context". In: *6th International Workshop on Consuming Linked Data (COLD) at ISWC*. Bethlehem, United States, Oct. 2015. URL: https://hal.archives-ouvertes.fr/hal-01184660.

[111] I. Stoica, R. Morris, D. R. Karger, M. F. Kaashoek, and H. Balakrishnan. "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications." In: *ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)*. San Diego, CA, USA, Aug. 2001.

[112] The official OBASHI Website. *http://www.obashi.co.uk/*. Last accessed May 2015.

[113] R. Verborgh, E. Mannens, and R. Van de Walle. "Initial Usage Analysis of DBpedia's Triple Pattern Fragments". In: *USEWOD Workshop*. 2015. URL: http://linkeddatafragments.org/publications/usewod2015.pdf.

[114] R. Verborgh, M. Vander Sande, O. Hartig, J. Van Herwegen, L. De Vocht, B. De Meester, G. Haesendonck, and P. Colpaert. "Triple Pattern Fragments: A Low-Cost Knowledge Graph Interface for the Web". In: *Journal of Web Semantics* 37 (2016).

[115] L. Viljanen. "Towards an Ontology of Trust". In: *Proceedings of the 2nd International Conference on Trust, Privacy and Security in Digital Business (TrustBus)*. 2005.

[116] S. Villata and F. Gandon. "Licenses Compatibility and Composition in the Web of Data". In: *Workshop Consuming Linked Data (COLD) collocated with ISWC*. Boston, MA, USA, Dec. 2012.

[117] A. N. Wilschut and P. M. Apers. "Dataflow query execution in a parallel main-memory environment". In: *International Conference on Parallel and Distributed Information Systems*. IEEE. 1991, pp. 68–77.

[118] Z. Yan and S. Holtmanns. "Computer Security, Privacy and Politics: Current Issues, Challenges and Solutions". In: IGI Global, 2007. Chap. Trust Modeling and Management: from Social Trust to Digital Trust.

[119] A. Zaveri, A. Rula, A. Maurino, R. Pietrobon, J. Lehmann, and S. Auer. "Quality Assessment for Linked Data: A Survey". In: *Semantic Web* 7.1 (2016), pp. 63–93.

[120] P. Zhang, A. Durresi, and L. Barolli. "Survey of Trust Management on Various Networks". In: *Proceedings of the 5th International Conference on Complex, Intelligent and Software Intensive Systems (CISIS)*. 2011, pp. 219–226. DOI: 10.1109/CISIS.2011.122.

UNIVERSITE BRETAGNE LOIRE / MATHSTIC

UNIVERSITÉ DE NANTES

Sur la protection des données utilisateur dans les systèmes répartis

Mots-clés : Confidentialité des données, licences, confiance, systèmes pair-à-pair, Web des données, traitement des requêtes fédérées, contrôle d'usage.

Résumé : Protéger les données des utilisateurs dans les systèmes distribués est aujourd'hui très difficile. Dans cette thèse, nous nous concentrons sur diverses questions liées à la protection des données utilisateur sur des systèmes de gestion de données distribuées, dont les architectures vont des architectures client-serveur aux fédérations de serveurs ou aux énormes organisations pair-à-pair. Notre première contribution concerne la confiance dans un système. Nous proposons un métamodèle basé sur la logique du premier ordre, qui permet de modéliser un système en considérant des entités des mondes sociaux et numériques et leurs relations. Nous proposons ensuite deux approches permettant aux utilisateurs d'évaluer la confiance envers les systèmes. Dans le contexte des systèmes pair-à-pair, nous proposons un modèle de confidentialité de données et son implémentation basée sur DHT, ainsi qu'un canevas permettant de mesurer et d'améliorer la satisfaction des utilisateurs. Dans le contexte du Web sémantique, en particulier des données liées, nous proposons deux approches pour déduire des BGP de requêtes SPARQL. Enfin, nous proposons un modèle basé sur treillis qui permet de positionner des licences en termes de compatibilité et de conformité.

Protecting user data in distributed systems

Keywords : Data privacy, licenses, trust, peer-to-peer systems, Linked Data, federated query processing, usage control.

Abstract: Protecting user data in distributed systems nowadays is very difficult. In this thesis, we focus on various issues related to the protection of user data on distributed data management systems, whose architectures range from client-server architectures to federations of servers or huge peer-to-peer organizations. Our first contribution is about trust in a system. We propose a metamodel based on first order logic, that allows to model a system considering entities of the social and digital worlds and their relations. Then we propose two approaches that allow users to evaluate trust in systems. In the context of peer-to-peer systems, we propose a data privacy model, and its DHT-based implementation, as well as a framework to measure and improve users' satisfaction. In the context of the semantic web, in particular the Linked Data, we propose two approaches for deducing BGPs of SPARQL queries. Finally, we propose a lattice-based model that allows to position licenses in terms of compatibility and compliance.