EMERGENT SEMANTICS: RETHINKING INTEROPERABILITY FOR LARGE SCALE DECENTRALIZED INFORMATION SYSTEMS

THÈSE N° 3690 (2006)

PRÉSENTÉE A LA FACULTÉ DE INFORMATIQUE ET COMMUNICATION

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

POUR L'OBTENTION DU GRADE DE DOCTEUR ÈS SCIENCES

PAR

Philippe CUDRÉ-MAUROUX

Ingénieur diplômé en Systèmes de Communications,
EPFL, Lausanne, Suisse

de nationalité suisse

acceptée sur proposition du jury:

Prof. Karl ABERER (EPFL), directeur de thése
Dr. Avigdor GAL (Technion), rapporteur
Prof. Monika HENZINGER (EPFL & Google Inc.), rapporteur
Prof. Zachary G. IVES (U. of Pennsylvania), rapporteur

Lausanne, EPFL
2006

Résumé

Les problèmes liés à l'interopérabilité sémantique des systèmes d'information furent jusqu'à présent résolus par le biais de solutions centralisées, tant au niveau systémique qu'au niveau logique. Cette approche, couronnée d'un certain succès pour les environnements statiques, offre cependant une extensibilité et une flexibilité limitées. De nouvelles architectures distribuées – tels les systèmes Pair-à-Pair (P2P) – ont récemment encouragé l'application des principes de décentralisation et d'auto-organisation. Ces architectures ont ainsi offert des solutions novatrices aux nombreux problèmes liés à la dimension et à la dynamique des systèmes d'information actuels. Notre recherche propose de résoudre les problèmes d'interopérabilité en se basant sur des interactions décentralisées et auto-organisationnelles.

La première partie de cette thèse présente les techniques d'intégration de données traditionnelles se basant sur des schémas de données globaux, des intégrations de données parfaites et des reformulations de requêtes bornées. L'écologie actuelle du Web est constituée de sources de données autonomes et dynamiques. Dans un tel environnement, les données, les schémas des données et les schémas d'intégration des données peuvent tous être générés de manière indépendante et automatisée. Les architectures d'intégration traditionnelles, centralisées, statiques et hiérarchiquement descendantes, sont ainsi clairement inapplicables aux nouveaux systèmes d'informations répartis. Nous proposons une évolution de ces architectures basée sur une intégration décentralisée et dynamique.

Dans la deuxième partie de cette thèse, nous proposons un ensemble de principes favorisant l'interopérabilité des systèmes d'information répartis. Nous commençons par introduire de nouvelles métriques en vue de qualifier les schémas d'intégration, se basant à la fois sur les pertes syntactiques et sémantiques liées aux reformulations itératives des requêtes. Nous détaillons des méthodes analytiques pour évaluer nos métriques, et montrons comment tirer avantage de nos techniques pour graduellement corriger les relations sémantiques inconsistantes à travers un réseau. Nous décrivons un mécanisme d'inférence décentralisée portant sur la transitivité des opérations d'intégration pour évaluer le degré d'hétérogénéité sémantique entre paires de systèmes d'information. Enfin, nous proposons une analyse théorique du graphe sémantique sous-jacent

au système d'information afin de quantifier la qualité de l'intégration globale pouvant être ainsi constituée.

La troisième et dernière partie de cette thèse est dédiée à la présentation de deux systèmes illustrant l'application pratique de nos idées. Le premier système, GridVine, est un réseau sémantique logique supportant une intégration de données décentralisée basée sur des schémas d'intégration Pair-à-Pair et sur un héritage monotone des schémas de données. GridVine est bâti suivant le principe d'indépendance des données et dissocie la couche logique, formée par le réseau sémantique intégrant les données, de la couche physique constituée d'un réseau Pair-à-Pair structuré utilisé pour un routage efficace des messages. Le second système, appelé PicShark, tire avantage de données semi-structurées pour partager des images annotées dans un environnement collaboratif. PicShark utilise nos principes pour créer dynamiquement des annotations et des schémas d'intégration sémantique, et pour graduellement limiter l'entropie du système global – en terme de méta-données manquantes et d'hétérogénéité schématique – afin de recontextualiser les données d'une manière décentralisée et auto-organisationnelle.

Tout au long de cette thèse, nous préconisons une définition holistique de la sémantique liée aux systèmes d'information répartis. Nous modélisons la sémantique globale comme émergent d'une multitude d'interactions locales et répétées provenant de systèmes hétérogènes. Nous considérons tant la représentation de la sémantique que la découverte de l'interprétation des symboles du système comme le produit d'un processus d'auto-organisation conduit par une collection d'agents dont la fonction d'utilité dépend directement de l'interprétation correcte des symboles. Notre vision contraste singulièrement avec les contributions précédentes analysant les sources d'informations de manière isolées ou se concentrant sur des schémas de données globaux et statiques. Dans un monde où l'information digitale est abondante mais où l'attention humaine reste limitée, nous pensons que des approches dynamiques et décentralisées telles que celles proposées dans cette thèse auront une part toujours plus importante dans la gestion des flots massifs de données hétérogènes, dynamiques et distribuées traversant les réseaux d'information globaux actuels.

Mots-clés: bases de données hétérogènes, interopérabilité sémantique, gestion des données pair-à-pair.

Abstract

In the past, the problem of semantic interoperability in information systems was mostly solved by means of centralization, both at a system and at a logical level. This approach has been successful to a certain extent, but offers limited scalability and flexibility. Peer-to-Peer systems as a new brand of system architectures indicate that the principles of decentralization and self-organization might offer new solutions to many problems that scale well to very large numbers of users, or to systems where central authorities do not prevail. Therefore, we suggest a new way of building global agreements, i.e., semantic interoperability, based on decentralized, self-organizing interactions only.

In the first part of this thesis, we discuss traditional data integration techniques relying on global schemas, perfect schema mappings and contained query rewritings. We elaborate on the current ecology of the World Wide Web, where autonomous information sources come and go in dynamic and unpredictable ways. In the current environment, data, schemas and schema mappings can all be generated without human intervention and get encoded in syntactic structures with limited expressivity. We argue that traditional top-down integration techniques are inapplicable to that new context and propose a new integration architecture based on decentralized mappings and dynamic self-organization.

In the second part of this thesis, we propose a set of principles to foster semantic interoperability in very large scale information systems. We start by introducing new metrics for the schema mappings, based on both syntactic losses (completeness) and semantic mismatch (soundness) to selectively reformulate queries in a decentralized network of heterogeneous parties. We detail analytical methods to evaluate our metrics, and show how to take advantage of those methods to gradually alleviate mapping inconsistencies across the network. We describe a totally decentralized message passing scheme using belief propagation on transitive closures of schema mapping operations to efficiently evaluate the degree of semantic mismatch between pairs of acquainted information systems. Finally, we propose a graph-theoretic analysis of the network of mappings to quantify the quality of the global agreement that can be achieved in that way.

The third and last part of this thesis is devoted to the presentation of two systems illustrating the practical applicability of our ideas. The first

system we introduce, GridVine, is a Semantic Overlay Network supporting decentralized data integration techniques through pairwise schema mappings and monotonic schema inheritance. GridVine follows the principle of data independence by separating a logical layer, the semantic overlay for managing and mapping data and schemas, from a physical layer consisting of a self-organizing Peer-to-Peer overlay network for efficient routing of messages. The second system, called PicShark, takes advantage of semi-structured metadata to meaningfully share pictures in collaborative settings. PicShark builds on our principles to dynamically create both annotations and mappings, and to gradually minimize information entropy – in terms of missing metadata and schematic heterogeneity – in a self-organizing and decentralized context.

Throughout this thesis, we advocate a holistic view on semantics in large-scale information systems: we model semantics as bottom-up and dynamic agreements among heterogeneous parties. We consider both the representation of semantics and the discovery of the interpretation of symbols as the result of a self-organizing process performed by distributed agents whose utility functions depend on the proper interpretation of the symbols. Our view sharply contrasts with previous top-down contributions analyzing data sources in isolation or focusing on global vocabularies and rigid sets of interpretations curated off-line. In a world where digital information is abundant but human attention remains scarce, we believe that autonomous, best-effort processes such as the ones proposed throughout this thesis will play an ever increasing role in complementing traditional top-down integration approaches to handle massive amounts of digitalized and heterogeneous information assets.

Keywords: heterogeneous databases, semantic interoperability, peer data management.

Acknowledgements

I want to thank first my advisor, Karl Aberer, for bringing me to this exciting research topic and for guiding me throughout my PhD. I will be eternally grateful to him for having so strongly believed in me from day one, and for having provided me with invaluable insight, not only in research, but also in many other aspects of life in general.

I am also very grateful to the whole LSIR lab; quite frankly, I could not have dreamt of a better team of co-workers. Thanks to Chantal Menghini for unparalleled logistical support. Special thanks to Manfred Hauswirth and Anwitaman Datta for their delightful company throughout my PhD. I also want to thank the students who helped me in the implementation of some the systems described hereafter: Julien Gaugaz for most of the graph-theoretic experiments, Andras Feher for the EON ontology alignment tool, and Tim van Pelt for the first version of GridVine.

I wish to express my gratitude to all the fine scientists who helped me over the last few years: the EPFL-HP group, more specifically Pascal Frossard and Eric Debes, for all their pieces of advice. Olivier Verscheure and Lisa Amini for six exciting months spent in New York working for IBM Research. Wei-Ying Ma for having invited me to Beijing on behalf of Microsoft Research Asia and the MSRA Refugees for a splendid summer. Special thanks to Mike Franklin and the whole Berkeley DB group for their warm welcome and for passionate research discussions.

On a more personal side, I feel greatly indebted to my exemplary friends from Bulle, to the whole Bout du Monde crew in Vevey and to my excellent EPFL friends from Lausanne. I don't know half of you half as well as I should like, and I like less than half of you half as well as you deserve.

Thanks to Jacques Pilet for his Search of Lost Time and to Jean-Pierre Fragnière for his Aristotelian lessons.

Finally, my gratitude goes to my family for love, support and encouragement during all my studies.

Contents

The three most important problems in Databases used to be Performance, Performance and Performance; in the future, the three most important problems will be Semantics, Semantics, and Semantics... (paraphrase) Stefano Ceri, 1998.

When you and I speak or write to each other, the most we can hope for is a sort of incremental approach toward agreement, toward communication, toward common usage of terms.
Douglas Lenat, 1995.

Rose is a rose is a rose is a rose. Gertrude Stein, 1913.

Chapter 1

Introduction

The recent success of Peer-to-Peer (P2P) systems and the initiatives to create a Semantic Web have emphasized once more a key problem in information systems: the lack of semantic interoperability. Semantic interoperability is a crucial element for making distributed information systems usable. It is a prerequisite for structured, distributed search and data exchange, and provides the foundations for higher level (Web) services and processing.

For instance, the technologies that are currently in place for P2P file sharing systems either impose a simple semantic structure *a priori* (e.g., on P2P systems such as Kazaa[1] or eMule[2]) and leave the burden of semantic annotation to the user, or do not address the issue of semantics at all (e.g., on the current Web or on systems like Freenet[3]). In the latter case, the systems only support unstructured data representation and leave the burden of *making sense* to the skills of the user, e.g., by creating pseudo-structured strings such as *Report-2x2006-P2P-Project-Version-2.3* to encode very simple semantics in the identifiers of the shared files.

Classical attempts to make information resources interoperable, in particular in the domain of database integration, were developed for relatively small sets of curated, immutable and highly structured data sources. The wrapper-mediator or ontology-based information integration approaches rely on global vocabularies and contained rewritings to provide certain answers to queries issued locally. Even if those approaches are arguably quite appropriate in confined, static environments, they require too much human attention and coordination to be enforced in the large.

With the evolution of global networking infrastructures and the democratization of tools facilitating the elicitation of knowledge in machine-processable formats, the problem of data integration reaches a whole new dimension today. Computers manipulate data originating from very dis-

[1] *http://www.kazaa.com/*
[2] *http://www.emule-project.net/*
[3] *http://freenetproject.org/*

parate sources – e.g., on the Semantic Web, where all resources are globally accessible and identifiable by URIs. Both systems and end-users are gradually gaining more autonomy: systems through automatic creation and manipulation of data (e.g., in sensor networks or self-maintaining community websites), and users through the availability of software allowing the creation, transformation and individualization of personal data. The result is an ocean of information – ranging from raw experimental data to semi-structured text or tagged video sequences – available on the Internet but mostly impossible to discover or manipulate at run-time due to their lack of semantics.

Throughout this thesis, we propose a new way of conceptualizing semantics. We consider global interoperability as emerging from collections of dynamic agreements between heterogeneous parties. We introduce a different view on the problem of semantic heterogeneity by taking a social, holistic perspective relying on self-organization and repeated local interactions among communities of autonomous agents. We argue that we can see independent and unsupervised interactions as an opportunity to improve semantic interoperability rather than as a threat, in particular in revealing new possibilities on how semantic agreements can be achieved. *Folksonomies*, i.e., collaborative labeling schemes used to classify resources on Web sites such as Flickr[4] or Del.icio.us[5], are simple examples of emergent semantics phenomena applied to unstructured and centralized information settings. In this thesis, we focus on semantics emerging from (semi) structured, large scale and decentralized settings.

1.1 On Syntax, Semantics and Syntactic Semantics

Syntax is classically considered as the study of the rules of *symbol* formation and manipulation [Mor38]. In the context of this thesis, we mostly associate syntax to the formalisms used to create, interrelate and query concepts – such as relational attributes or ontological classes. Note that a common syntax (e.g., XML) is often a prerequisite for establishing semantic interoperability, but is in general insufficient by itself as semantic equivalence (e.g., between *book* and *livre*) does not generally imply equivalence on the syntactic level.

Despite its wide usage in many contexts, the notion of *semantics* often lacks a precise definition. As a least common denominator, we can characterize semantics as a relationship or mapping established between a syntactic structure and some domain. The syntactic structure is a set of symbols that can be combined following syntactic rules. The possible domains relating the symbols through semantics can vary widely.

[4] *http://www.flickr.com/*
[5] *http://del.icio.us/*

In linguistics, this domain is often considered as a domain of conceptual interpretations. In mathematical logic, a semantic interpretation for a formal language is specified by defining mappings from the syntactic constructs of the language to an appropriate mathematical model. Denotational semantics applies this idea to programming languages. Natural language semantics classically concerns a triadic structure comprising a *symbol* (how some idea is expressed), an *idea* (what is abstracted from reality) and a *referent* (the particular object in reality) [OR23].

By considering the union of the syntactic and semantic domains, however, semantics can be regarded as syntax, i.e., semantics can be turned into a study of relations within a single domain among the symbols and their interpretations [Rap03]. Dictionaries are simple examples of constructs based on that paradigm, where the interpretations of symbols (i.e., words) are given by means of the same symbols, creating a closed correspondence continuum. *Syntactic semantics* is the name given to the study of semantics through the analysis of syntactic constructs.

Beyond its implication in linguistics – where it is conjectured that human beings inevitably understand meaning in terms of some syntactic domain – we consider syntactic semantics as mostly relevant to the computer science domain. Programs, database schemas, models, or ontologies are unconscious artifacts and have no capacity (yet?) to refer to reality. However, software agents have various mechanisms at their disposal for establishing relationships between internal symbols and external artifacts. In the setting where humans provide semantics, relationships among symbols – such as constraints in relational databases – are means to express semantics. In order to rectify some of the problems related to the implicit representation of semantics relying on human cognition, some have proposed the use of an explicit reference system for relating sets of symbols in a software system. Ontologies serve this purpose: an ontology vocabulary consists in principle of formal, explicit but partial definitions of the intended meaning of a domain of discourse [Gru93, Gua98]. In addition, formal constraints (e.g., on the mandatoriness or cardinality of relationships between concepts) are added to reduce the fuzziness of the informal definitions. Specific formal languages (e.g., OWL) allow to define complex notions and support inferencing capabilities (generative capacity). In that way, explicitly represented semantics of syntactic structures in an information system consist of relationships between those syntactic structures and some generally agreed-upon syntactic structure. Thus, the semantics is itself represented by a syntactic structure.

1.2 Emergent Semantics in Distributed Information Systems

In a distributed environment of information agents, such as in the Semantic Web or Peer-to-Peer systems, the aim is to have the agents interoperate irrespective of the source of their initial semantics. To that aim, an agent has to map its vocabulary (carrying the meaning as initially defined in its *base* schema or ontology) to the vocabulary of other agents with which it wants to interoperate. In that way, a relationship between local and distant symbols is established. This relationship may be considered as another form of semantics, independent of the initial semantics of the symbols.

Assuming that autonomous software agents have acquired their semantics through relationships to other agents and that agents interact without human intervention, the original *human assigned* semantics would loose its relevance; from an agent's perspective, *new* semantics would then result from the relationships to its environment. We view this as a novel way of providing semantics to symbols of autonomous agents relative to the symbols of other agents they are interacting with. Typically, this type of semantic representation is distributed such that no agent holds a complete representation of a generally agreed-upon semantics.

With the classical notion of semantics in information systems, the process of generating semantic interpretations, e.g., the generation of ontologies reflecting shared semantics, is somewhat left outside the operation of the information system. The process is assumed to rely on social interactions among humans, possibly supported in their collaborative effort by some computational and communicational tools. Viewing semantics of information agents as a relationship to other agents allows us to internalize the discovery process of those relationships to their operation. We abandon the idea of a preexisting outside agency for forming semantic agreements, but see those as a result of the interaction of autonomous, self-interested agents. This is in line with the concept of expressing semantics through internal relationships in a distributed system. By this approach, we aim at consolidating the local semantics of autonomous information systems into global semantics that result from a continuous interaction of the agents. The structures emerging from the continuous interactions provide meaning to the local symbols. We consider semantics constructed incrementally in that way as *emergent semantics*.

From a global perspective, considering a society of autonomous agents as one system, we observe that the agents form a complex, self-referential, dynamic system. It is well-accepted and known from many examples that such systems (often) result in global states, which cannot be properly characterized at the level of local components. This phenomenon is frequently characterized by the notion of *self-organization*. Thus, emergent semantics is not only a local phenomenon, where agents obtain interpretations

locally through adaptive interactions with other agents, but also a global phenomenon, where a society of agents agree on a common, global state as a representation of the current *semantic agreement* among the agents. This view of semantics as the emergence of a distributed structure from a dynamic process – or more specifically as an equilibrium state of such a process – is in-line with the generally accepted definitions of emergence and emergent structures in the complex systems literature [BY97].

1.3 Scope of Research

In this thesis, we analyze and iteratively refine semantic agreements between information systems acquainted through local schema mappings. We take an emergent, holistic view on the problems of semantics by analyzing transitive closures and composite operations on mapping networks. We exploit interactions between the various agents in the system in order to hypothesize meaning from context and to analyze the degree of acceptance of conventionalized, global semantics.

Our study focuses on systems storing data according to structured and declarative representations such as schemas or ontologies. Viewing schematic elements (e.g., relational attributes, XML elements or RDF classes) as internal or external referents used in schema mappings to relate pieces of information across heterogeneous domains allows us to treat them on a uniform basis. The specific syntactic constructs used to define the schematic elements in a given system are, however, of utmost importance, as they have a direct impact on the ways we can manipulate or detect semantic agreements.

1.4 What the Thesis is not about

As we are interested in structured or semi-structured data stored according to declarative schemas, we do not in the following consider agreements on unstructured text, sets of keywords, or natural language sentences. Also, we concentrate on some of the most basic and universal schematic constructs, such as class definition, extension, equivalence or subsumption, and do not consider more sophisticated declarations (e.g., constraints or type restrictions).

Our analyses assume the existence of schema mappings relating elements from one schema to elements from another schema. However, we are not directly interested in the creation of those mappings (except in Chapter 9, where we propose emergent semantics methods to create the mappings), which might be created manually, semi-automatically or in totally automated manners. Automatic mapping creation is a popular research topic and many approaches have already been proposed (see Chapter 2) to create mappings in dynamic ways.

We take advantage of mappings to reformulate queries iteratively. However, we are not concerned with the complexity of query reformulation or answering directly, as those topics have received significant attention in various recent research efforts (see the following chapter). Finally, we are not interested in the ways agents retrieve or make use of data gathered from other agents, as we mainly focus on efficient ways to propagate queries in order to locate relevant information.

1.5 Outline of the Thesis

This document is divided into three parts. The first part, *Foundations*, introduces the main concepts used throughout the rest of the thesis. *Methods*, the second part, proposes new models and a set of algorithms to foster and analyze semantic interoperability in decentralized settings. The third and last part, *Systems*, describes two systems implementing our approach.

Part I: Foundations

We start with a discussion of traditional data integration techniques in Chapter 2. We give a brief overview of federated databases and wrapper-mediator architectures. We introduce the vision of the Semantic Web and discuss the new interoperability challenges introduced by large-scale, decentralized settings. We then give an overview of P2P architectures in Chapter 3, and explain the reasons why such architectures are particularly attractive to process information in the large. Finally, we introduce the Peer Data Management paradigm and the Semantic Overlay Network architecture to support information interoperability and query processing capabilities in very large scale decentralized settings.

Part II: Methods

Chapter 4 starts with the description of a formal model for Peer Data Management Systems. The rest of the chapter is devoted to Semantic Gossiping, a mechanism to selectively forward queries and iteratively analyze semantic agreements in decentralized networks of heterogeneous parties. We introduce two metrics to quantitatively measure the losses incurred by approximate query reformulations: *syntactic* similarity based on the notion of mapping completeness and *semantic* similarity based on the notion of mapping soundness. The former similarity is derived by analyzing syntactic losses in the reformulated queries, while the latter is obtained by analyzing transitive closures of mapping operations and classification of query results. Chapter 5 extends Semantic Gossiping to automatically correct potentially erroneous mappings and discusses experimental results pertaining to the performance of our approach. We introduce a totally decentralized message passing scheme to detect mapping inconsistencies in

a decentralized but parallel manner in Chapter 6. Our scheme is based on factor-graphs and loopy belief propagation to confront local beliefs on the correctness of the mappings to evidences gathered around the network. We develop our algorithms for both undirected mapping networks and directed mapping networks with inclusive (i.e., subsumption) mappings. Chapter 7, finally, proposes a graph-theoretic analysis of the network of mappings. We derive a necessary condition for obtaining semantic interoperability in the large based on statistical properties of the mapping network. We expand our analysis to derive the extent to which a local query can be propagated throughout the network taking into account both the number and quality of the mappings. We test our heuristics on randomly-generated topologies and on an existing bioinformatic database system.

Part III: Systems

The third part of this thesis starts with a description of GridVine in Chapter 8. GridVine is a Semantic Overlay Network based the P-Grid P2P access structure. Built following the principle of data independence, it separates a logical layer – where data, schemas and schema mappings are managed – from a physical layer responsible for the organization of the peers. GridVine is totally decentralized, yet fosters semantic interoperability through Semantic Gossiping and monotonic schema inheritance. We discuss a reference implementation of GridVine and experiments focusing on various query resolution mechanisms. PicShark, our second system presented in Chapter 9, builds on GridVine to meaningfully share annotated pictures in decentralized settings. We highlight the two main issues preventing structured annotations from being shared in large scale settings: scarcity of annotations and semantic heterogeneity. We propose a formal framework to capture both of those issues and detail mechanisms to alleviate annotation entropy – in terms of missing and heterogeneous annotations – in a community-based and self-organizing way. We describe the architecture of the application and discuss experimental findings relating annotation entropy to the correctness of the annotations generated by the system.

1.6 Contributions

The main contribution of this work is the development of techniques and systems promoting global interoperability in large scale, decentralized settings through self-organizing and local processes. Specific contributions include:

- the introduction of a decentralized, collaborative paradigm focusing on transitive closures of mappings to foster interoperability in decentralized settings

- the description of an architecture for Semantic Overlay Networks, where the physical organization of the machines, the logical organization of the peers and the organization of the semantic mediation layer are all uncorrelated
- the presentation of a formal model for peers, schemas, mappings and queries in a Peer Data Management System
- the introduction of a syntactic similarity measure between iteratively reformulated queries based on the notion of mapping completeness
- the introduction of two semantic similarity measures between iteratively reformulated queries based on the notion of mapping soundness and related to the analyses of cyclic mappings and retrieved results
- the introduction of a tradeoff between the precision and recall of the set of answers to a query in a Peer Data Management System taking advantage of both syntactic and semantic analyses
- the introduction of self-healing semantic networks autonomously repairing potentially erroneous mappings detected through cyclic and result analyses
- the modeling of mapping networks as global and local factor-graphs supporting efficient sum-product operations in a decentralized manner
- the description of a probabilistic message passing scheme to detect mapping inconsistencies in a parallel manner based on loopy belief-propagation
- the modeling of Peer Data Management Systems as collections of bipartite and directed weighted graphs
- the derivation of a necessary condition for semantic interoperability in the large and its extension to approximate the degree of diffusion of a local query throughout a semantic network
- the description of an architecture to manage data, schemas and mappings in a scalable and totally decentralized way through a Distributed Hash Table
- the presentation of two mechanisms to resolve queries in a Semantic Overlay Network in an iterative or recursive fashion
- the formalization of annotation scarcity and semantic heterogeneity in a information entropic framework

- the description of novels methods to generate and interrelate semantic annotations in the context of a media sharing application based on data indexing, data imputation and decentralized data integration techniques.

Part I

Foundations

Chapter 2

On Integrating Data in the Internet Era

With the development of networking infrastructures came the need to exchange digital information across distributed organizations. While various efforts focused on standardizing the computer-to-computer exchange of structured messages, e.g., in the context of the *Electronic Data Interchange* EDI format[1], others tackled the difficult problem of relating heterogeneous data sources designed independently. *Data integration* aims at combining data residing at heterogeneous sources and providing the user with a unified view over those data.

Various paradigms were developed to integrate heterogeneous databases. The simplest approach – sometimes referred to as *Global Schema Integration* [BHP92] – consists in imposing a single global schema to all local databases. Imposing a global schema results in a tightly coupled collection of data sources with only limited autonomy and limited scalability, as designing the global schema requires an in-depth knowledge of all local databases. At the other end of the spectrum, *Multidatabase Languages* such as MSQL [LAZ+89] were designed to query collections of totally decoupled databases at run-time. Though preserving the autonomy of the data sources, this approach requires the formulation of elaborated queries involving all schemas, which is impractical in large-scale dynamic settings.

2.1 Federated Databases

Most research efforts in data integration focused on designing systems and methods to interoperate multiple databases, allowing to query the whole system as a single unit while requiring only loose coupling. In the following, we use the notion of *semantic interoperability* in that context, i.e., we say that two systems are semantically interoperable when

[1] see *http://www.x12.org/*

they can be queried using a uniform query interface. *Federated Databases* were developed towards that goal and allow the retrieval of data from multiple noncontiguous databases with a single query, even when the constituent databases are heterogeneous. Thus, federated databases provide a solution to integrate data coming from heterogeneous databases interconnected via a computer network. They come in different flavors (see Sheth and Larson [SL90] for a taxonomy) but today often revolve around a *wrapper-mediator* architecture [Wie92] to *reformulate* a query posed against a global schema into local queries processed by individual data sources. Figure 2.1 gives an overview of this architecture: multiple data sources (at the bottom of Figure 2.1) created independently but sharing similar information agree on a common – typically relational – data model to export their data. The databases are only loosely-coupled as they can keep their original structured schema internally. The *wrappers* are responsible to adapt local data such that they adhere to the common data model once exported. The *mediator* stores a global integrated schema and its relationships to the local schemas exported by the wrappers. Query resolution proceeds as follows: applications pose queries against the global schema at the mediator, which reformulates the queries in terms of the local schemas and sends subplans to be executed by the wrappers [Len02]. The wrappers process the queries sent by the mediator such that they are understandable by the sources, query the sources and translate the results to the common data model. Finally, the mediator collect all results from the wrappers and returns an integrated *fused* set of results to the applications.

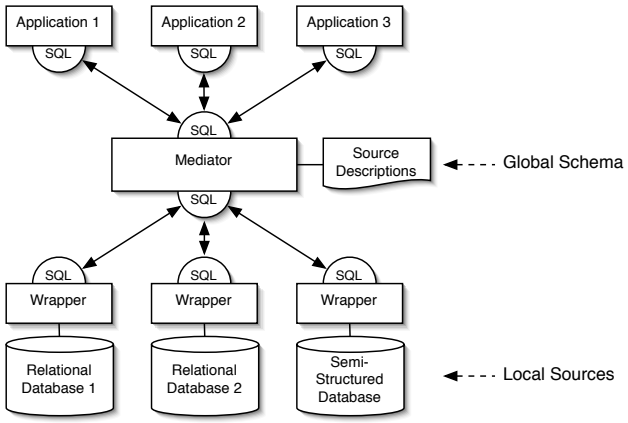

Figure 2.1: The Wrapper-Mediator Architecture. The Mediator offers an integrated interface to query heterogeneous data sources. Local sources retain some autonomy as their schema are only loosely coupled to the global schema through the schemas exported by the Wrappers and the source descriptions stored at the Mediator.

The relationships between the local sources and the global schema are stored at the mediator as *source descriptions*. Typically, sources are described by means of logical formulae like those used to express queries or views. Two main approaches have been proposed to relate the local sources to the global schema using views: the Global-As-View (GAV) approach, which describes the global schema in terms of the local sources, and the Local-As-View (LAV) approach, which describes the contents of a data source as a view over the global schema.

Reformulating a query posed against the global schema with GAV source descriptions is easy: since attributes from the global schema are given as views on the local sources, one simply has to replace the atoms of the global query by their corresponding expressions in terms of the local sources. This process in knows as *query unfolding* and was studied in the context of several research efforts [ACPS96, GMPQ+97, FTS00] including the Garlic project [HMN+99] and OBSERVER [MKSI00]. Figure 2.2 shows a simple example of query unfolding.

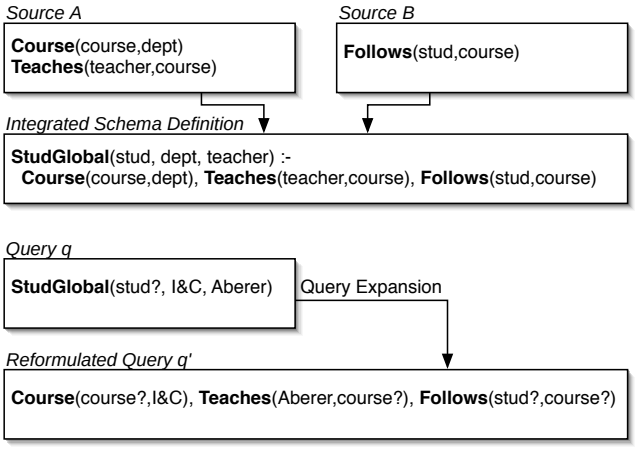

Figure 2.2: GAV-style query reformulation: query unfolding or expansion.

LAV, on the other hand, is source-centric, as it describes the local sources in terms of the global integrated schema. Query reformulation requires in that case techniques borrowed from query answering using views [Hal01], such as the Bucket [LRO96], the inverse-rules [DG97] or the MiniCon [PL00] algorithms. LAV systems such as Infomaster [GKD97] or the Information Manifold [OLR96] have received significant attention as their schemas scale gracefully with the number of sources: with GAV, adding a new source requires the redefinition of the global schema. With LAV, one only has to add a new source description at the mediator. In this context, sources are often considered as incomplete (open-world assumption, i.e., the extensions of the views might be missing tuples). Figure 2.3 shows an example of query answering using incomplete LAV sources.

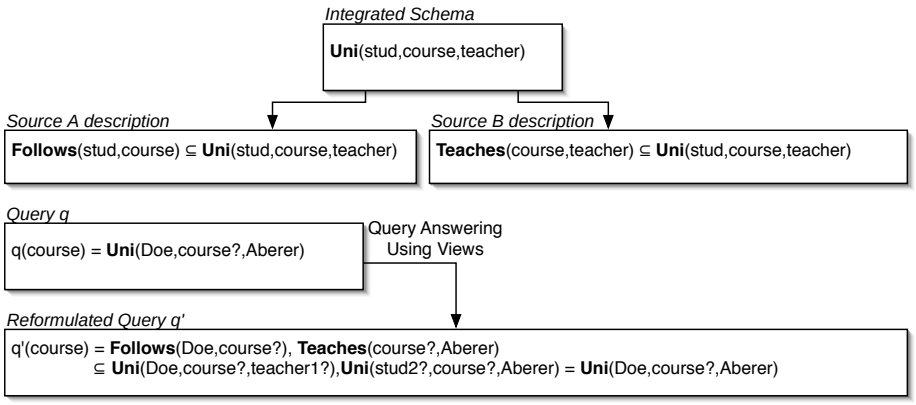

Figure 2.3: LAV-style query reformulation: query answering using views.

Note that there also exists a generalization of both GAV and LAV called Global-Local-As-View (GLAV) [FLM99]. In that model, the relationships between the global schema and the sources are established by making use of both LAV and GAV assertions.

The various assertions composing the source descriptions can be seen as *schema mappings* since they map attributes of local schemas onto attributes of the global schema (or the other way around). Mappings are typically created manually, though lots of recent research efforts [RB01] propose to partially automate the process. The mappings considered in this context typically produce equivalent or maximally-contained reformulations of the queries. A reformulated query q' is *equivalent* to the original query q if it always produces the same results as q, independent of the state of the database or of the views. A reformulated query q' is *maximally-contained* if it only produces a subset of the answers of q for a given state of the database [Hal01]. The maximality of q' is defined with respect to the other possible rewritings in a particular query language considered. The answers returned by equivalent or contained queries are called *certain answers* in the sense that they are answers for any of the possible database instances that are consistent with the given extensions of the views. Finding all certain answers is co-NP-hard in general (in terms of *data complexity*, i.e., complexity of the problem as a function of the size of the instance data), but can be done in polynomial time in many practical cases (e.g., when both the query and the mapping are defined as conjunctive queries) [AD98].

Finally, note that a slightly different notion of interoperability was recently proposed in the context of *data exchange* settings. Data exchange is the problem of taking data structured under a source schema and creating an instance of a target schema that reflects the source data as accurately as possible [FKMP05]. This contrasts with the data integration scenario

where data is retrieved and shared on-demand at query time.

2.2 XML, RDF and the Semantic Web

Federated databases were conceived before the World Wide Web, when information was typically created, curated and processed locally or at a few distant sites known a priori. Today, the focus is on developing new tools and standards for creating and processing data originating from multiple, heterogeneous and dynamic sources. The Extensible Markup Language (XML) [BPSM+04] was a first important step towards this vision. XML provides a common syntax to share user-defined data across distributed systems, to describe the schemas of those data (for example through XML Schemas [FW04]) and to query the data (for example using XQuery [BCF+06]).

Recently, the Semantic Web[2] vision attracted much attention in a similar context. The Semantic Web envisions an evolutionary stage of the World Wide Web in which software agents create, exchange, and process machine-readable information distributed throughout the Web. The Resource Description Framework (RDF) [MM04] is one of the building blocks of the Semantic Web (see Figure 2.4). RDF is a framework designed to create statements about resources in the form of *subject-predicate-object* expressions called *RDF triples*. RDF vocabularies – classes of instances and predicates – can be defined with the RDF-Schema language RDFS [BG04]. More expressive definitions of the vocabulary terms can be written as ontologies, for example with The Web Ontology Language (OWL) [Mv04]. Ontologies can be seen as explicit integrated schemas used by distributed and potentially numerous information sources. However, ontologies are not the panacea in data integration [OA99]: by statically fixing semantic interpretations, ontologies do not always provide the flexibility necessary to handle query dependent integration between autonomously and independently designed information systems. Furthermore, standardization efforts aiming at defining upper ontologies promoting data interoperability in the large have failed so far[3]. In the Semantic Web context as well, semantically related ontologies need to be integrated if one wants to be able to interact with as many informations sources as possible. It is thus unsurprising to witness today an intense research activity on specialized integration techniques for the Semantic Web (also referred to as *ontology alignment* [Euz04b] techniques).

Whether or not the vision of the Semantic Web will one day be realized is still subject to discussion. The proliferation of machine-processable and structured data – encoded in XML, RDF, or in proprietary formats

[2] see the W3C Semantic Web Activity at *http://www.w3.org/2001/sw*

[3] see *http://www.ontologyportal.org/* for an example related to the Suggested Upper Merged Ontology (SUMO)

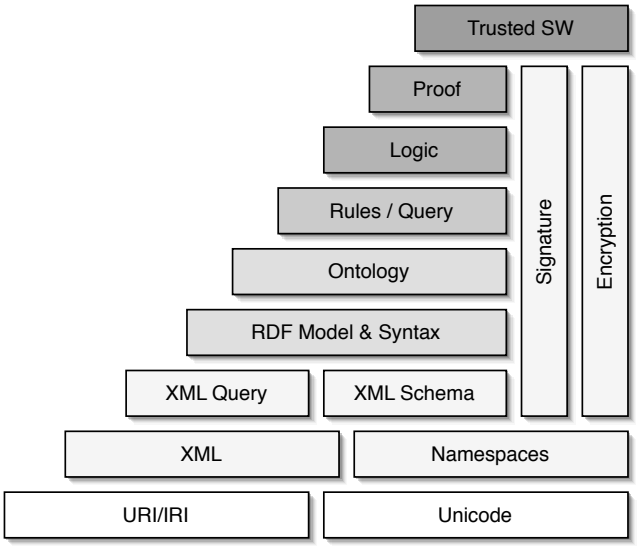

Figure 2.4: The Semantic Web stack: The Semantic Web builds on existing standards such as XML or RDF to define and share data, but also requires higher-layer standards related to logic or trust.

such as those promoted by Google[4]– is however already a reality on the Internet. As all the aforementioned formats are extensible – in the sense that end-users can define new schemas to organize their data according to their own preferences – the research community is confronted with a new grand challenge: integrating data originating from thousands of heterogeneous, dynamic and potentially unknown data sources. This new challenge sharply contrasts with previous challenges faced in the field of data integration in several ways:

Scale: Federated databases were designed to integrate a limited number of sources, typically a few dozens. Today, hundreds of heterogeneous sources often have to be integrated for one particular application (see Section 7.7 for an example related to the bioinformatic domain). More general applications such as the Semantic Web – where all resources are uniquely identified and globally shared – require the integration of tens of thousands of disparate sources.

Uncertainty: Databases integrated via a wrapper-mediator architecture were typically created and curated by database administrators. With the new formats described above, end-users themselves are supposed to create schemas and data. Also, more and more applications attempt to create data and metadata automatically (e.g., in the context of sensor networks), with all the associated

[4]see *http://base.google.com/* or *http://www.google.com/coop*

problems in terms of data quality. Thus, on an open system such as the World Wide Web, one cannot expect any level of quality for the schemas, schema mappings or data provided by a distant and potentially unknown source. Uncertainty on all pieces of information has to be taken into account in all aspects of data processing and retrieval to filter out potentially spurious information.

Dynamicity: Federated databases used to integrate a relatively stable set of sources. On a large-scale decentralized infrastructure such as the Internet, on the other hand, information sources come and go on a continuous basis. New sources have to be integrated on the fly with minimum overhead. Also, systems have to be resilient to all sorts of node failures, including the failure of central indexes or centralized servers such as mediators.

Limited expressivity: Compared to relational data, data available on the World Wide Web in XML or RDF often suffer from relatively simplistic structures (schema and data models). Data are available in very crude ways, often comparable to simple Web forms. Few constraints other than foreign key relationships are supported, and transactional support is typically nonexistent. Languages used to map schemas or ontologies are equally limited and often revolve around simple one-to-one matching of attributes or classes [Euz04a, Mv04].

The emergence of this new information ecology requires new techniques to process and integrate data in a consistent and scalable way. We propose hereafter a new paradigm to integrate information in the large based on self-organization and decentralization principles. The following chapter starts with a discussion of several decentralized architectures in the context of Peer-to-Peer systems.

Chapter 3

Peer-to-Peer Information Management

Peer-to-Peer (P2P) systems rely on machine-to-machine ad-hoc communications to offer services to a community. Contrary to the classical client-server architecture, P2P systems consider all peers, i.e., all nodes participating in the network, as being equal. Hence, peers can at the same time act as clients consuming resources from the system, and as servers providing resources to the community. Figure 3.1 shows a client-server and a P2P system side by side.

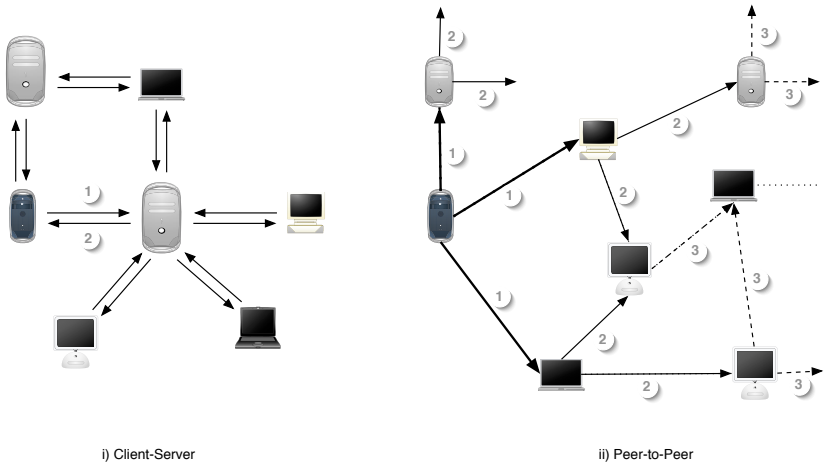

Figure 3.1: Two architectures: i) the client-server paradigm where clients' requests all converge to a set of centralized servers and ii) the P2P paradigm where peers collectively gossip a request originating from the peer located on the extreme left.

The resources shared by the peers through the P2P infrastructure can vary widely: USENET [HA87] was an early P2P system for disseminating

news articles. Napster[1] used to focus on the P2P streaming of MP3 songs. Groove[2] is a desktop software designed to facilitate collaboration and communication among small groups, where workspaces (e.g., calendars, sketchpads) can be created and edited in a P2P way. Skype[3] is a P2P Internet telephony network. A few distinct advantages nurture the use of P2P technologies in those various domains:

Scalability: P2P systems tend to eliminate all central components. Instead of requesting services from centralized servers, the peers collaboratively contribute resources to support shared applications. This ensures a graceful scalability of most P2P networks: new clients requesting services have to connect to a few existing peers to join the P2P network. Once they are part of the system, they in turn have to provide resources and act as servers intermittently to support shared services.

Autonomy: As P2P systems function without any central administration, peers have to proactively and autonomously ensure that the P2P network is running correctly in their vicinity. This contrasts with centralized applications where dedicated machines administered by a central authority have to be monitored continuously to ensure the continuity of the service. This of course comes at a price in the P2P network, which has to implement self-examination and self-stabilization mechanisms.

Robustness: As by-product of the first two points, P2P systems continue their operations in case of node failures. P2P networks do not present any single point of failure, and can circumvent the failure of any node by provisioning resources from other peers.

3.1 From unstructured to structured P2P Systems

P2P applications function on top of existing routing infrastructures – typically on top of the IP network – and organize peers into logical structures called *overlay* networks. The structure of the overlay network can vary; the right-hand side of Figure 3.1 presents an *unstructured* overlay network, similar to the structure used for example in Gnutella [Cli01]. In unstructured overlays, peers establish connections to a fixed number of other peers, creating a random graph of P2P connections.

Requests originating from one peer are forwarded by the other peers in a cooperative manner. The propagation of the query is regulated by

[1] *http://www.napster.com*
[2] *http://www.groove.net/*
[3] *http://www.skype.com/*

a Time-To-Live (TTL) value, which indicates the number of times the query is to be iteratively forwarded before being discarded. This relatively simple and robust mechanism is however network-intensive, as it broadcasts all queries within a certain radius irrespective of their content. To discover new nodes, peers can rely on similar mechanisms by sending out special queries, e.g., *ping* descriptors in Gnutella, that trigger an automatic response, e.g., *pong* messages in Gnutella, from all the other peers receiving the query.

Structured overlay networks were introduced to alleviate network traffic while maximizing the probability of a query locating a specific peer. The left-hand side of Figure 3.2 shows how peers can be organized into a structured virtual binary search tree, as promulgated by the P-Grid P2P system [Abe01, ACMD+03]. Other well-known structured P2P systems organize the peers on a multi-dimensional torus [RFH+01] or around a circle [RD01, SMK+01]. Those systems provide hash-table functionalities on an Internet-like scale, and are today known as Distributed Hash-Tables (DHTs). They typically find a data item in a totally decentralized way in $\mathcal{O}(log(N))$ messages, where N is the number of peers in the system.

Observing that some peers are more equal than others, i.e., that some peers can provide more resources than others in a P2P system, several systems – most notably Kazaa [LKR06] – adopted a two-layered P2P structure called *super-peer* [YGM03] architecture. Figure 3.2 ii) shows a super-peer overlay network where peers are organized in a typical two-tier hierarchy: simple peers connect to one of the four super-peers, which supposedly enjoy greater stability, superior bandwidth or CPU capabilities. The super-peers act as proxies for the simple peers: they take care of indexing the data items of the simple peers, and forward their requests in the super-peer network. The super-peer network of Figure 3.2 ii) is organized in a structured hyper-cube overlay, similar to the structure used in Edutella [NWS+03].

Note that the crude classification adopted above does not do justice to the wealth of research directions currently explored in the P2P field. Other overlay structures have been suggested (e.g., Butterfly networks, de Bruijn graphs), and numerous mechanisms have been proposed to tackle problems ranging from resilience to attacks to load balancing, reputation or identity management. We refer the interested reader to recent surveys of the field [RM04, Hel04] for further details.

3.2 Peer Data Management

P2P systems originally dealt with very simple data and query models: only filenames were shared and queries were composed of a single hash value or a keyword. Rapidly, several research efforts tried to enrich P2P systems with more expressive data models. Edutella [NaCQD+02] is a P2P system for exchanging metadata in RDF. Originally built on top

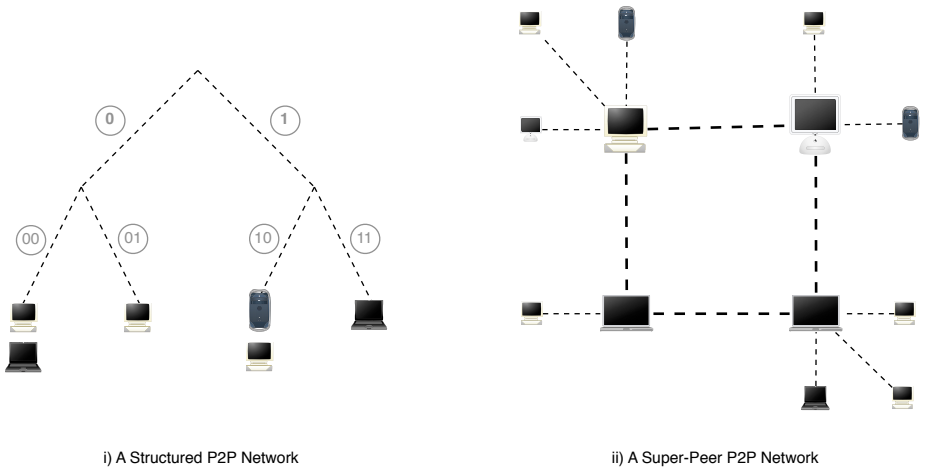

Figure 3.2: Two P2P architecture: i) a structured P2P network *à la* P-Grid where peers are organized in a virtual binary shared tree and ii) a super-peer network *à la* Edutella where simple peers cluster around four super-peers.

of JXTA[4], it later evolved to support publish-subscribe functionalities for RDF and RDF Schema data on a super-peer architecture [NWS+03]. RDFPeers [CF04a] indexes RDF and RDF Schema data in a DHT. PeerDB [BOT03] is based on the BestPeer [NOT02] P2P system and allows the sharing of relational data through attribute-keyword matching. PIER [HCH+05] is a full-blown, distributed and relational database system built on top of a DHT.

In early 2002, we proposed a radically different approach [ACMH02]: instead of augmenting P2P systems with richer query processing capabilities, we decided to extend the wrapper-mediator architecture in a P2P way. We introduced two fundamental concepts:

1. A decentralized mediator: the centralized mediator (see Figure 2.1) represents a single point of failure for traditional federated databases. Also, the definition and maintenance of a global schema is impractical in large scale decentralized environments. As a result, we decided to decentralize the mediator. Local data sources continue to operate in total autonomy, but define a few mappings to related databases. Disposing of a query reformulator locally, individual databases can in that way query neighboring databases by reformulating a query posed against their local schema thanks to local schema mappings.

2. A query routing mechanism based on iterative reformulations: requiring the definition of local mappings between all pairs of seman-

[4] *http://www.jxta.org/*

tically related data sources would be unrealistic in large scale settings. Instead, we devised a query gossiping mechanism to propagate queries in an iterative and collaborative manner throughout the network as in an unstructured P2P system (Figure 3.1 ii): once a database sends a reformulated query to its immediate neighbors, its neighbors (after processing the query) can in turn propagate the query to their own neighbors, and so on and so forth until the query reaches all (or a predefined subset of) databases.

This architecture, explored in different ways in the context of the Piazza project [HIMT03], is today known as a *Peer Data Management System* (PDMS). Figure 3.3 depicts a PDMS where all peers support a SPARQL [PS06] query interface to retrieve RDF data from neighboring peers.

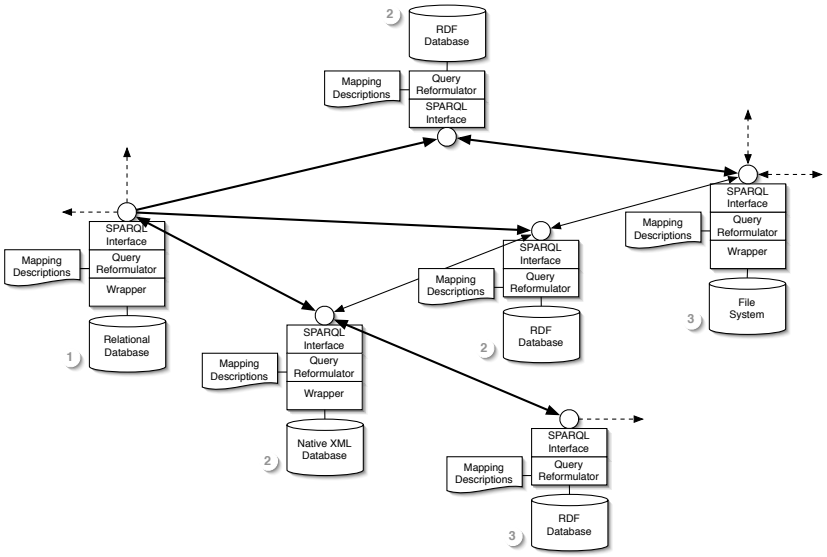

Figure 3.3: A Peer Data Management System: the peer on the left-hand side reformulates a local query (1) thanks to local schema mappings and sends it to its neighbors (2); its neighbors, in turn, can reformulate the query and propagate it to their own neighbors (3).

Peer data management is a natural paradigm for integrating data in the large. For a new peer wishing to join an existing PDMS, the cost of entry is minimal as for most P2P systems: the new peer only has to define a few schema mappings between its schema and the schemas of other peers already connected to the system. The peers can continue to handle their data the way they want, and only have to perform timely local updates when their schema or the schema of their direct neighbors evolve. In case of an intermediate node failure, peers can reroute their queries through different schema mappings or create new mappings to

circumvent the offline peer and continue to query distant sources.

Note that in practice, a single schema (or ontology) can be used simultaneously by many independent parties. Furthermore, some peers might choose more than one schema to structure their data locally, as they realistically might have to handle very diverse pieces of information. Hence, the organization of the schemas and mappings can often be uncorrelated with the organization of the peers themselves. Figure 3.4 shows a Semantic Overlay Network (SON), where physical machines form a P2P overlay network, which is itself independent of the logical overlay handling data integration. Chapter 8 discusses in more details this three layer architecture in the context of the GridVine system. The following chapter starts by examining query resolution in a simpler, unstructured PDMS setting.

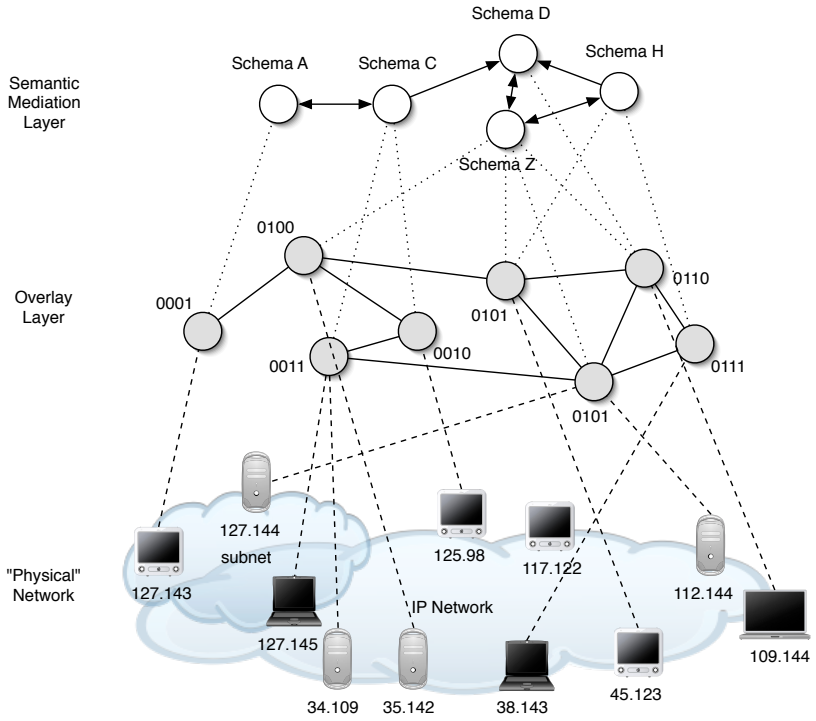

Figure 3.4: A Semantic Overlay Network: in many practical settings, the semantic mediation layer is independent from the organization of the peers, which is itself dissociated from the physical network structure of the machines.

Part II

Methods

Chapter 4

Semantic Gossiping

This chapter introduces *Semantic Gossiping* [ACMH03b], a new technique to selectively forward queries from one peer database to the others in a PDMS network. A main contribution of this chapter is the introduction of a tradeoff between the precision and recall of the set of answers to a query. Another contribution is the description of different methods that can be applied to estimate the quality of local query reformulations in large-scale, uncertain PDMS networks. We elaborate the details of each of these methods for a simple data model, that is yet expressive enough to cover many practical cases (see Section 4.2). The methods that we introduce consist of:

1. A syntactic analysis of search queries after reformulations have been applied in order to determine the potential information-loss incurred through a reformulation. We analyze to which degree query constituents are preserved during reformulation (Section 4.4).

2. A semantic analysis of composite reformulations along cycles in the schema mapping graph, in order to determine the level of agreement that the peers achieve throughout the cycle. We analyze whether cyclic transformations preserve semantics. If the semantics of a query are not preserved after a cyclic series of reformulations, we assume that some semantic confusion has occurred (Section 4.5.1).

3. A semantic analysis of search results obtained through composite mapping reformulations. We assume that structured data is used to annotate content and that the peers can classify their documents both using content analysis and metadata-based classification rules. From that classification, peers derive to which degree transformed metadata annotations match the actual content and thus how reliable the reformulations were. (Section 4.5.2).

The first analysis is related to the completeness of a mapping, i.e., to the extent to which a mapping can reformulate arbitrary query constituents. The second and third analyses are related to the soundness of

the mapping, i.e., to the degree of correctness of the reformulations. We start below by discussing those two notions in the context of uncertain schema mappings in large-scale decentralized settings, which emphasizes the need for query-specific forwarding schemes. We propose a generic model for PDMSs in Section 4.2. Our syntactic and semantic analyses are described in Section 4.4 and Section 4.5 respectively. Section 4.6 is dedicated to query forwarding. We describe experimental findings in Section 4.7 and discuss related work in Section 4.8 before concluding.

4.1 On Uncertain Schema Mappings in Decentralized Settings

Our focus is quite different from previous work in federated databases query processing, which only considered certain schema mappings generating maximally contained query rewritings (see Chapter 2). In what follows, we devise methods taking advantage of the incompleteness and uncertainty of schema mappings to direct searches in a network of semantically heterogeneous information sources. Our methods are purely local and query-specific, offering a tradeoff between the recall of the set of answers retrieved – related to the completeness of the mappings – and the precision of the results – related to the notion of mapping soundness.

4.1.1 Mapping Completeness

Schema mappings can not always reformulate all the constituents of a query. They can be incomplete for several reasons: for large schemas or ontologies, some mappings can be specifically designed to handle the reformulation of some of the attributes only, while ignoring the rest of the schema. As the information sources are considered as being autonomous in our setting, they can structure their data according to their own preference and activities. Thus, we can expect irreconcilable differences on conceptualizations (e.g., epistemic or metaphysical differences on a given modeling problem [Bou04]) among the databases. Also, the limited expressivity of the mappings, usually defined as queries or using an ontology definition language like OWL, precludes the creation of correct mappings in many cases (e.g., mapping an attribute onto a relation). Depending on the situation, the creator of the mapping might then either produce an approximate mapping (see below), or simply leave the mapping incomplete by ignoring the attributes that are irreconcilable.

We introduce the notions of mapping completeness to characterize the exhaustiveness of the mappings connecting semantically related schemas. We say that a mapping between a source and a target database is *complete* if it can reformulate all atoms of all queries from the source database. Conversely, a mapping is incomplete when there exists a source query that cannot be reformulated into a target query. Note that the notion of

mapping completeness can also be given relative to a subclass of queries. We say that a query reformulation is incomplete when there is at least one atom of the query that cannot be reformulated. Incomplete reformulations are typically discarded in federated database settings. In the following, we introduce the notion of syntactic similarity to quantify the degree of incompleteness of a reformulation. We proceed in a best-effort manner and process incomplete query reformulations as long as they can still be used to retrieve sensible results.

4.1.2 Mapping Soundness

Federated databases systems reformulate queries without any concern on the validity or quality of the mappings used. This obviously represents a severe limitation in a our setting, as one cannot expect any level of consistency or quality on schema mappings in PDMSs: as PDMSs target large-scale, decentralized and heterogeneous environments where autonomous parties have full control on the design of the schemas, it is not always possible to create correct mappings between two given schemas (see above, irreconcilable difference on conceptualizations and limited expressivity of the mapping languages). In many situations, an approximate mapping relating two similar but semantically slightly divergent concepts might be more beneficial than no mapping at all. Also, given the vibrant activity on (semi) automatic alignment techniques [Euz04b], we can expect some (most?) of the mappings to be generated automatically in large-scale settings, with all the associated issues in terms of quality.

We introduce the notion of mapping soundness to characterize the correctness of the reformulations. We say that a mapping is *sound* if it always produces equivalent rewritings (see Section 2.1) of all query atoms it reformulates. Conversely, a mapping is unsound if there exists at least one query atom it reformulates into a non-equivalent rewriting. A query reformulation is sound when it only contains equivalent rewritings of the atoms of the original query. Note that a reformulation can be sound but incomplete, e.g., when one atom of the original query is dropped, or complete but unsound, e.g., when all atoms are reformulated in semantically incorrect ways. In the following, we introduce the notion of semantic similarity to quantify the degree of soundness of the various rewritings. Note that we extend the notion of mapping soundness in Chapter 6 to include contained rewritings and subsumption hierarchies.

4.2 The Model

We start our discussion with a generic model for PDMSs that will be used throughout the rest of this thesis. Our model consists of a data model, describing the local databases of the peers, and a network model, characterizing the schema mappings and the organization of the peers.

4.2.1 The Data Model

We model each information system as a peer $p \in \mathcal{P}$. A peer stores data in a database DB_p according to a structured schema S_p taken from a global set of schemas $\mathcal{S}$. As we wish to present an approach as generic as possible, we do not make any assumption on the exact data model used by the databases in the following but illustrate some of our claims with examples in XML and RDF. We only require the schemas to store information with respect to some concepts we call *attributes* $A \in \mathcal{S}_p$ (e.g., attributes in a relational schema, elements or attributes in XML and classes or properties in RDF). We only consider one relation per local database DB_p for simplicity in the following, but discuss extensions of our methods to support multi-relations and local join operations in Section 4.9.

Each local attribute is assigned a set of fixed interpretations A^I from an abstract and global domain of interpretations Δ^I with $A^I \subset \Delta^I$. Arbitrary peers are not aware of such assignments. We say that two attributes A_i and A_j are *equivalent*, and write $A_i \equiv A_j$ if and only if $A_i^I = A_j^I$. Even if equivalent attributes theoretically have the same extensions, some tuples might be missing in practice (open-world assumption), i.e., DB_{p_i} is not always equivalent to DB_{p_j} even if p_i and p_j share identical or equivalent schemas. Those sets of interpretations are used to ground the semantics of the various attributes in the PDMS from an external and human-centered point of view (see Chapter 1).

Attributes may have complex data types and NULL-values are possible. We do not consider more sophisticated data models to avoid diluting the discussion of the main ideas through technicalities related to mastering complex data models. Moreover, many practical applications, in particular in P2P systems, digital libraries or scientific databases, use exactly the type of data model we have introduced, at least at the meta-data level.

We use a query language for querying and transforming databases. The query language builds on basic relational algebra operators since we do not care about the practical encoding, e.g., in SQL, XQuery or SPARQL. For a peer p structuring its data according to a schema S_p, we consider the following operators:

- Projection $\pi_{\boldsymbol{pa}}$, where $\boldsymbol{pa}$ is a list of attribute names $(A_1, \ldots, A_k) \in S_p$.
- Selection $\sigma_{\boldsymbol{pred(sa)}}$, where $\boldsymbol{sa}$ is a list of attributes $(A_1, \ldots, A_k) \in S_p$, and $\boldsymbol{pred}$ are predicates on the attributes $\boldsymbol{sa}$ using comparison predicates on the respective data types, e.g., $\sigma_{\boldsymbol{pred(sa)}} = \sigma_{A_i<A_j,\ A_k='Doe'}$.
- Renaming $\rho_{\boldsymbol{f(ra)}}$, where $f \in \boldsymbol{f}$ are functions of the form $A_0 := f(A_1, \ldots, A_k)$ with $(A_1, \ldots, A_k) \in S_p$. The functions f are specific to the attributes and encompass string operations and simple

arithmetic operations to syntactically align heterogeneous data values. A concatenation of two strings could for example be written as $\rho_{A_{concat}:=(A_1\ cat\ A_2)}$. A special case is the renaming of a single attribute A_{old} into a new attribute A_{new}: $\rho_{A_{new}:=A_{old}}$.

We write $q(S_i)$ to denote a query formulated in terms of a particular schema S_i, and $q(DB_i)$ to pose a query against a database DB_i.

4.2.2 The Network Model

Let us now consider a (potentially large) set of peers $\mathcal{P}$ with their related schemas and data. We suppose that all peers and schemas can be identified by unique identifiers ID_p (e.g., an IP address or a peer ID in a P2P network) and ID_S. Each peer $p \in \mathcal{P}$ has a basic communication mechanism that allows it to establish connection to other peers. Without loss of generality, we assume in the following that it is based on an unstructured P2P access structure *à la* Gnutella (we detail how to implement our mechanisms on a structured DHT in Chapter 8). Thus, peers send *ping* messages with a certain Time-To-Live value and receive *pong* messages in order to learn about the network structure. In extension to the Gnutella protocol, peers also send their schema identifier ID_{S_p} as part of the *pong* messages.

Schema Mappings

Peers can define schema mappings $\mu_{S_i \to S_j}$ between a source schema S_i and a target schema S_j. These mappings can be created manually, semi-automatically or fully automatically depending on the peers and the setting. A mapping $\mu_{S_i \to S_j}$ allow to reformulate a query posed against the source schema S_i into a new query against the target schema S_j. Mappings $\mu_{S_{p_i} \to S_{p_j}}$ can be created by a source peer p_i, by a target peer p_j, or by a third-party peer that sends it to the source peer p_i.

Schema mappings can be expressed in various ways (see Chapter 2); in our case, we consider mappings $\mu_{S_i \to S_j}$ given as queries $q_{i \to j}$ against the target schema S_j:

$$\mu_{S_i \to S_j} = q_{i \to j}(S_j) = \rho_{\boldsymbol{f(ra)}}(\pi_{\boldsymbol{pa_\mu}}(\sigma_{\boldsymbol{pred(sa_\mu)}}(S_j)))$$

where source attributes $A_k \in S_i$ appear as new attribute names (A_0's) in the renamings of $q_{i \to j}$. Target attributes are thus mapped onto source attributes $A_k \in S_i$ in subparts of the query $q_{i \to j}$ we call *attribute mappings* $m_k(A_k)$. Source attributes cannot appear in more than one attribute mapping in given schema mapping. An attribute mapping is sound if it relates equivalent pieces of information at the intensional level, that is if $A_k^I \equiv m_k^I(A_k)$.

In that way, a mapping defines a surjective operation from the set of target attributes onto the set of source attributes, where source attributes

that do not appear in the mappings are mapped by an implicit attribute mapping onto a null value. Hence, we express definitional mappings for the source attributes at the intensional level and can reformulate a source query $q_i(S_i)$ into a target queries $q_j(S_j)$ using GAV-style query expansion:

$$q_j(S_j) = \mu_{S_i \to S_j}(q_i) = q_i(q_{i \to j}(S_j)).$$

Figure 4.1 gives an example of query reformulation in an XML/XQuery context. The attribute mapping for p_1's *lenghth* shows an example of value conversion, as the length of the project is derived from the start and end dates stored at p_j. Additional data conversions could be handled in that context using standard XML functions and operators, e.g., additions, concatenations etc. [MME06].

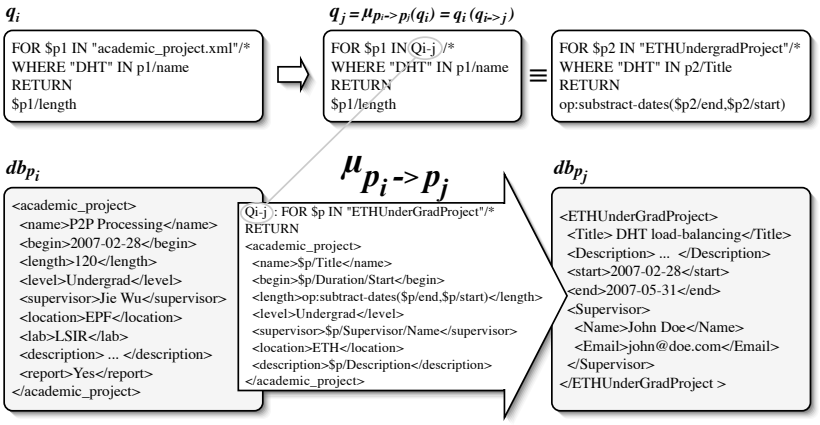

Figure 4.1: A query reformulation: query q_i gets reformulated into query q_j thanks to mapping $\mu_{p_i \to p_j}$ composed of seven attribute mappings m mapping target attributes onto source attributes.

The authors propose an efficient method to measure the structural similarity of pairs of XML documents; the method starts by linearizing the structure of the documents (e.g., by considering a flow of elements as in SAX). Then, a compression method is used to encode the linearized versions, using Ziv-Lempel (gzip) or Ziv-Merhav compression techniques. The distance between the two documents is derived by comparing the size of the two compressed versions of the documents and the compressed version of the concatenation of the two documents. The time complexity of the algorithm is O(n) where n is the number of tags in both documents. Experimental evaluations show that the algorithm is as efficient as previous algorithms based on edit-distances of Fourier transforms.

Definitional mappings are expressive enough to handle most data integration cases on the Internet. Furthermore, they can be handled by standard query processors and do not require any additional mechanism to reformulate queries using views. However, the techniques described

hereafter can easily be extended to other classes of mappings: we discuss more general conjunctive mappings and GLAV mappings in Section 4.9, and containment mappings in Chapter 6.

Semantic Neighborhoods

Every peer p maintains a neighborhood $N(p)$ selected from the peers that it identified through *pong* messages. The peers in this neighborhood are distinguished into those that share the same schema, $N_e(p)$, and those that have a different schema, $N_d(p)$. A peer p_i includes another peer p_j with a different schema into its neighborhood if it knows a schema mapping $\mu_{S_{p_i} \to S_{p_j}}$ for reformulating queries against its own schema into queries against the foreign schema. For simplicity, such mappings will be written as $\mu_{p_i \to p_j}$ in the following. Figure 4.2 gives an example of both neighborhoods for one peer.

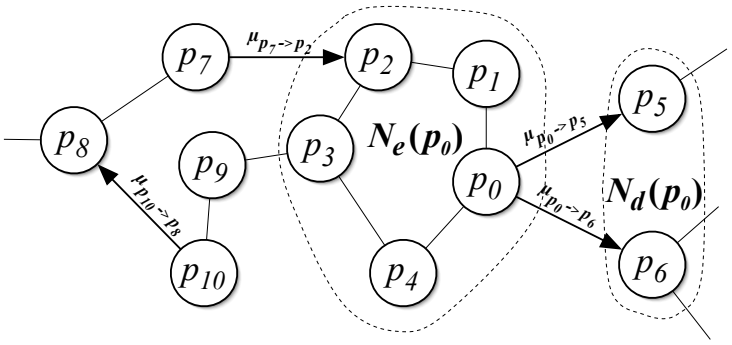

Figure 4.2: The network model: p_0 maintains a neighborhood of nearby peers using its own schema ($N_e(p_0)$) and using different schemas ($N_d(p_0)$).

Query Dissemination

Queries get disseminated in our PDMS network in an unstructured and collaborative way (see Chapter 3). A peer receiving a reformulated query may decide to reformulate it in turn. Thus, queries can be reformulated several times iteratively:

$$\begin{aligned} q_N(S_N) &= \mu_{S_{N-1} \to S_N} \circ \ldots \circ \mu_{S_1 \to S_2}(q_1) \\ &= q_1(q_{S_1 \to S_2}(\ldots(q_{S_{N-1} \to S_N}(S_N))\ldots)) \end{aligned}$$

In that way, queries might traverse several semantic domains through a succession of schema mappings. Figure 4.3 shows an example of a logical semantic graph, where nodes stand for schemas, and edges represent schema mappings created by individual peers and used to reformulate the

queries. Note that a given pair of nodes can be related through more than one edge, e.g., when two independent parties provide two different mappings between a given pair of schemas.

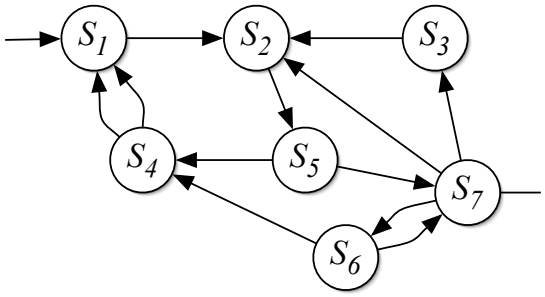

Figure 4.3: A logical semantic graph, where schemas are interlinked by schema mappings provided by the peers.

Queries can get propagated through the semantic graph in various ways, depending on the query forwarding paradigm in use. Forwarding a query irrespective of its content within a certain radius with a constant Time-To-Live (TTL) value is highly inefficient: for low TTL values, *recall* [Rij79] is low as the system cannot reach all databases relevant to the query. A high TTL value, on the other hand, potentially forwards the query through many incomplete or unsound mappings, which results in retrieving many irrelevant results (low precision).

Adopting well-known optimization techniques for search in unstructured networks, such as distributed replication [LCC+02] or dynamic routing tables [CSWH00], would largely fail as well: as queries targeting the same pieces of information can consider very different selection predicates, they may require very different reformulation paths in the semantic graph, thus making it very difficult to optimize the network based on source / destination records only. In the following, we introduce query-dependent per-hop forwarding behaviors to selectively disseminate queries throughout the semantic network.

4.3 Overview

Before delving into the technical details, this section provides an informal overview of our approach and of the rest of the chapter.

The semantic graph (Figure 4.3) has two interesting properties: (1) as already pointed out earlier, based on existing mappings and the ability to learn about new mappings, queries can be propagated to peers for which no direct mapping exists by means of transitivity, and (2) the graph has cycles, for example $S_2 \rightarrow S_5 \rightarrow S_7 \rightarrow S_2$. We call (1) *Semantic Gossiping*. (2) gives us the possibility to assess the degree of *semantic agreement*

along a cycle, i.e., to measure the soundness of the mappings and the degree of semantic agreement in a community in a decentralized way.

In the following, we expect peers to perform several tasks: (1) upon receiving a query, a peer has to decide to where to forward the query based on a set of criteria that will be introduced; (2) upon receiving results or feedback along mapping cycles, it has to analyze the quality of the results at the schema and at the data level and adjust its criteria accordingly and (3) update its view of the overall semantic agreement by modifying its query forwarding criteria or by adjusting the mappings themselves.

The criteria to assess the quality of the reformulations – which in turn is a measure of the degree of semantic agreement – can be categorized as *context-independent* and *context-dependent*. Context-independent criteria, discussed in Section 4.4, are syntactic in nature and relate to the completeness of the reformulations involving local mappings. We introduce the notion of *syntactic similarity* to quantify the extent to which a query reformulation is complete.

Context-dependent criteria, which are discussed in Section 4.5, relate to the degree of agreement that can be achieved among different peers upon sets of attributes. Degrees of agreement can be evaluated using feedback mechanisms. We introduce two such feedback mechanisms below, based on cycles appearing in the mapping graph and on results returned by different peers. A peer might locally obtain both returning queries and data through multiple feedback paths. In case a disagreement is detected (e.g., a wrong attribute mapping at the schema level or a concept mismatch at the content level), the peer has to suspect that at least some of the mappings involved in the reformulation path were unsound, including the mapping it has used itself to propagate the query. Even if an agreement is detected, it is not clear whether it is not accidentally the result of compensating mapping errors along the path. Thus, analyses are required to assess the most probable sources of errors along the paths and to determine the extent to which local mappings can be trusted and therefore used in future routing decisions. At a global level, we can view the problem as follows: the mappings between domains of semantic homogeneity form a directed graph. Each mapping cycle or query result allows to return feedback to the query originator, which in turn can make an analysis to assess the degree of semantic agreement pertaining to the mapping used. We introduce the notion of *semantic similarity* to quantify the soundness of the query reformulations in that context.

Each of the similarity measures characterizing the query reformulations results in a *feature vector*. The decision whether or not to forward a query using a mapping is then based on the values of those feature vectors. The details of the query forwarding process are provided in Section 4.6.

Assuming that all the peers implement this approach, we expect the network to self-organize into a state where queries get disseminated to the subset of the peers most likely able to process them in a sensible way,

where the correct mappings are increasingly reinforced by adapting the per-hop forwarding behaviors and where incorrect mappings are rectified (see also Chapter 5). Implicitly, this is a state where a global agreement on the semantics of the different schemas has been reached.

4.4 Syntactic Similarity

Parts of the queries might get lost during reformulation, due to incomplete mappings or schemas missing a representation for some of the attributes present in the original query. *Syntactic similarity* provides a measure that is related to information loss during reformulation. This measure is context-independent, since its evaluation relies exclusively on the inspection of the syntactic features of the reformulated queries. A high syntactic similarity does not ensure that forwarding a query is useful, but conversely a low syntactic similarity implies that it might not be useful to forward a query any further.

Let us suppose we have a query q_1, originally posed against a database DB_1 with schema S_1, which has the generic form of a selection-projection query

$$q_1(S_1) = \pi_{\boldsymbol{pa}}(\sigma_{\boldsymbol{pred(sa)}}(S_1)),$$

where $\boldsymbol{pa}$ is the list of attributes used in the projection and $\boldsymbol{sa}$ is the list of attributes used in the selection predicates.

Let us assume that a mapping $\mu_{S_1 \to S_2}$ is given, such that q_1 can be reformulated into a second query q_2 in terms of a second database DB_2 according to schema S_2. The transformation is specified by a query q_μ defining a view on S_2

$$\mu_{S_1 \to S_2} = q_\mu(S_2) = \rho_{\boldsymbol{f(ra_\mu)}}(\pi_{\boldsymbol{pa_\mu}}(\sigma_{\boldsymbol{pred(sa_\mu)}}(S_2))).$$

The reformulated query q_2 is given in terms of schema S_2 and takes the following form

$$q_2(S_2) = \pi_{\boldsymbol{pa}}(\sigma_{\boldsymbol{pred(sa)}}(\rho_{\boldsymbol{f(ra_\mu)}}(\pi_{\boldsymbol{pa_\mu}}(\sigma_{\boldsymbol{pred(sa_\mu)}}(S_2))))).$$

This form will also be achieved after multiple transformations after normalization.

It might occur that attributes appearing in q_1 are no longer available after applying mapping $\mu_{S_1 \to S_2}$ onto q_1. This happens when an attribute from S_2 required for the derivation of an attribute from S_1 and occurring in $\boldsymbol{pa}$ or $\boldsymbol{sa}$ is missing, or is not computed by one of the functions from $\boldsymbol{f(ra_\mu)}$.

We now determine which attributes are needed in order to properly evaluate the query q_1. For an attribute $A \in \boldsymbol{sa}$ resp. $A \in \boldsymbol{pa}$ we define $target_\mu(A)$ as the set of attributes required in schema S_2 of database DB_2 in order to derive A by means of a mapping μ. If attribute A cannot be

derived we set $target_{\mu}(A) = \perp$. For a composite reformulation $\mu_2 \circ \mu_1$ we have the following criterion: if $target_{\mu_1}(A) = \{A_1, \ldots, A_k\}$ and for all $i = 1, \ldots, k$ there exists $f_i \in \boldsymbol{f(ra_{\mu_2})}$ such that $A_i = f_i(A_1^i, \ldots, A_{k_i}^i)$ then

$$target_{\mu_2 \circ \mu_1}(A) = \bigcup_{i=1,\ldots,k} \{A_1^i, \ldots, A_{k_i}^i\}.$$

If $target_{\mu_1}(A) = \perp$ or if for some attribute A_i no derivation of the attribute using a function $f_i \in \boldsymbol{f(ra_{\mu_2})}$ is possible we have

$$target_{\mu_2 \circ \mu_1}(A) = \perp.$$

In order to ground the definition we assume that $target_{\epsilon}(A) = \{A\}$ and $\mu \circ \epsilon = \mu$ for the empty sequence of reformulation ϵ.

To determine the effects of multiple transformations $\mu_n \circ \ldots \circ \mu_1$ we have to evaluate $target_{\mu_n \circ \ldots \circ \mu_1}(A)$. This allows to determine which of the attributes required for evaluating a query containing attribute A are available after applying the reformulations $\mu_n \circ \ldots \circ \mu_1$. The definition of *target* is given such that it can be evaluated locally, i.e., for each reformulation step in an iterative manner. Using this information we can now define the syntactic similarity between a transformed query and its corresponding original query.

The decision on the importance of the attributes is query dependent. We have two issues to consider after applying a composite transformation $\mu = \mu_n \circ \ldots \circ \mu_1$:

1. Not all attributes in $\boldsymbol{sa}$ are preserved. Therefore, some of the atomic predicates in $\pi_{\boldsymbol{sa}}$ will not be correctly evaluated, i.e., the atomic predicates will simply be dropped in that case. Depending on the selectivity of the predicate, this might be harmful to different degrees. We capture this by calculating a value FV_i^{σ} for every attribute $A_i \in \boldsymbol{sa}$ as follows: if $A_i \in \boldsymbol{sa}$ and $target_{\mu}(A_i) \neq \perp$ then $FV_i^{\sigma} = 1$ else $FV_i^{\sigma} = sel_{A_i}$, where sel_{A_i} is the selectivity of the predicated $pred_{A_i}$. The selectivity is ranging over the interval $[0, 1]$, with low values indicating highly selective predicates, i.e., predicates selecting a small proportion of the database. In that way, dropping highly selective, critical attributes leads to lower values of FV_i^{σ}.

2. Not all attributes in $\boldsymbol{pa}$ are preserved. Therefore, some of the results may be incomplete or even erroneous (due to the loss of key attributes, for example). Following the method used above for the selection, we capture this by calculating a value FV_i^{π} for every attribute $A_i \in \boldsymbol{pa}$ as follows: if $A_i \in \boldsymbol{pa}$ and $target_{\mu}(A_i) \neq \perp$ then $FV_i^{\pi} = 1$ else $FV_i^{\pi} = 0$.

Given the values $FV_0^{\sigma} \ldots FV_k^{\sigma}$ for $A_0 \ldots A_k \in \boldsymbol{sa}$, we introduce a feature vector $\boldsymbol{FV^{\sigma}}$ capturing the syntactic effects for the reformulated

query $(\mu_n \circ \ldots \circ \mu_1)(q)$.

$$\boldsymbol{FV^{\sigma}}((\mu_n \circ \ldots \circ \mu_1)(q)) = (FV_1^{\sigma}, \ldots, FV_k^{\sigma}).$$

Using this feature vector, we define a syntactic similarity measure with respect to the selection including a user-defined weight vector $\boldsymbol{W} = (W_1, \ldots, W_k)$ pondering the importance of the attributes as:

$$SIM_{\sigma}(q, (\mu_n \circ \ldots \circ \mu_1)(q)) = \frac{\boldsymbol{W} \cdot \boldsymbol{FV^{\sigma}}}{|\boldsymbol{W}| \; |\mathbf{1}_k|}$$

where

$$\boldsymbol{W} \cdot \boldsymbol{FV^{\sigma}} = W_1 FV_1^{\sigma} + \ldots + W_k FV_k^{\sigma}$$

and

$$|\boldsymbol{X}| = \|\boldsymbol{X}\|_2 = \sqrt{x_1^2 + \ldots + x_k^2}$$

and $\mathbf{1}_k$ is a k-dimensional unit vector. This value is normalized on the interval $[0, 1]$. Originally, the similarity will be one, and it will decrease proportionally to the relative weight and selectivity of every attribute lost in the selection predicates.

For projections, analogous similarity measures SIM_{π} are derived based on feature vectors $\boldsymbol{FV^{\pi}}$. As for the selection similarity, the projection similarity decreases with the number of mappings applied to the query, until it reaches 0 when all the projection attributes are lost.

We illustrate the concepts introduced for the syntactic similarity by means of a small example. Assume a peer p_1 is connected to peers p_2 and p_3 as illustrated in Figure 4.4. Mappings in Figure 4.4 are given as simple attribute renamings, e.g.,

$$\mu_{p_1 \to p_3}(S_{p_3}) = \rho_{A_3:=C_1, B_3:=B_1, C_3:=A_1} \pi_{A_3, B_3, C_3}(S_{p_3}).$$

p_1 considers a query

$$q = \pi_{A_1, B_1, C_1}(DB_1)$$

and reformulates it for its two neighbors. It evaluates $\boldsymbol{FV^{\pi}}(\mu_{p_1 \to p_2}(q))$ as follows:

$$target_{\mu_{p_1 \to p_2}}(A_1) = \{A_2\}$$
$$target_{\mu_{p_1 \to p_2}}(B_1) = \perp_1$$
$$target_{\mu_{p_1 \to p_2}}(C_1) = \perp_2.$$

Therefore,

$$\boldsymbol{FV^{\pi}}(\mu_{p_1 \to p_2}(q)(S_{p_2})) = (1, 0, 0)$$

and

$$SIM_{\pi}(q, \mu_{p_1 \to p_2}(q)) = 1/3,$$

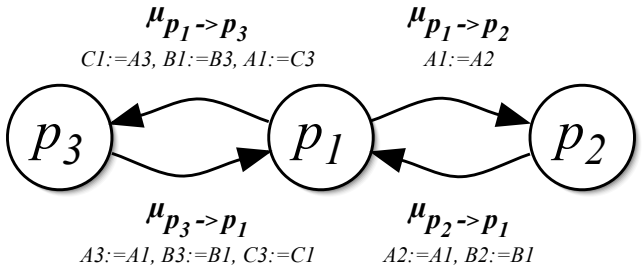

Figure 4.4: An example for syntactic similarity.

assuming all user-defined weights are equal to one. If p_2 sends the query back to p_1 through $\mu_{p_2 \to p_1}$, the similarity still equals to

$$SIM_\pi(q, (\mu_{p_2 \to p_1} \circ \mu_{p_1 \to p_2})(q)) = 1/3,$$

since only attribute A_1 remains intact after the two composite mappings.

On the other hand,

$$\begin{aligned} \text{target}_{\mu_{p_1 \to p_3}}(A_1) &= \{C_3\} \\ \text{target}_{\mu_{p_1 \to p_3}}(B_1) &= \{B_3\} \\ \text{target}_{\mu_{p_1 \to p_3}}(C_1) &= \{A_3\}. \end{aligned}$$

Thus,

$$\boldsymbol{FV}^{\boldsymbol{\pi}}(\mu_{p_1 \to p_3}(q)(S_3)) = (1, 1, 1)$$

and

$$SIM_\pi(q, \mu_{p_1 \to p_3}(q)) = 1.$$

If p_3 decides to send the query back to p_1, one would still obtain

$$SIM_\pi(q, (\mu_{p_3 \to p_1} \circ \mu_{p_1 \to p_3})(q)) = 1.$$

The fact that an obvious mistake occurs, i.e., that attribute $C3$ is wrongly mapped onto $A1$ in mapping $\mu_{p_1 \to p_3}$, is not detected by the syntactic similarity measure, but will be handled by the semantic similarity measures introduced in the following section.

4.5 Semantic Similarity

The context-independent measure of syntactic similarity is based on the assumption that the query reformulations are semantically correct, i.e., sound, which might not be the case for various reasons (see Section 4.1). As introduced earlier, we consider semantics as an agreement among peers. If two peers agree on the meaning of their schemas, then they will generate compatible mappings. From that basic observation, we now derive

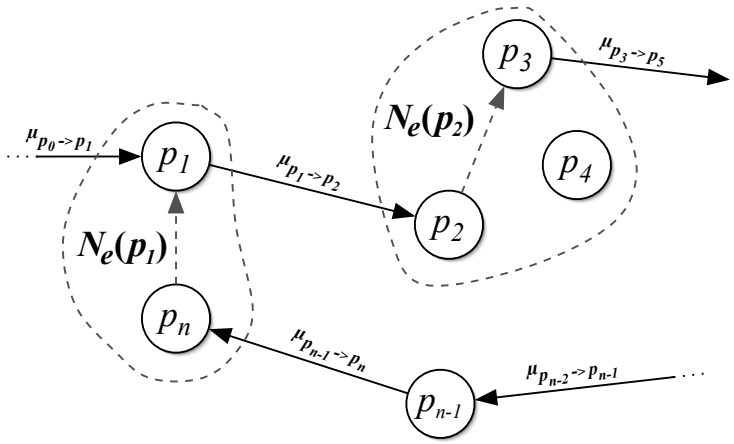

Figure 4.5: A semantic cycle: a query, sent out of $N_e(p_1)$ by peer p_1, returns to $N_e(p_1)$ through peer p_n.

context-dependent measures of semantic similarity related to the soundness of the reformulations for the attributes that are preserved in the mappings.

To that end, we introduce two mechanisms for deriving the soundness of a mapping. One mechanism is based on analyzing the fidelity of mappings at the schema level, the other one is based on analyzing the quality of the correspondences in the query results obtained at the data level.

4.5.1 Cycle Analysis

For the first mechanism, we exploit cycles in the semantic graph. Figure 4.5 shows an example of a semantic cycle. It starts with a peer p_1 transmitting a query q_1 to a peer p_2 through a mapping $\mu_{p_1 \to p_2}$.

In the example, after a few hops, the query is finally sent to a peer p_n, which, sharing the same schema as p_1, detects a cycle and informs p_1 (see Section 4.6 for more details on cycle detections). The returning query q_n is of the form

$$q_n = (\mu_{p_{n-1} \to p_n} \circ \ldots \circ \mu_{p_3 \to p_5} \circ \mu_{p_1 \to p_2})(q_1) = T(q_1).$$

p_1 may now analyze what happened to the attributes $A_1 \ldots A_k$ originally present in q_1:

- Case 1: $target_T(A_i) \equiv \{A_i\}$, this means that A_i has been maintained throughout the cycle. It usually indicates that all the peers along the cycle agree on the meaning of the attribute. Such an observation increases the confidence in the soundness of the attribute mappings used.

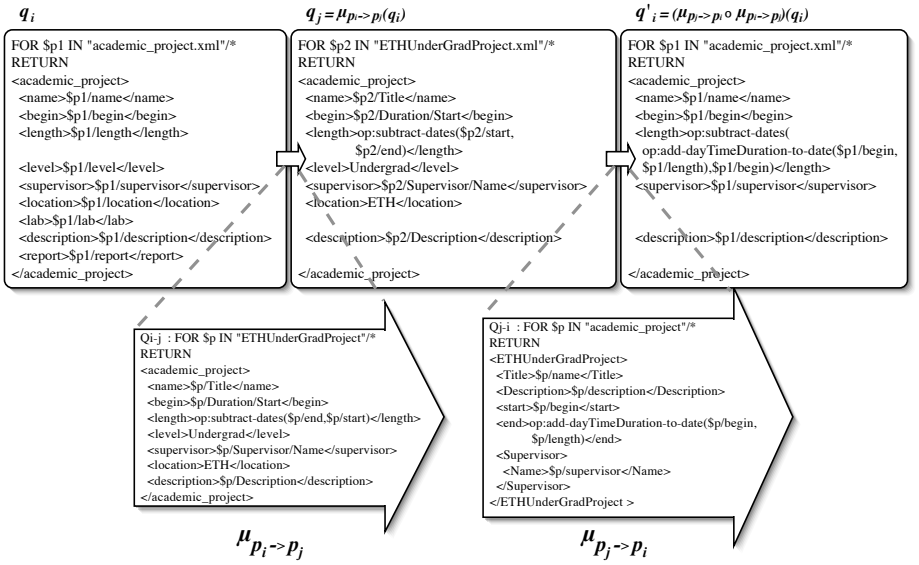

Figure 4.6: Cycle analysis: comparing a query to a reformulated query coming back to the same semantic domain.

- Case 2: $target_T(A_i) = \perp$, this means that someone along the cycle had no representation for A_i. A_i is not part of the common semantics. This leaves the confidence in the mappings unchanged.

- Case 3: Otherwise, if none of the two previous cases occurs, e.g., $target_T(A_i) = \{A_j\}, j \neq i$, this indicates some semantic confusion along the cycle. Subcases can occur depending on what happens to A_j. This lowers the confidence in the mappings.

In the following, we consider simple equality checks on the attributes and test whether $target_T(A_i) = \{A_i\}$ but note that these tests can be more elaborated in general (see for example Chapter 6 where we consider tests relative to subsumption relations). Checking whether or not the composed mapping transformation is identity might be difficult in general when considering syntactic transformations in the renamings: Figure 4.6 depicts a cycle of reformulations for the two schemas already described in Figure 4.1; the identity checks are simple, except for the attribute mapping m_{length}, where *op:subtract-dates(op:add-dayTimeDuration-to-date($p1/begin, $p1/length),$p1/begin)* is actually equivalent to *$p1/length*. Many of these test are however easy to handle at the syntactic level (e.g., checking that 26 days is equal to 26 days, see also Section 4.6).

We now derive heuristics for p_1 to assess the soundness of each attribute mappings m_i in $\mu_{p_1 \to p_2}$, based on the different cycle messages it receives. Let us consider some feedback $f_{\circlearrowleft}$ corresponding to the analysis

of a cycle composed of $\|f_{\circlearrowleft}\|$ schema mappings and starting with $\mu_{p_1 \to p_2}$. On an attribute basis, $f_{\circlearrowleft}$ may result in *positive* feedback $f^+_{\circlearrowleft}$ (case 1 above), *neutral* feedback (case 2, not used for the rest of this analysis but taken into account by the syntactic similarity), or *negative* feedback $f^-_{\circlearrowleft}$ (case 3). We model individual attribute mappings m_i in $\mu_{p_1 \to p_2}$ as independent Bernoulli variables; we write $P(m_i = 1)$ for the probability of a given attribute mapping to be sound and $P(m_i = 0)$ for the probability of an attribute mapping to be unsound. We denote by ϵ_{cyc} the probability of a foreign mapping (i.e., non-local as in $\mu_{p_3 \to p_5} \dots \mu_{p_{n-1} \to p_n}$ for p_1 in Figure 4.5) along a cycle being wrong for the attribute in question. Considering those error probabilities as being independent and identically distributed random variables, the probability of not having a foreign mapping error along the cycle is

$$(1 - \epsilon_{cyc})^{\|f_{\circlearrowleft}\|-1}. \tag{4.1}$$

Moreover, *compensating errors*, i.e., series of independent unsound mappings resulting in a sound reformulation, may occur along the cycle of foreign links without being noticed by p_1, which only has the final compound result q_n at its disposal. Thus, assuming a local attribute mapping m to be sound and denoting by δ_{cyc} the probability of errors being compensated somehow, the probability of getting some positive feedback from a given cycle is

$$P(f^+_{\circlearrowleft}|m = 1) = (1 - \epsilon_{cyc})^{\|f_{\circlearrowleft}\|-1} + (1 - (1 - \epsilon_{cyc})^{\|f_{\circlearrowleft}\|-1})\delta_{cyc} \tag{4.2}$$

while, under the same assumptions, the probability of getting some negative feedback is

$$P(f^-_{\circlearrowleft}|m = 1) = (1 - (1 - \epsilon_{cyc})^{\|f_{\circlearrowleft}\|-1})(1 - \delta_{cyc}). \tag{4.3}$$

Similarly, if we assume m to be unsound, the probability of getting respectively negative and positive feedback for the attribute in question are

$$P(f^-_{\circlearrowleft}|m = 0) = (1 - \epsilon_{cyc})^{\|f_{\circlearrowleft}\|-1} + (1 - (1 - \epsilon_{cyc})^{\|f_{\circlearrowleft}\|-1})(1 - \delta_{cyc}) \tag{4.4}$$

and

$$P(f^+_{\circlearrowleft}|m = 0) = (1 - (1 - \epsilon_{cyc})^{\|f_{\circlearrowleft}\|-1})\delta_{cyc}. \tag{4.5}$$

Let us assume that a peer p_1 obtains a set of positive and negative feedback values $\boldsymbol{f}_{\circlearrowleft} = \{f_{\circlearrowleft 1}, \dots, f_{\circlearrowleft n}\}$ for a given attribute A and n cycles. Some of the cycles may be positive, i.e., $source_T(A) = \{A\}$, other negative. We denote by $\boldsymbol{f}_{\circlearrowleft}{}^+ \subseteq \boldsymbol{f}$ the set of positive cycles and by $\boldsymbol{f}_{\circlearrowleft}{}^- \subseteq \boldsymbol{f}$ the set of negative cycles and have $\boldsymbol{f}_{\circlearrowleft} = \boldsymbol{f}_{\circlearrowleft}{}^+ \cup \boldsymbol{f}_{\circlearrowleft}{}^-$.

Assuming that all cycles are independent (which is actually an oversimplification for a real mapping graph, as erroneous mappings often have

an impact on several cycles; see Chapter 6 for a discussion on cycle correlations in PDMSs), p_1 can now calculate the likelihood on the soundness of its local attribute mapping m by combining Equations 4.2 to 4.5 using Bayes' rule:

$$P(m=1|\boldsymbol{f}_{\circlearrowleft}) = K\ P(m=1) \prod_{f^+_{\circlearrowleft}\in\boldsymbol{f}^+_{\circlearrowleft}} P(f^+_{\circlearrowleft})^{-1}P(f^+_{\circlearrowleft}|m=1) \prod_{f^-_{\circlearrowleft}\in\boldsymbol{f}^-_{\circlearrowleft}} P(f^-_{\circlearrowleft})^{-1}P(f^-_{\circlearrowleft}|m=1) \quad (4.6)$$

and

$$P(m=0|\boldsymbol{f}_{\circlearrowleft}) = K\ P(m=0) \prod_{f^+_{\circlearrowleft}\in\boldsymbol{f}^+_{\circlearrowleft}} P(f^+_{\circlearrowleft})^{-1}P(f^+_{\circlearrowleft}|m=0) \prod_{f^-_{\circlearrowleft}\in\boldsymbol{f}^-_{\circlearrowleft}} P(f^-_{\circlearrowleft})^{-1}P(f^-_{\circlearrowleft}|m=0) \quad (4.7)$$

where K is a normalizing constant ensuring that

$$P(m=1|\boldsymbol{f}_{\circlearrowleft}) + P(m=0|\boldsymbol{f}_{\circlearrowleft}) = 1.$$

Assuming that we have no prior knowledge on m and f, we set (principle of maximum entropy) $P(m=0) = P(m=1) = P(f^+_{\circlearrowleft}) = P(f^-_{\circlearrowleft}) = 1/2$. Those assumptions will be revisited in Chapter 6, where we take into account prior distributions on the soundness of the mappings and compute $P(f^+_{\circlearrowleft})$ and $P(f^-_{\circlearrowleft})$ by marginalization.

Since we have no knowledge about ϵ_{cyc} and δ_{cyc}, we assume these probabilities to be uniformly distributed (we give methods to get more accurate estimates for both values in Chapter 5). We integrate over ϵ_{cyc} and δ_{cyc} in order to obtain the expected posterior probability on the soundness of the attribute mappings. We can take into account density functions here if we have any *a priori* knowledge about those two random variables. The resulting expectation value for the soundness of the mapping is derived from Equation 4.6:

$$\gamma^{\circlearrowleft m_i} = \int_0^1 \int_0^1 P(m_i=1|\boldsymbol{f}_{\circlearrowleft})\,\mathrm{d}\epsilon_{cyc}\,\mathrm{d}\delta_{cyc}.$$

If no relevant feedback is obtained for a particular attribute mapping m_i, we set by default $\gamma^{\circlearrowleft m_i} = 1$.

This analysis can be performed by any peer p_1 for every outgoing mapping $\mu_{p_1\to p_2}$ to a peer p_2 and every attribute mapping m_i independently, resulting in values $\gamma^{\circlearrowleft m_i}_{\mu_{p_1\to p_2}}$ indicating the probability of the attribute mapping m_i in mapping $\mu_{p_1\to p_2}$ being sound given the feedback received from various mapping cycles.

As for the preceding section, we define a feature vector and a similarity measure to capture the semantic losses along a sequence of mappings $\mu_1,\ldots,\mu_n$, where μ_j connects peer p_j to peer p_{j+1} via a mapping.

Let us suppose that peer p_1 reformulates a query

$$q_1(S_{p_1}) = \pi_{\boldsymbol{pa}}(\sigma_{\boldsymbol{pred(sa)}}(S_1))$$

for p_2 through a mapping $\mu_1 = \mu_{p_1 \to p_2}$. p_1 computes a feature vector for q_1 based on the cycle messages it has received as follows:

$$\boldsymbol{FV}^{\circlearrowleft}(\mu_1(q)) = (FV_1^{\circlearrowleft}(A_1), \ldots, FV_k^{\circlearrowleft}(A_k))$$

where feature values $FV_i(A_i)$ correspond to the posterior probability $\gamma^{\circlearrowleft m_i}$ of an attribute $A_i \in \boldsymbol{pa} \cup \boldsymbol{sa}$ to be mapped correctly through $\mu_1 = \mu_{p_1 \to p_2}$:

$$FV_i^{\circlearrowleft}(\mu_1(q)) = \gamma_{\mu_1}^{\circlearrowleft m_i}.$$

In the following reformulations, these values are updated by iteratively multiplying the values obtained for the degree of soundness for each mapping. We consider here that the soundness of the various mappings are independent, i.e., if two mappings μ_1 and μ_2 have degrees of soundness of $\gamma_{\mu_1}^{\circlearrowleft m_i}$ and $\gamma_{\mu_2}^{\circlearrowleft m_j}$ for attribute A_i and A_j with $target_{\mu_1}(A_i) = A_j$, the degree of soundness of the composite mapping $(\mu_2 \circ \mu_1)$ is $\gamma_{\mu_1}^{\circlearrowleft m_i} \gamma_{\mu_2}^{\circlearrowleft m_j}$. Note that other ways of combining the soundness values could be used (e.g., *never* use a mapping whose soundness probability is smaller than a given value, see Section 7.6). When forwarding a reformulated query using a mapping μ_j, peer p_j updates each value $FV_i^{\circlearrowleft}((\mu_{j-1} \circ \ldots \circ \mu_1)(q))$ it has received along with the transformed query $(\mu_{j-1} \circ \ldots \circ \mu_1)(q)$ in the following way:

$$FV_i^{\circlearrowleft}((\mu_j \circ \ldots \circ \mu_1)(q)) = FV_i^{\circlearrowleft}((\mu_{j-1} \circ \ldots \circ \mu_1)(q)) \prod_{A_k \in target_{(\mu_{j-1} \circ \ldots \circ \mu_1)}(A_i)} \gamma_{\mu_j}^{\circlearrowleft m_k}.$$

The associated semantic similarity between the original query q and the reformulated query $(\mu_j \circ \ldots \circ \mu_1)(q)$ is then

$$SIM_{\circlearrowleft}(q, (\mu_j \circ \ldots \circ \mu_1)(q)) = \frac{\boldsymbol{W} \cdot \boldsymbol{FV}^{\circlearrowleft}}{|\boldsymbol{W}|\ |\boldsymbol{1}_k|}.$$

This value starts at one (in the semantic domain which the query originates from) and decreases as the query traverses more and more semantically heterogeneous domains.

We illustrate the cycle analysis by means of the example given in Figure 4.4. Assume a query $q = \pi_{A_1,B_1,C_1}(S_1)$ is forwarded iteratively through mappings $\mu_{p_2 \to p_1} \circ \mu_{p_1 \to p_2}$ and $\mu_{p_3 \to p_1} \circ \mu_{p_1 \to p_3}$ respectively. p_1 analyzes the two returning queries:

$$q_{2 \to 1} = (\mu_{p_2 \to p_1} \circ \mu_{p_1 \to p_2})(q) = \pi_{A_1}(S_1)$$
$$q_{3 \to 1} = (\mu_{p_3 \to p_1} \circ \mu_{p_1 \to p_3})(q) = \pi_{C_1,B_1,A_1}(S_1).$$

p_1 receives $q_{2\to1}$ from p_2. A_1 is correctly mapped again onto A_1 in the returning query. Based on this unique positive feedback, p_1 computes the degree of soundness for $\gamma^{\circlearrowleft m_{A_1}}_{\mu_{p_1\to p_2}}$:

$$\gamma^{\circlearrowleft m_{A_1}}_{\mu_{p_1\to p_2}} = 3/4.$$

Since no feedback is obtained for the other attributes, p_1 sets

$$\gamma^{\circlearrowleft m_{B_1}}_{\mu_{p_1\to p_2}} = \gamma^{\circlearrowleft m_{C_1}}_{\mu_{p_1\to p_2}} = 1.$$

p_1 computes similar values for the second returning query $q_{3\to1}$ for mapping $\mu_{p_1\to p_3}$. For B_1, the feedback conveyed by the returning query is positive, whereas it is negative for A_1 and C_1:

$$\begin{aligned}\gamma^{\circlearrowleft m_{B_1}}_{\mu_{p_1\to p_3}} &= 3/4\\ \gamma^{\circlearrowleft m_{A_1}}_{\mu_{p_1\to p_3}} = \gamma^{\circlearrowleft m_{C_1}}_{\mu_{p_1\to p_3}} &= 1/4.\end{aligned}$$

Imagine now that p_1 decides to send a new query q' to p_2 and p_3, with

$$q' = \pi_{C_1}\sigma_{A_1='EPFL'}(S_1).$$

p_1 first reformulates the query into two transformed queries:

$$\begin{aligned}q'_{1\to2} &= \mu_{p_1\to p_2}(q') = \sigma_{A_2='EPFL'}(S_2)\\ q'_{1\to3} &= \mu_{p_1\to p_3}(q') = \pi_{A_3}\sigma_{C_3='EPFL'}(S_3).\end{aligned}$$

Based on the analyses performed previously for q, p_1 can now compute the feature vectors and the semantic similarity values for the newly reformulated queries:

$$\boldsymbol{FV}^{\circlearrowleft}(\mu_{p_1\to p_2}(q')) = (1, 3/4)$$
$$SIM_{\circlearrowleft}(q', \mu_{p_1\to p_2}(q')) = 0.875$$

and

$$\boldsymbol{FV}^{\circlearrowleft}(\mu_{p_1\to p_3}(q')) = (1/4, 1/4)$$
$$SIM_{\circlearrowleft}(q', \mu_{p_1\to p_3}(q')) = 0.25$$

assuming that all user weights are equal to one. The semantic analysis has thus correctly identified some semantic mismatch in the way A_1 and C_1 get mapped through $\mu_{p_1\to p_3}$ (low value for $SIM_{\circlearrowleft}(q', \mu_{p_1\to p_3}(q'))$). Depending on the situation, p_1 then decides whether to send this poorly reformulated query to p_3.

4.5.2 Result Analysis

The second mechanism for analyzing the semantic quality of the translations is based on the analysis of the results returned. In [ACMH03b], we introduced a method using functional dependencies at the data level in order to assess the quality of the mappings. The method was based on analyzing to which extent integrity constraints are preserved after reformulation.

We present below an alternative approach based on classification mechanisms. We assume that peers annotate documents $\mathcal{D}$ using meta-data expressed according to our data model. Thus, each document $d \in \mathcal{D}$ owned by peer p is associated with an annotation $annot(d)$ according to the schema S_p of the peer. Having sent a query, peers start to receive result documents with semantically rich content, e.g., images or full text. Based on this content, they attempt to assess to which extent the queries expressed at the meta-data level were properly reformulated and thus led other peers to return the correct result documents.

Queries in our meta-data model are thus an intensional way of expressing semantic concepts, whereas extensionally the concepts are related to sets of documents. The problem that we address is of how to arrive at annotation schemes and schema mappings at the intensional level that result in concept definitions that are compatible with the extensional notion of the concepts.

In the following, we assume that a peer has a finite set of classes $\mathcal{C}$ to classify documents. The extensional notion of a class is based on methods of content analysis. Here, we do not make any assumption about the methods (e.g., layout analysis, lexicographical analysis, contour-detection, etc., or even simple manual classification) used to extract meaningful features out of the documents; we simply treat them as high-level abstractions used to unambiguously classify any possible retrieved documents $d \in \mathcal{D}$ into classes $c \in \mathcal{C}$ using a decision rule $\mathcal{R}_{content}$:

$$\mathcal{R}_{content} : \mathcal{D} \rightarrow \mathcal{C}.$$

In a more general setting, $\mathcal{R}_{content}$ could be a probabilistic rule (see Chapter 9). Using their local classification based on content analysis, peers can thus determine for every received document the concept to which it belongs.

The intensional notion of classes is based on classification rules applied to the metadata annotations of the documents:

$$\mathcal{R}_{annot} : annot(\mathcal{D}) \rightarrow \mathcal{C}.$$

Again, we do not make assumptions on the specific form of the classification rules, except that they apply predicates to the metadata annotations and derive from those predicates the class associated to the document. Examples of classification rules are extensively discussed in

the data mining literature. The document classifications obtained from content analysis and by classification rules are presumed to be consistent up to a mean classification error ϵ_{res}, i.e., we assume that with a probability $1 - \epsilon_{res}$

$$\mathcal{R}_{content}(d) = \mathcal{R}_{annot}(annot(d)).$$

By analyzing its own document collection, a peer can locally estimate the value of ϵ_{res}.

Imagine now a peer p_1 classifying documents according to rules $\mathcal{R}^{p_1}_{content}$ and $\mathcal{R}^{p_1}_{annot}$. Peer p_1 issues a query q against its own schema for retrieving documents. Upon reception of a document d from a foreign peer $p_2 \in N_d(p_1)$, p_1 performs the classification operation according to its own rules $\mathcal{R}^{p_1}_{content}$ and $\mathcal{R}^{p_1}_{annot}$. Different situations can then occur:

- Case 1: $\mathcal{R}^{p_1}_{content}(d) = \mathcal{R}^{p_1}_{annot}(d)$: this is the result p_1 was expecting; it is an indication that the outgoing mapping used to forward q to p_2 was semantically correct for query q. We treat this as positive feedback ($f^{+}_{\rightleftarrows}$).

- Case 2: $\mathcal{R}^{p_1}_{content}(d) \neq \mathcal{R}^{p_1}_{annot}(d)$: p_1 receives a document whose content analysis does not match the classification obtained from the metadata annotation obtained by reformulation. Since the document content is not changed during transmission of the query result, this implies that some semantic confusion occurred in the metadata query reformulation along the path from p_1 to p_2. In that case, we consider this as negative feedback ($f^{-}_{\rightleftarrows}$).

If p_1 and p_2 are directly connected through a mapping, this gives us a clear indication about the semantic (un)soundness of the mapping $\mu_{p_1 \rightarrow p_2}$: given the mean classification error probability ϵ_{res}, the probability of an attribute mapping being sound in case of positive feedback is $1 - \epsilon_{res}$.

If the two peers are separated by one or more semantic domains, the situation is somewhat more complicated since we have to take into account all the successive mapping links used to forward the query from peer p_1 to a peer p_n. Let us suppose that a peer receives some feedback f after the query has gone through $\|f\|$ different mappings; analogously to the derivation of the probabilities for the cycle analysis (see Equation 4.2), the probability of receiving a positive feedback assuming the mapping being analyzed is sound is:

$$P(f^{+}_{\rightleftarrows}|m = 1) = (1 - \epsilon_{res})((1 - \epsilon_{cyc})^{\|f_{\rightleftarrows}\|-1} + (1 - (1 - \epsilon_{cyc})^{\|f_{\rightleftarrows}\|-1})\delta_{cyc}) + \epsilon_{res}\delta_{res}(1 - (1 - \epsilon_{cyc})^{\|f_{\rightleftarrows}\|-1})(1 - \delta_{cyc}).$$

The first term covers the case where the peer performs a proper classification on a result obtained from a proper query reformulation (see Equation 4.2). The situation where the transitive closure of the mappings is

erroneous (see Equation 4.3) and the peer still believes it has obtained a positive feedback is more intricate and is covered by the second term. Receiving an erroneously annotated result, a peer can still perform a misclassification with probability ϵ_{res}. However, only in exceptional cases with probability δ_{res} will this misclassification correct the improper reformulation of the query, namely when the "wrong" classification matches exactly the content of the improper reformulation (third term). A peer can estimate the probability δ_{res} by $(\|\mathcal{C}\|-1)^{-1}$, where $\|\mathcal{C}\|$ is the number of different classes used by the peer. The probability of receiving negative feedback is then calculated analogously.

Performing an analysis analogous to the one given in Section 4.5.1, we compute the posterior probability of a local attribute mapping being sound given some positive $\boldsymbol{f}^{+}_{\rightleftarrows}$ and negative $\boldsymbol{f}^{-}_{\rightleftarrows}$ feedback results

$$P(m=1|\boldsymbol{f}_{\rightleftarrows}) = K\ P(m=1) \prod_{f^{+}_{\rightleftarrows}\in\boldsymbol{f}^{+}_{\rightleftarrows}} P(f^{+}_{\rightleftarrows})^{-1}P(f^{+}_{\rightleftarrows}|m=1) \prod_{f^{-}_{\rightleftarrows}\in\boldsymbol{f}^{-}_{\rightleftarrows}} P(f^{-}_{\rightleftarrows})^{-1}P(f^{-}_{\rightleftarrows}|m=1)$$

and the expected value for the soundness of the attribute mapping

$$\gamma^{\rightleftarrows m_i} = \int_0^1 \int_0^1 P(m=1|\boldsymbol{f}_{\rightleftarrows})\,\mathrm{d}\epsilon_{cyc}\,\mathrm{d}\delta_{cyc}.$$

If no relevant feedback result is obtained for a particular attribute mapping m_i, we set by default $\gamma^{\rightleftarrows m_i} = 1$. The corresponding feature values are

$$FV_i^{\rightleftarrows}(\mu(q)) = \gamma_{\mu}^{\rightleftarrows m_i}.$$

Analogous to the cycle analysis, these values are forwarded and updated iteratively along with the query by multiplying the values obtained for each mapping, such that a measure for the semantic similarity between an original query and a reformulated query based on results analyses follows similarly

$$SIM_{\rightleftarrows}(q,(\mu_j\circ\ldots\circ\mu_1)(q)) = \frac{\boldsymbol{W}\cdot\boldsymbol{FV}^{\rightleftarrows}}{|\boldsymbol{W}|\ |\boldsymbol{1}_k|}.$$

Some illustrating examples of the result analysis are given in Chapter 5.

4.6 Gossiping Algorithm

We now devise an algorithm to route queries throughout the P2P network based on syntactic and semantic criteria. At this point, we have four measures (SIM_{σ}, SIM_{π}, $SIM_{\circlearrowleft}$ and $SIM_{\rightleftarrows}$) for evaluating the losses incurred through the reformulations. We take advantage of those values to decide whether or not it is worth forwarding a specific query to a

foreign semantic domain. Queries can be issued to any peer through a query message. The basic query message format is

$$query(q_0, p_0, q_{last}, p_{last}, \boldsymbol{RT}, \boldsymbol{target_{q_0,q_{last}}}).$$

A query message contains the original query q_0 issued by the query originator p_0, the latest reformulation q_{last} of the query forwarded by peer p_{last}, a reformulation trace $\boldsymbol{RT}$ to keep track of the reformulations already performed, and the source-to-target dependencies between the attributes in q_0 in and in q_{last}. The reformulation trace $\boldsymbol{RT}$ is a list of triples $\{(p_\mu, S_{p_\mu}, \mu)$ keeping track of the peers p_μ with schema S_{p_μ} having already reformulated the query through a schema mapping μ.

Furthermore, we require the query originator p_0 to attach a few user-defined or generated values to the query it issues:

- the weights $\boldsymbol{W}$ pondering the importance of the attributes in the query
- the respective selectivity $\boldsymbol{sel}$ of the selection attributes $\boldsymbol{sa}$
- the minimal threshold values $\boldsymbol{SIM}_{min} = (SIM_\sigma^{min}, SIM_\pi^{min}, SIM_{\circlearrowleft}^{min}, SIM_{\rightleftarrows}^{min})$ for the similarity measures under which a transformed query is so deteriorated that it can no longer be considered as being equivalent to the original query.

Those values can be used to introduce a tradeoff between the precision and recall of the result set: peers can ask for precise answers by setting the similarity thresholds to high values. In that way, they ensure that their queries get only propagated to the peers that most probably can process them correctly. Setting SIM_σ^{min} and SIM_π^{min} to one ensures that only complete reformulations get forwarded through the PDMS. Setting SIM_σ^{min} and SIM_π^{min} to high values ensures that only the most probably sound reformulations get disseminated. Under different circumstances, some peers might prefer low similarity thresholds to favor recall and eliminate all inappropriate results locally *a posteriori*. Local filtering of improper results can be facilitated by ranking the results according to the similarity values of the query through which they were obtained. We extend the format of a query message to include the values we have just discussed as well as the iteratively updated feature vectors:

$$query(q_0, p_0, q_{last}, p_{last}, \boldsymbol{RT}, \boldsymbol{target_{q_0,q_{last}}}, \boldsymbol{W}, \boldsymbol{sel}, \boldsymbol{SIM_{min}}, \boldsymbol{FV_\sigma}, \boldsymbol{FV_\pi}, \boldsymbol{FV_{\circlearrowleft}}, \boldsymbol{FV_{\rightleftarrows}}).$$

Now, upon reception of a query message, we require a peer p to perform a series of tasks:

1. detect any semantic cycles by searching for the local schema S_p in the list of schemas S_{p_μ} in the reformulation trace. If a cycle is detected, p forwards the current query message $query()$ to p_μ for inspection.
2. check whether or not the query has already been received; discard the query if it has already been processed
3. forward the query to the local neighborhood $N_e(p)$
4. process the query and return potential results to the query originator p_0

and, for each of its outgoing mappings:

5. apply the reformulation to the query and update the similarity measures for the reformulated query
6. forward the query using the mapping if all similarity values are greater or equal to the thresholds given by $\boldsymbol{SIM_{min}}$.

Additionally, peers are expected to perform semantic analyses when they receive results or cycle messages. The full gossiping algorithm is given in Algorithm 4.1.

Note that the gossiping algorithm presented above piggybacks on query forwarding to detect cycles. Alternatively, peers can send special probe queries with a certain TTL periodically to detect semantic cycles. Probe queries can be created as selection-free queries projecting on all attributes of a give schema (see Figure 4.6, where q_i is a probe query for the left-hand side schema of Figure 4.1).

4.7 Case Study

To illustrate how to apply the abstract model detailed above in a concrete setting, we now describe one of the experiments we conducted in order to realize Semantic Gossiping in an XML/XQuery environment. Seven people were first asked to design a simple XML document containing some project meta-data. The outcome of this deliberately imprecise task definition was a collection of structured documents lacking common semantics though overlapping partially for a subset of the embraced meta-data (e.g., *name of the project* or *start date*). Figure 4.1 shows two of those documents. Viewing the documents as seven distinct semantic domains in a decentralized setting, we then randomly produced a graph connecting the different domains.

In the next step of the experiment, we asked the authors to create mappings in XQuery for every link departing from their domain. Finally,

Algorithm 4.1 Semantic Gossiping

```
if q.p_last == q.RT.p_μ.last then
  /*newly reformulated query; search for semantic cycles*/
  for all p in q.RT.p_μ such that q.RT.S_pμ == mySchema do
    send cycleMesssage(q) to p;
  end for
end if
if q in ProcessedQueries then
  /*this query has already been processed*/
  break;
end if
for all p in SemanticNeighborhood do
  /*forward the query throughout the neighborhood*/
  send queryMessage(q) to p;
end for
for all μ_{p1→p2} in LocalMappings do
  /*reformulate the query*/
  q' = μ_{p1→p2}(q);
  if all q'.SIM ≥ q'.SIM_min then
    send queryMessage(q') to p2;
  end if
end for
/*process the query*/
results = process(q);
send results to p0
```

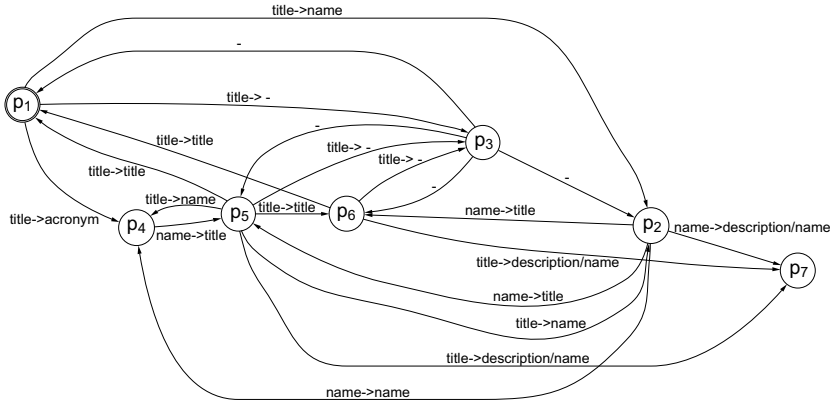

Figure 4.7: A semantic graph of reformulations, where attribute mappings are depicted for the attribute *Title*.

using the IPSI-XQ[1] XQuery libraries and the Xerces[2] XML parser, we built a query reformulator capable of handling and forwarding the queries following the Gossiping Algorithm. The resulting topology is depicted in Figure 4.7. In this figure, we also provide an example of how a particular attribute gets mapped: all the domains have some representation for the title of the project (usually referred to as *name* or *title*, see Figure 4.7 where the reformulations for the attribute *title* are represented on top of the links), except p_3, which only considers a simple ID for identifying the projects.

Let us assume that a single-attribute query $q = \pi_{title}(S_1)$ is issued by p_1 to obtain all the titles of the different projects. In XQuery, q can be written in the following way:

```
Query =
     FOR $project
     IN "project_P1.xml"/*
     RETURN <title>$project/title</title>
```

Let us now determine how the query gets propagated from p_1 with all thresholds $\boldsymbol{SIM}_{min}$ set to 1/2. Note that the weights attached to the query do not matter here, as the query contains a single projection attribute. Moreover, we do not need to consider SIM_σ as it always evaluates to one for this projection query.

Following the gossiping algorithm, p_1 first attempts to transmit the query to its direct neighbors, i.e., p_2, p_3 and p_4. p_2 and p_4 in turn forward

[1] *http://ipsi.fhg.de/oasys/projects/ipsi-xq/*
[2] *http://xml.apache.org/xerces2-j/index.html*

the query to the other nodes, but p_3 will in fact never receive the query: As p_3 has no representation for the *title*, the only projection attribute would be lost in the translation process from p_1 to p_3, lowering SIM_π to zero.

Let us now examine the semantic similarity $SIM_\circlearrowleft$. For the topology considered, thirty-one semantic cycles could be detected by p_1 in the best case. As the query never traverses p_3, only eight cycles remain relevant to our analysis (see Table 4.1, which lists those cycles). For its first outgoing mapping link (i.e., the link going from p_1 to p_2), p_1 receives five positive cycles, raising the semantic similarity measure for this link and the attribute considered to 0.79.[3] p_1 does not receive any semantically significant feedback for its second outgoing link $T_{p_1 \to p_3}$, which is anyway handled by the syntactic analysis. Yet, it receives three negative cycle messages for its last outgoing link $\mu_{p_1 \to p_4}$. This link is clearly semantically erroneous, mapping *title* onto *acronym*. This results in p_1 excluding the link for forwarding the query, since the semantic similarity drops to 0.27 in that case.

Table 4.1: Cycles resulting in positive (+) or negative (-) feedback for the graph of Figure 4.7.

Cycle	$\mu_{p_1 \to p_4}$ *wrong*	$\mu_{p_2 \to p_4}$ *wrong*
p_1, p_2, p_4, p_5, p_1	+	-
$p_1, p_2, p_4, p_5, p_6, p_1$	+	-
p_1, p_2, p_5, p_1	+	+
p_1, p_2, p_5, p_6, p_1	+	+
p_1, p_2, p_6, p_1	+	+
p_1, p_4, p_5, p_1	-	+
$p_1, p_4, p_5, p_2, p_6, p_1$	-	+
p_1, p_4, p_5, p_6, p_1	-	+

The situation may be summarized as follows: p_1 restrains from sending the query through p_3 because of the syntactic analysis (too much information is lost in the translation process) and excludes p_2 because of the high semantic dissimilarity. The situation somewhat changes if we correct the erroneous link $\mu_{p_1 \to p_4}$ and introduce a faulty mapping for the link $\mu_{p_2 \to p_4}$. For the attribute considered, the semantic similarity drops to 0.69 for the outgoing link $\mu_{p_1 \to p_2}$ (two long cycles are negative, see third column in Table 4.1). Even though it is not directly connected to an erroneous link, p_1 senses the semantic incompatibilities affecting some of the messages

[3] Remember that we did not make any assumption regarding the distribution of erroneous links. In this case, the positive feedback received may as well come from a series of compensating errors.

traversing p_2.

4.8 Related Work

GLAV query reformulation in PDMS was studied in the context of the Piazza [HIMT03] project. Piazza defines a PDMS as a collection of *peer* and *stored* relations interlinked with GLAV mappings. Peer mappings can take the form of inclusion (containment) or equivalence mappings and provide semantic glue between the schemas of different peers. Storage descriptions link the stored relations to the peer relations, typically through containment mappings to indicate that the local databases are actually incomplete (open-world semantics). The problem of finding all certain answers to a conjunctive query in a PDMS with GLAV mappings is undecidable in general. However, it can become polynomial or $co - NP$ complete when inclusion mappings are acyclic (see the Piazza papers [HIST03, HIST05] for details). The problem of composing GLAV mappings in a PDMS is decidable [MH03a] for a large class of conjunctive queries (i.e., for CQ_k queries defined as non-recursive datalog programs with at most k variables in each rule and containing a single definition for each predicate).

Hyperion [AKK+03] is a project inspired by the Local Relational Model [BGK+02] using mapping tables and coordination rules to share data in decentralized environments. Mapping table [MH03b] are instance-level mappings defining how data from different sources can be associated. Specific algorithms [KA04] can be used to compute both sound and complete reformulations of a local query using mapping tables. SomeWhere [ACG+06] is a distributed inference systems for RDF/S data. It implements a decentralized algorithm for consequence finding of a clause with respect to a set of distributed propositional theories. Semantic mappings based on epistemic logic were proposed by Calvanese *et. al.* [CGLR04] to preserve decidability of query reformulation with GLAV mappings.

Semantic routing was recently discussed in various contexts: Loeser *et al.* [LST05] direct semantic searches based on peers that have answered or issued similar queries in the past. SQPeer [KC04] proposes a publish-subscribe mechanism *à la* Edutella and an algorithm relying on query/view subsumption techniques to produce routing information in decentralized Semantic Web environments. Zhuge *et al.* [ZLF+05] use structural similarity between pairs of schemas to route queries in P2P semantic networks.

4.9 Conclusions

Semantic gossiping introduces a radically new way of querying data in a decentralized setting, by considering both incomplete and unsound query reformulations. Incomplete query reformulations originate from mappings that only consider partial translations of a given schema, and are handled by an iterative analysis of the transformed query at a syntactic level. Unsound query reformulations stem from inaccurate or erroneous schema mappings relating semantically divergent attributes. Viewing semantics as an agreement, we detect semantic mismatch by analyzing transitive closures of mapping operations and by examining low-level attributes of results received from distant peers. Taking advantage of both syntactic and semantic analyses, we introduce a tradeoff between the precision and recall of the set of possible answers to a query.

Our framework is based on a decentralized version of the mediator architecture, where arbitrary peers autonomously provide mappings between pairs of semantically related schemas. We model schema mappings as GAV-like queries and reformulate and route queries in a totally decentralized fashion. One can easily extend our techniques to handle conjunctive queries and peers with multiple local relations or schemas: local joins can be supported by introducing a natural join operator $\bowtie$ that can be handled like the renamings in the reformulations. As for the renamings, the join operations can incur a higher number of atoms that have to be handled in the subsequent reformulations. The reformulations will however always remain finite as we still only consider series of GAV mappings [MH03a]. Extending our framework to more general GLAV mappings is also possible, though slightly more complex; the composition of GLAV mappings can in general be infinite, but can be precisely computed for conjunctive queries in which nested expressions have at most k variables (CQ_k queries, see the work by Madhavan and Halevy [MH03a]). Also, query reformulation would in that case require a non-standard query processor to reformulate LAV queries. Moreover, we believe that LAV loses quite a bit of its attractiveness in a PDMS setting: LAV was primarily conceived to offer greater scalability to the mediator, which disappears in our setting. In a PDMS, LAV mappings would only be useful when one has to define a surjection from the source attributes to the target attributes (i.e., when mapping a function of two or more source attributes onto one target attribute), which is pretty uncommon in the setting we consider (see Section 2.2 or concrete examples of mappings in Section 6.5.3).

Chapter 5

Self-Repairing Semantic Networks

In the preceding chapter, we presented a model for PDMSs and several methods to analyze syntactic and semantic losses when reformulating a query through a mapping. Those methods were used to forward queries in a self-organizing way, where peers collaborate to disseminate a query in a network of loosely-coupled and heterogeneous information parties. In this chapter, we take our approach one step further: rather than only guiding searches by the results obtained from analyzing the reformulations, we also take advantage of the network feedback to modify the mappings in an automatic manner. Thus, we make a step towards self-learning networks of peers collaboratively establishing semantic interoperability in an automated way [ACMH03a]. We give experimental results that demonstrate how the different kinds of semantic analyses of reformulations interact with the modification of potentially unsound mappings. The initial results interpreted below provide promising evidence that Semantic Gossiping can be used to automatically reach semantic agreement in large networks of computer-generated and dynamic mapping links. In particular, they indicate in which cases each of the two semantic similarity measures derived from cycle and result analysis are more suitable, and how our approach scales with different parameters.

5.1 Experimental setup

The setup we used in the experiments is as follows: we assume a network of heterogeneous peers representing each an individual semantic domain. Peers share a finite set of semantically similar concepts, i.e., operate in a certain semantic domain (for example, biological databases) inside the network. They share annotated documents (or data) related to those concepts, but refer to concepts using different attributes (they denominate the concepts differently). From this basic setup, we attempt to create

global interoperability by applying Semantic Gossiping techniques using purely pair-wise, local schema mappings.

The exact description of the process is as follows: first, we create a topology of n peers $p_1 \dots p_n$, each of them connected through mappings links to l other peers. The peers share $\|\mathcal{C}\|$ concepts $c_1 \dots c_{\|\mathcal{C}\|}$, but use distinct attributes (i.e., names) to refer to them. Thus, we study the problem of peers sharing the same concepts but lacking knowledge of how to refer to them by names. Without loss of generality, we assume that the same set of attributes $A_1 \dots A_{\|\mathcal{C}\|}$ is used by all peers (this simplifies the subsequent presentation). We write $(A_i \mapsto_p c_k)$ if peer p uses attribute A_i to refer to concept c_k. In that way, we can use a single attribute A to store the name the peer associates to a concept. Also, peers can verify whether a document belongs to a concept or not and thus annotate documents they store with attributes A_i .

We then generate mappings $\mu_{p_1 \to p_2}$ for every mapping link between two peers p_1 and p_2. The mapping function μ relates attributes from the first peer to attributes from the second peer, with every attribute used by the first peer mapped onto some attribute used by the second peer by a renaming ρ. Hence, μ is a permutation of the domain of the attributes. We write every attribute mapping $m_{p_1 \to p_2} \in \mu_{p_1 \to p_2}$ as $m_{p_1 \to p_2}(A_i) = A_j$ to indicate that m maps attribute A_i used by p_1 onto attribute A_j used by p_2. For every attribute mapping m, the mapping is sound if and only if the two attributes bounded by the attribute mapping actually refer to the same concept, that is if

$$m_{p_1 \to p_2}(A_i) = A_j \wedge A_i \mapsto_{p1} c_k \wedge A_j \mapsto_{p2} c_k.$$

Thus, random schema mappings only have a probability of $\frac{1}{\|\mathcal{C}\|!}$ of being sound in our setting. In the following experiments, we generate a fraction *eRate* of erroneous attribute mappings initially. In the end, the mappings generated in that way are quite similar to the real mappings generated by state-of-the-art automatic schema matchers: they correctly map most attributes from p_2 onto semantically similar attributes from p_1, while producing a fraction of unsound attribute mappings for a subset of the attributes they misinterpret (see the following chapter for concrete examples).

Unless specified otherwise, we use small-world graphs [WS98] to interconnect peers with mapping links since small-world topologies have been extensively applied to model computer networks or social behaviors. They are typically characterized by high clustering coefficients (average fraction of pairs of neighbors of a node that are also neighbors of each other) and relatively small path length (average minimal distance between two nodes). In the following, we generate graphs with an average clustering coefficient of 0.1 and with 10% of shortcuts (i.e., links rewired to a random peer in the network).

Starting from that original setting, we apply Semantic Gossiping techniques in order to detect and rectify erroneous mappings iteratively. At every experimental step, each peer issues a generic probe query (i.e., the query consists of a projection on all its local attributes, see Section 4.6) through its mapping links. The query is propagated to the other peers (semantic domains) in a Gnutella-like fashion with a low TTL value.

For detecting and repairing potentially erroneous mappings, we slightly modify the semantic analyses; we forward the probe queries irrespectively of their similarity values in order to get as many evidences as possible, and use the network feedback to reach semantic agreements by gradually modifying the mappings.

Before taking a closer look at the final results, we evaluate in the following sections each of the semantic analyses (cycle and result analysis) separately to emphasize their specificities.

5.2 Cycle Analysis

For every iteration step, each peer randomly selects one of its local attributes, sends a probe query and analyzes the cycle messages it gets in return. Here, we do not only estimate the soundness of the actual attribute mappings as explained in Section 4.5.1, but also determine which of the potential attribute mappings is most likely correct and adopt it as a new mapping. Therefore, peers view mappings resulting from returned queries as new mapping candidates. Consider for example Figure 5.1, where peer p_1 systematically receives A_1 mapped onto A_2 in returning queries (negative feedback). In addition to evaluating the correctness of the current mapping, p_1 considers other potential mappings as well, by permuting the target attribute of individual attribute mappings in the schema mapping. It adopts the most probably sound mapping candidate if its probability of being sound is above 50%. In the example of Figure 5.1, p_1 evaluates the soundness of the attribute mapping m_1 mapping A_1 onto A_2, and might consider to modify it into a new candidate mapping m' mapping A_1 onto A_1.

As noted in Section 4.5.1, preexisting knowledge on the distribution of error probabilities δ_{cyc} and ϵ_{cyc} may be used in the computation of the semantic similarity. We give below a method to get estimates for both values.

We approximate δ_{cyc} – the probability of a series of different errors being compensated along a cycle – to $(\|\mathcal{C}\| - 1)^{-1}$; it corresponds to the probability of the last erroneous mapping in the cycle to map by accident to the original attribute and thus to correct previous errors. This value decreases with the size of the schemas.

We estimate ϵ_{cyc} with standard maximum-likelihood techniques applied to the feedback information we receive. From the probability of receiving positive feedback from a cycle of length $\|f\|$ knowing that the

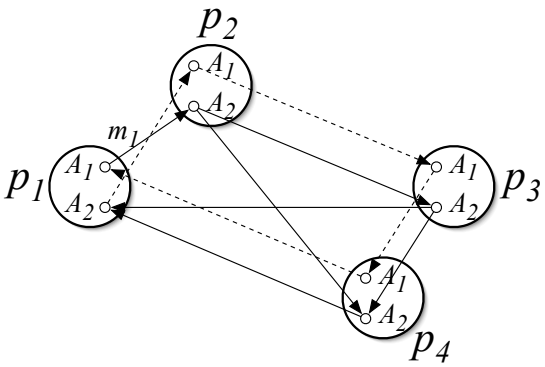

Figure 5.1: New mapping candidates: peer p_1 always receive negative feedback when reformulating queries through attribute mapping m_1; it might instead consider a new mapping m'_1 mapping its own A_1 onto peer p_2's A_1 and resulting in positive feedback.

mean error probability on the mapping links is ϵ_{cyc},

$$P(f_{\circlearrowleft}^{+}|\epsilon_{cyc}) = (1-\epsilon_{cyc})^{\|f\|} + (1-(1-\epsilon_{cyc})^{\|f\|})\delta_{cyc},$$

and from its negative counterpart, we derive the density function for the likelihood of ϵ_{cyc}:

$$L(\epsilon_{cyc}|\boldsymbol{f}_{\circlearrowleft}) = K \prod_{f_{\circlearrowleft}^{+}\in\boldsymbol{f}_{\circlearrowleft}^{+}} ((1-\epsilon_{cyc})^{\|f_{\circlearrowleft}^{+}\|} + (1-(1-\epsilon_{cyc})^{\|f_{\circlearrowleft}^{+}\|})\delta_{cyc}) \prod_{f_{\circlearrowleft}^{-}\in\boldsymbol{f}_{\circlearrowleft}^{-}} (1-(1-\epsilon_{cyc})^{\|f_{\circlearrowleft}^{-}\|})(1-\delta_{cyc})$$

where K encompasses the prior distributions on ϵ_{cyc} and $\boldsymbol{f}_{\circlearrowleft}$. The local maximum of this function over $[0,1]$ gives a good approximation of ϵ_{cyc}, supposing we have sufficient feedback information.

What is the result of this process in the long run? It depends of course on the initial setting but in the end, this method attempts to obtain a mapping consensus based on the different feedback cycles detected in the network. Considering a high density of links and relatively few unsound mappings, the method converges (i.e., repairs all unsound mappings) rapidly, since peers can base their decisions on numerous and meaningful feedback cycles. For settings where links are scarce, peers do not have sufficient information for making sensible choices, and results may diverge.

Several parameters are of particular interest: the number of peers n, the fraction of attribute mappings initially unsound $eRate$, the number of concepts $\|\mathcal{C}\|$, the initial time-to-live TTL of the probe queries, and the

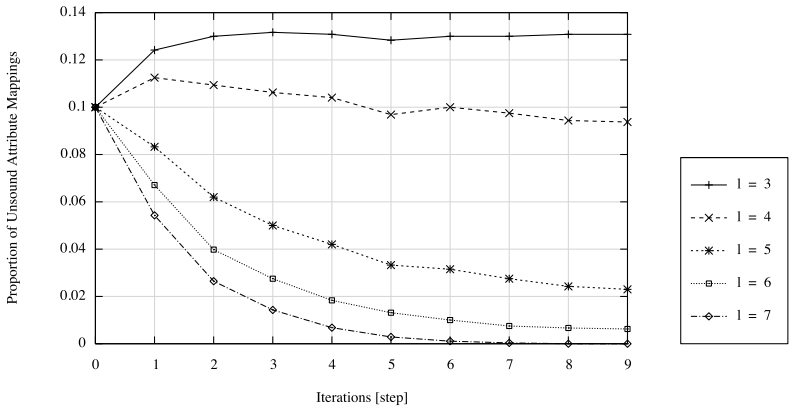

Figure 5.2: Sensitivity to the number of outgoing edges, with 25 semantic domains, $eRate = 0.1$, $\|\mathcal{C}\| = 4$ and $TTL = 5$.

number of outgoing translation links l per peer. The figures below show experimental results for topologies where $n = 25$, $eRate = 0.1$, $\|\mathcal{C}\| = 4$, $TTL = 5$ and $l = 5$ and where one of those parameters varies. All the curves are averaged over ten consecutive runs. At every step, the peers start by issuing a couple of queries with a high TTL for estimating the error rate as explained in the preceding section. Then, each peer sends a query picking a random concept for every outgoing edge and modifies the attribute mapping corresponding to that concept depending on the results of the analysis explained above. Steps are represented on the x-axis. The graph shows the evolution of the percentage of unsound attribute mappings, starting at a rate $eRate$ initially. Clearly, the outcome depends on the density of links, which has a direct impact on the number of cycles we have at our disposal for taking mapping decisions. For $l = 4$ and the topology considered, we get on average only one feedback message per mapping candidate, which is obviously insufficient to take sensible decisions. For $l = 5$ and $l = 6$, the value raises to 1.8 and 2.9 respectively, and most of the unsound attribute mappings get corrected after ten iterations. Finally, for $l = 7$, we get enough evidences (4.5 per mapping candidate on average) for correcting all the erroneous attribute mappings, thus reaching a perfect semantic agreement in eight steps.

Similar results may be observed for variable TTLs. Figure 5.3 shows results using the same parameters as before, but this time for a fixed number of outgoing edges ($l = 4$) and TTLs ranging from 3 to 6. Again, for low values, peers do not get sufficient feedback information to correct the mappings. Starting with $TTL = 4$ (1.8 positive feedbacks per decision), peers receive sufficient information to correct more than 75%of the unsound mappings after nine iterations. Low-connectivity networks may thus benefit from increasing the TTL value of their queries in order to get

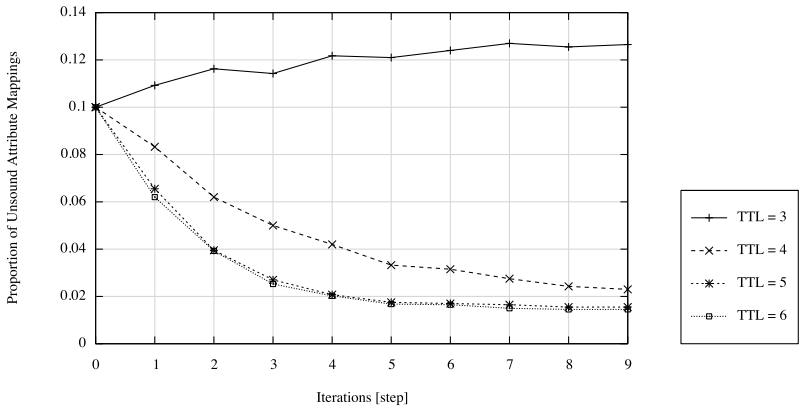

Figure 5.3: Sensitivity to the TTL, with 25 semantic domains, $eRate = 0.1$, $\|\mathcal{C}\| = 4$, and $l = 5$.

sufficient feedback information for the peers.

Our approach is rather insensitive to variations of the initial error rate (see Figure 5.4) until a certain threshold, where too few sound mappings are present initially to reach a correct consensus based on the feedback cycles. When too many unsound mappings coexist initially, a common semantic agreement can still emerge (see for example $eRate = 0.4$ in Figure 5.4), without necessary corresponding to *human assigned* semantics (see Section 1.1). In that case, some additional information – introduced by human agents or collected through some additional analysis (see Section 5.4) – is necessary to rectify the semantic agreement.

Finally, it is worth mentioning that our approach scales very well with the number of nodes. This is not surprising, considering that our methods rely solely on local interactions (no central component or computation). In Figure 5.5, we consider networks ranging from 50 to 800 peers, without any fundamental variation related to the results of our analyses. The small deviations are due to the *shortcuts* in the small world topology, which connect two random peers in the network. The bigger the graph, the less likely it is that those links can be used to form cycles within a certain neighborhood.

5.3 Result Analysis

Let us now consider the second part of the analysis, in which peers analyze and categorize documents they receive. The process is as follows: at every step, the peers first issue a couple of queries with a high TTL for estimating the error rate as explained in the preceding section. Then, for each of their outgoing links, the peers pick a concept randomly and issue a

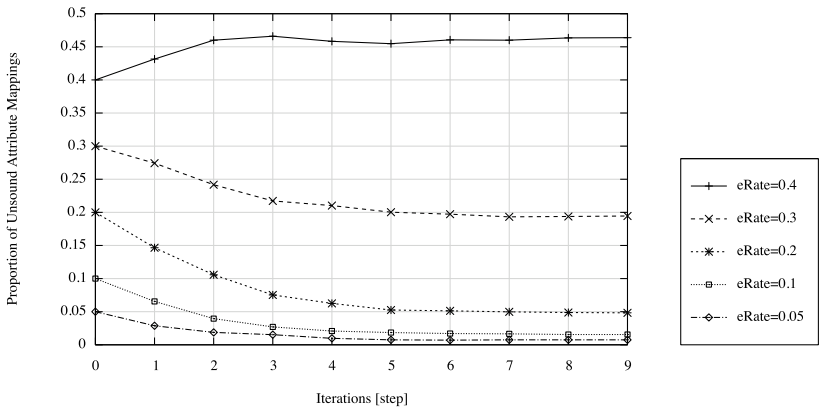

Figure 5.4: Sensitivity to the initial error rate, with 25 semantic domains, $\|\mathcal{C}\| = 4$, $TTL = 5$ and $l = 5$.

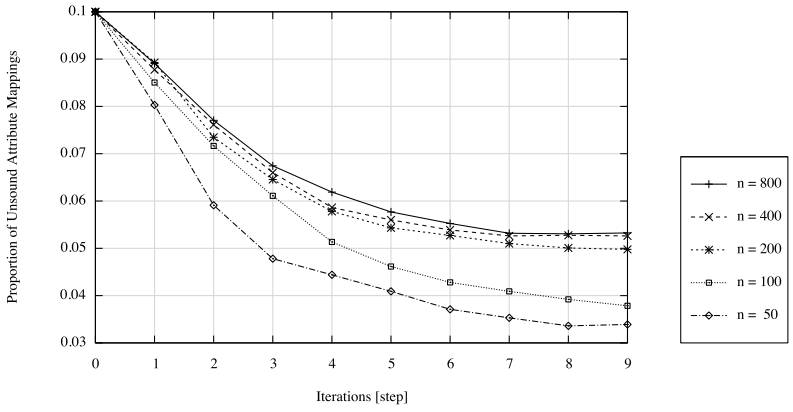

Figure 5.5: Scalability, with $eRate = 0.1$, $\|\mathcal{C}\| = 4$, $TTL = 5$, $l = 5$ and a varying number of semantic domains.

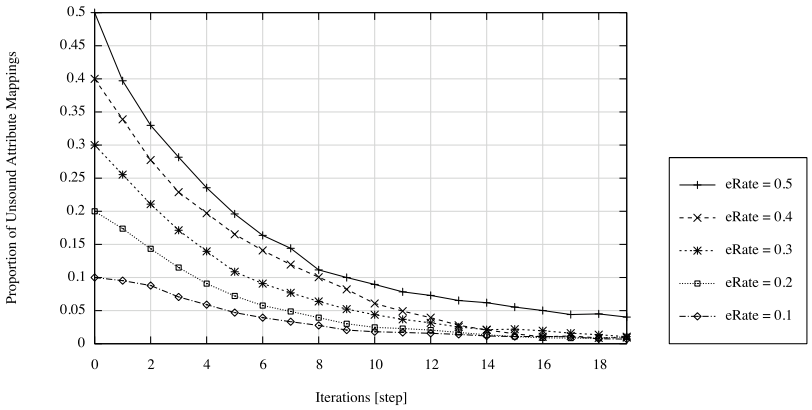

Figure 5.6: Sensitivity to the initial error rate, with 50 semantic domains, 100 documents, $eRate = 0.1$, $\epsilon_{res} = 0.1$, $\|\mathcal{C}\| = 4$, $TTL = 3$ and $l = 2$.

query asking for documents related to that concept. In return, they receive documents they analyze following the method described in Section 4.5.2. They modify the attribute mapping they have used to forward the query with the most probable mapping candidate if its likelihood of being sound is greater or equal to 0.5.

For our experiments, we used a fixed set of documents scattered randomly among the peers. Each documents correspond to one concept. The owner of a document has a probability (ϵ_{res}) of misclassifying the document by relating it to a wrong concept. The peers approximate ϵ_{res} to a fixed, low value $\epsilon_{res} = 5\%$ in the following experiments. For our setting, we set δ_{res} to $(\|\mathcal{C}\| - 1)^{-1}$.

Unless specified otherwise, we used a network of 50 peers sharing in total 100 documents, 2 outgoing mappings per peer, 4 concepts, a TTL of 3, an initial error rate of 10%, and a probability of 10% of misclassifying a document returned by another peers.

First, it is interesting to observe that this approach is very robust against the initial error rate, mainly because of the short feedback loop (one mapping link suffices here to return documents) compared to the relatively long cycles used previously. Figure 5.6 shows the results for a varying initial proportion of unsound mappings.

Nevertheless, the approach is rather sensitive to the rate of misclassification of documents, as shown in Figure 5.7. This is especially true since we do not try to evaluate this parameter but consider a mere fixed value.

The approach taken here is completely local as it does not take into consideration any global behavior, and scales well with the number of peers (see Figure 5.8). Note that we increase the number of documents linearly with the number of peers to keep the average number of docu-

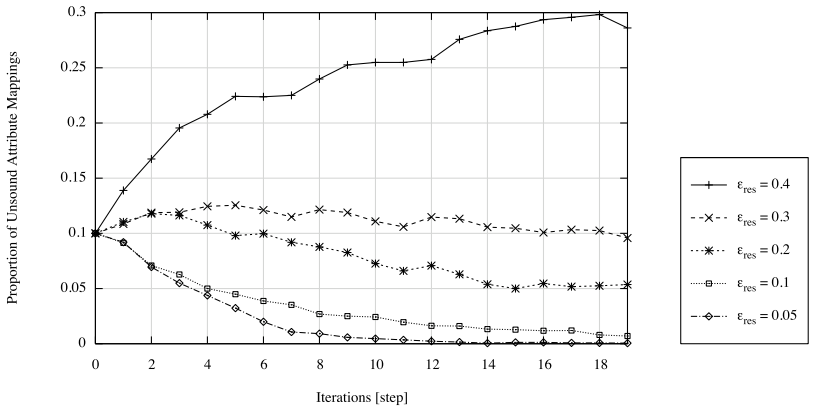

Figure 5.7: Sensitivity to misclassification rate, with 50 semantic domains, 100 documents, $eRate = 0.1$, $\|\mathcal{C}\| = 4$, $TTL = 3$ and $l = 2$.

ments per peer constant. This number is essential to our analysis, since it is directly proportional to the number of evidences a peer gathers for every probe query. This effect is depicted in Figure 5.9: peers start having trouble correcting the mappings as they get less and less documents returned for their queries (documents scarcity).

5.4 Combined Analysis

Many possibilities exist for combining the two analyses. We chose a simple one: at each step, every peer first performs a result analysis step (modifying the attribute mapping depending on the results returned) and then performs a cycle analysis step (trying to reach some local agreement on mappings based on cycle feedback). The results for topologies with 25 peers, 50 documents, $\epsilon_{res} = 0.1$, 4 concepts, 2 outgoing edges, TTLs of 3 (results) or 6 (cycles) and varying error rates on initial mappings are depicted in Figure 5.10.

This method takes more time to converge than the two analyses applied separately, as the analyses keep interfering with each other until some state is reached that is consistent from both a cycle and a feedback analyses point of view. Note that the combined method in the end outperforms the two individual methods applied separately (e.g., more than 98% of unsound mappings corrected after 50 steps with 50% erroneous mappings initially).

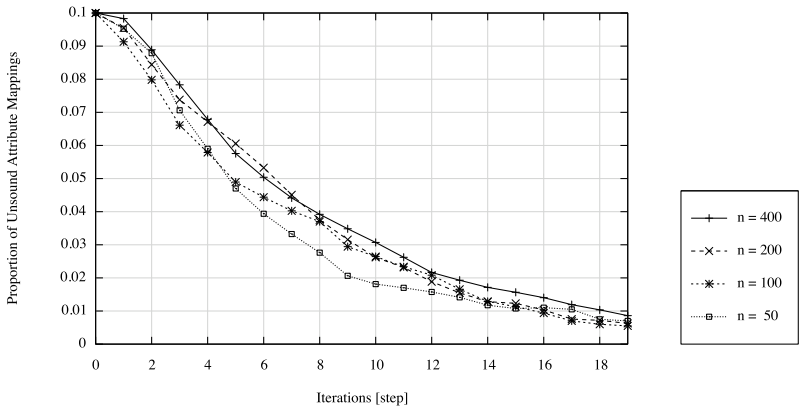

Figure 5.8: Scalability, with 2 documents/peer on average, $eRate = 0.1$, $\epsilon_{res} = 0.1$, $\|\mathcal{C}\| = 4$, $TTL = 3$, $l = 2$, and a varying number of semantic domains.

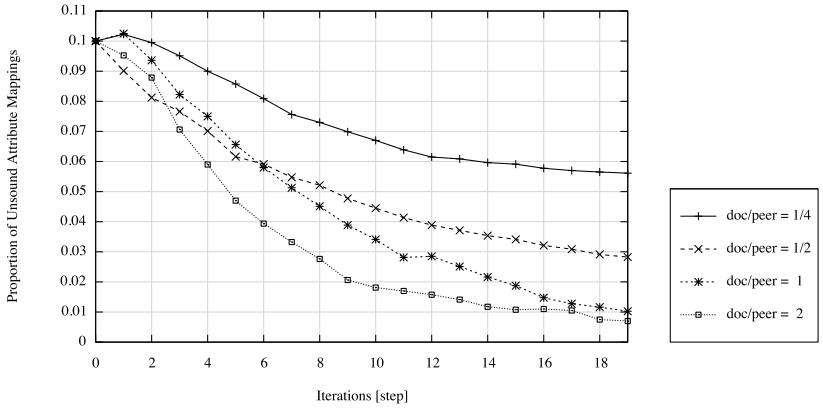

Figure 5.9: Sensitivity to number of documents, with 50 semantic domains, $eRate = 0.1$, $\|\mathcal{C}\| = 4$, $TTL = 3$ and $l = 2$.

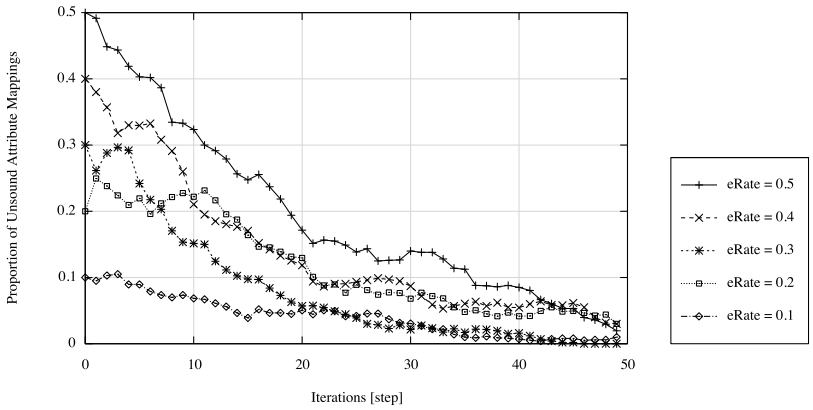

Figure 5.10: Combined analyses, with 25 semantic domains, 50 documents, $\epsilon_{res} = 0.1$, $\|\mathcal{C}\| = 4$, $TTL = 3$ (results) or 6 (cycles), $l = 2$, and a varying initial error rate.

5.5 Related Work

Our research is related to the newly emerging field of evolutionary linguistics [SH06], which investigates the self-organization of languages and the co-evolution of vocabularies and meaning. Evolutionary linguistics research is based on the hypothesis that language is a complex adaptive system that emerges through adaptive interactions between agents and continues to evolve in order to remain adapted to the needs and capabilities of the agents. Self-organizing vocabularies [Ste96], for example, are based on sets of agents, which communicate by randomly associating a fixed set of words to a fixed set of meanings and repeatedly evaluate how successful their communicative acts have been. Depending on the success, the binding between a word and a concept is maintained or replaced by a new random coupling. Contrary to evolutionary vocabularies, however, our approach requires an additional level of indirection, i.e., mappings, as the sets of heterogeneous vocabulary terms (the attributes used by the individual information sources) are fixed in our context. Thus, we do not aim at obtaining universally agreed upon names but focus on ways to relate and adapt schema mappings between fixed terms.

The approach presented above is also related to methods to iteratively refine schema or ontology mappings. Clio [MHH+01] features a semi-automatic schema matcher using manual intervention to revise mappings. LSD [DDH01] exploits domain constraints and user feedback to iteratively adapt mappings. QOM [ES04] provides a series of matching steps based on similarity metrics to iteratively produce mappings with increasing quality. COMA [DR02] is a framework to combine several schema matchers; it introduces an approach aiming at reusing results from previ-

ous match operations, and several mechanisms to combine the results of various matcher executions. Avigdor Gal recently proposed [Gal06] the simultaneous generation and examination of K best schema mappings to identify useful mappings. Our method radically differs from all those approaches in the sense that we see semantics as an agreement and propose a method that solely relies on collaborative, decentralized computations while all the aforementioned method are local.

5.6 Conclusions

In this chapter, we have proposed an approach facilitating interoperability of autonomous parties by deriving global semantics (agreements) from a set of initial and partial agreements and series of local interactions. We take advantage of explicit local mappings to derive an implicit global agreement by iteratively modifying some of the mappings. We have developed our approach in the formal model presented in Chapter 4, which is built around a set of instruments that enable us to assess the relative soundness of the query reformulations. We showed in a series of experiments that the quality of the agreements that can be obtained with our methods depends on the initial implicit agreement (soundness of the mappings) and of the amount of information that can be obtained from the network (e.g., feedback cycles). We see our approach as a complementary effort to the on-going standardization in the area of semantics, which may help to improve their acceptance and application by augmenting the top-down approach towards creating standard semantics with a dual bottom-up strategy.

Chapter 6

Probabilistic Message Passing

6.1 Introduction

Semantic Gossiping (Chapter 4) introduced novel methods relying on the analysis of mapping closures to detect semantic disagreement and selectively forward queries in a network of heterogeneous parties. Those methods suffer however from a few issues that limit their applicability and performance in practice:

1. In Semantic Gossiping, peers perform their analyses autonomously and independently of the other peers. Peers forming a semantic cycle all examine the cycle in isolation without exchanging or reusing the computations of the other peers on the cycle.

2. Schema mapping cycles are considered to be independent in Semantic Gossiping. This, however, is a gross approximation as an incorrect mapping has a simultaneous effect on all the cycles it belongs to. Thus, taking into account the correlation of the various cycles would inherently lead to better results in the detection of mappings diverging in their semantics with the rest of the network.

3. While Semantic Gossiping only considers equality of attributes or concepts, subsumption of concepts plays an ever increasing role on the Internet. XML and other semi-structured representations often organize their attributes in subsumption hierarchies, either implicitly through nested structures, or explicitly through type extensions (e.g., types derived by extension or restriction in XML Schemas). The situation is even more radical for new formats such as RDF/S and OWL, which base most of their language constructs on subsumption relations.

In the following, we develop a new framework based on sum-product message passing (i.e., belief propagation) that tackle those three issues [CMAF06]. This new approach pushes Semantic Gossiping to the next level, as we do not limit our analyses to a few cycles, but take into account the full correlation of the mapping operations throughout the network in a decentralized and parallel analysis.

We concentrate below on two distinct settings for elaborating our approach: Section 6.3.2 focuses on a simple setting where mappings are composed of renamings relating pairs of attribute. Section 6.3.3 extends our approach to analyze directed cycles and parallel paths of reformulations on hierarchies of subsumed attributes and inclusion mappings. For clarity, we focus below on the semantic mediation layer (upper layer of Figure 3.4) and consider each peer as a distinct semantic domain. The methods presented in this chapter can easily be extended to more expressive mappings (see Chapter 4) or more complex network models (see Chapters 4 and 8).

We start below with a brief problem definition extending the PDMS model introduced in Chapter 4 and an illustrating example of a PDMS with subsumption relations. We then present our probabilistic techniques to determine the soundness of schema mappings in PDMS settings in a totally automated way and without any form of central coordination. Our methods are based on the analysis of cycles and parallels paths in the graph of schema mappings: after detecting mapping inconsistencies by comparing transitive closures of mapping operations, we build a global probabilistic inference model spanning the entire PDMS system. We show how to construct this model in a totally decentralized way by involving local information only. We describe a decentralized and efficient method to derive mapping quality measures from our model. We show how to embed our approach in PDMSs with a very modest communication overhead by piggybacking on normal query processing operations. Finally, we present an evaluation of our technique applied on both generated and real-world data.

6.2 Problem Definition

We start with an extension of the PDMS model of Chapter 4 to include subsumption relations and inclusive mappings. We model PDMSs as collections of peers; each individual peer $p \in \mathcal{P}$ represents a database DB_p storing data according to a distinct structured schema S_p. Note that a peer could in practice represent a (potentially large) cluster of databases all adhering to the same schema. The basis of the data model is similar to the one presented in Section 4.2.1: peers store information with respect to some concepts we call *attributes* $A \in \mathcal{S}_p$ (e.g., attributes in a relational schema, elements or attributes in XML and classes or properties in RDF). Each local attribute is assigned a set of fixed interpretations A^I from an

abstract and global domain of interpretations Δ^I with $A^I \subseteq \Delta^I$. Arbitrary peers are not aware of such assignments. We say that two attributes A_i and A_j are *equivalent*, and write $A_i \equiv A_j$ if and only if $A_i^I = A_j^I$. Local attributes can be organized in hierarchies by relating a sub-attribute A_{sub} to a super attribute A_{super}: we say that a super-attribute A_{super} *subsumes* a sub-attribute A_{sub}, and write $A_{sub} \sqsubseteq A_{super}$ if and only if $A_{sub}^I \subseteq A_{super}^I$.

Peers are connected one another through (un)directed edges representing (un)directed pairwise schema mappings. A schema mapping $\mu_{p_i \to p_j}$ allows a query posed against a local schema at peer p_i to be evaluated against another schema at peer p_j. Schema mappings are similar to the ones introduced in Section 4.2.2 and are made of attribute mappings m_k. The mapping operations can be uni or bi-directional depending on the setting. For clarity, we limit below the renamings appearing in the attribute mappings to one-to-one renamings; thus, each attribute mapping $m_k \in \mu_{p_i \to p_j}$ connects a pair of semantically related attribute through a renaming $f(A_l) := A_k$ with $A_k \in S_{p_i}$ and $A_l \in S_{p_j}$. Note that the renamings can also be used for syntactic transformations (e.g., transforming a date from one format to another, see Chapter 4). Each renaming can either take the form of an equivalence renaming as described above, or of a containment renaming $f(A_l) :\sqsubseteq A_k$. We call the attribute mappings with containment renamings *containment attribute mappings*. Containment attribute mappings can be used as any other attribute mappings to reformulate query atoms (see Section 4.2.2) from the source to the target peer, but produce contained rewritings of the atoms. Containment attribute mappings are directed in essence but can also express undirected equivalence mappings, e.g., $A_1 \sqsubseteq A_2 \wedge A_2 \sqsubseteq A_1$ entails $A_1 \equiv A_2$. In that context, we say that a mapping is *sound* if it always produces contained rewritings of all the query atoms it reformulates through containment mappings, and equivalent rewritings of all the query atoms it reformulates through equivalence mappings. A reformulation is contained if it is obtained by one or more containment mappings.

Peers can reformulate queries locally on their subsumption hierarchies: they can substitute sub-attributes for super-attributes in local queries q to create local, contained and sound reformulations: replacing (some of) the attributes appearing in a local query q by their corresponding sub-attributes creates a contained rewriting of the query $q' \sqsubseteq q$ returning a subset of the set of results $q(DB_p)$ of the original query: $q'(DB_p) \subseteq q(DB_p)$. Those local reformulations might in turn be reformulated through schema mappings and explicitly appear in the reformulation history (e.g., reformulation trace in Section 4.6) of a given query.

Given this setting, our goal is to provide probabilistic guarantees on the soundness of the mappings, i.e., to determine $P(m_i = sound)$, where m_i is a local attribute mapping for attribute A_i taking the form of an equality or containment mapping. As any process in a PDMS, we want our method to operate without any global coordination, in a purely decen-

tralized manner. Also, we would like our methods to be totally automated, as precise as possible and fast enough to be applied on large schemas or ontologies.

6.2.1 An Introductory Example

Before delving into technicalities, we start with a high-level, introductory example of our approach. Let us consider the simple PDMS network depicted in Figure 6.1. This network is composed of four databases $p_1, \ldots, p_4$. All databases store a collection of XML documents related to pieces of art, but structured according to four different schemas (one per database). Each database supports XQuery as query language. Various pairwise XQuery schema mappings have been created (both manually and automatically) to link the databases.

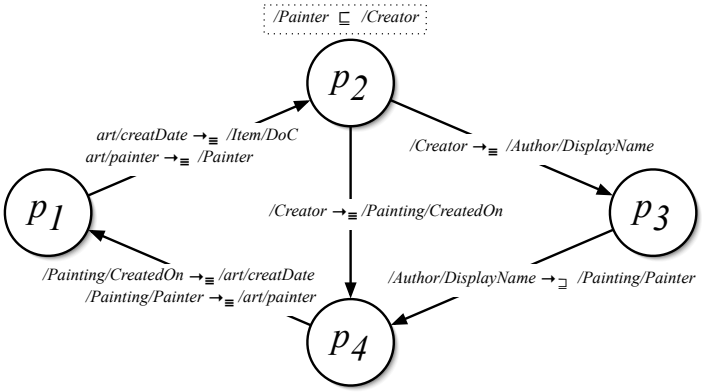

Figure 6.1: A simple directed PDMS network of four peers and five schema mappings, here depicted for the attribute *Creator*.

Let us suppose that a user in p_2 wishes to retrieve the names of all artists having created a piece of work related to some river. The user could locally pose an XQuery like the following:

```
q_1 =
FOR $c IN distinct-values (ArtDatabank//Creator)
WHERE $c//Item LIKE "%river%"
RETURN <myArtist> $c </myArtist>
```

This query basically boils down to a selection on the title $\sigma_{Item=\%river\%}$ followed by a projection on the attribute *Creator* $\pi_{Creator}$. The user issues the query and patiently awaits for answers, both from his local database and the rest of the network.

In a standard PDMS, the query would be forwarded through both outgoing mappings of p_2, generating a fair proportion of false positives as one of these two mappings (the one between p_2 and p_4) is unsound for the attribute *Creator* (the mapping erroneously maps *Creator* in p_2 onto *CreatedOn* in p_4, see Figure 6.1). Luckily for our user, the PDMS system he is using implements our message passing techniques. Without any prior information on the mappings, the system detects inconsistencies for the mappings on *Creator* by analyzing the cycles $p_1 \rightarrow p_2 \rightarrow p_4 \rightarrow p_1$ and $p_1 \rightarrow p_2 \rightarrow p_3 \rightarrow p_4 \rightarrow p_1$, as well as the parallel paths $p_2 \rightarrow p_4$ and $p_2 \rightarrow p_3 \rightarrow p_4$ in the mapping network. In a decentralized process, the PDMS constructs a probabilistic network and determines that the semantics of the attribute *Creator* will most likely be preserved by all mappings, except by the mapping between p_2 and p_4 which is more likely to be faulty. Thus, this specific query will be routed through mapping $p_2 \rightarrow p_3$, and then iteratively to p_4 and p_1. In the end, the user will retrieve all artist names as specified, without any false-positive since the mapping $p_2 \rightarrow p_4$ was ignored in the query resolution process.

6.3 Modeling PDMSs as Factor-Graphs

We take advantage of query messages being forwarded from one peer to another to detect inconsistencies in the network of mappings. We represent individual mappings and network information as related random variables in a probabilistic graphical model. We will then efficiently evaluate marginal probabilities, i.e., mapping soundness, using those models.

6.3.1 A Quick Reminder on Factor-Graphs and Message Passing Schemes

We give below a brief overview of message passing techniques. For a more in-depth coverage, we refer the interested reader to one of the many overviews of this domain [KFL01]. Probabilistic graphical models are a marriage between probability theory and graph theory. In many situations, one can deal with a complicated global problem by viewing it as a factorization of several local functions, each depending on a subset of the variables appearing in the global problem. As an example, suppose that a global function $g(x_1, x_2, x_3, x_4)$ factors into a product of two local functions f_A and f_B: $g(x_1, x_2, x_3, x_4) = f_A(x_1, x_2) f_B(x_2, x_3, x_4)$. This factorization can be represented in a graphical form by the *factor-graph* depicted in Figure 6.2, where variables (circles) are linked to their respective factors (black squares). Often, one is interested in computing a

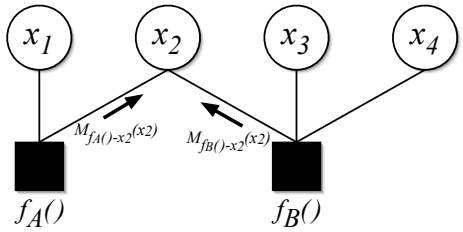

Figure 6.2: A simple factor-graph of four variables and two factors.

marginal of this global function, e.g.,

$$\begin{aligned} g_2(x_2) &= \sum_{x_1}\sum_{x_3}\sum_{x_4} g(x_1, x_2, x_3, x_4) \\ &= \sum_{\sim\{x_2\}} g(x_1, x_2, x_3, x_4) \end{aligned}$$

where we introduce the summary operator $\sum_{\sim\{x_i\}}$ to sum over all variables but x_i. Such marginals can be derived in an efficient way by a series of *sum-product* operations on the local function, such as:

$$g_2(x_2) = \left(\sum_{x_1} f_A(x_1, x_2)\right)\left(\sum_{x_3}\sum_{x_4} f_B(x_2, x_3, x_4)\right).$$

Interestingly, the above computation can be seen as the product of two messages $M_{f_A()\to x_2}(x_2)$ and $M_{f_B()\to x_2}(x_2)$ sent respectively by $f_A()$ and $f_B()$ to x_2 (see Figure 6.2). The *sum-product* algorithm exploits this observation to compute all marginal functions of a factor-graph in a concurrent and efficient manner. Message passing algorithms traditionally compute marginals by sending two messages — one in each direction — for every edge in the factor-graph:

message from variable x to local factor $f()$:

$$M_{x\to f()}(x) = \prod_{h\in n(x)\setminus\{f()\}} M_{h\to x}(x)$$

message from local factor $f()$ to variable x

$$M_{f()\to x}(x) = \sum_{\sim\{x\}}\left(f() \prod_{y\in n(f())\setminus\{x\}} M_{y\to f()}(y)\right)$$

where $n(\cdot)$ stands for the neighbors of a variable / function node in the graph. The above computations are known to be exact for cycle-free

factor-graphs; in contrast, applications of the sum-product algorithm in a factor-graph with cycles only result in approximations of the marginals [MWJ99]. However, some of the most exciting applications of the sum-product algorithms (e.g., decoding of turbo or LDPC codes) arise precisely in such situations. We show below that this is also the case for factor-graphs modelling Peer Data Management Systems. Finally, note that Belief Propagation as introduced by Judea Pearl [Pea88] is actually a specialized case of a standard message passing sum-product algorithm.

6.3.2 On Factor-Graphs in Undirected PDMSs

We start with the modeling of a network of undirected equivalence mappings as a factor-graph. That factor-graph will in turn be used in Section 6.4 to derive quality measures on the soundness of the mappings in the network. Note that, depending on the PDMS, one can choose between two levels of granularity for storing factor-graphs and computing related probabilistic values: coarse granularity – where peers only store one factor-graph per schema mapping and where they derive only one global value on the soundness of the mapping – and fine granularity – where peers store one instance of the local factor-graph per attribute mapping and where they derive one probabilistic soundness value per attribute. We suppose we are in the latter situation but show derivations for only one attribute in the following. Values for the other attributes can be derived in a similar fashion.

Cyclic Mappings

Semantic overlay network topologies are not generated at random. On the contrary, they are constructed by (computerized or human) agents aiming at interconnecting semantically overlapping information sources. We can expect very high clustering coefficients in those networks, since similar sources tend to bond together and create clusters of sources. As an example, a study of an online network of related bioinformatic schemas – all available through an SRS repository, see the following chapter – shows an exponential degree distribution and an unusually high clustering coefficient. Consequently, we can expect semantic schema graphs to exhibit *scale-free* properties and an unusually high number of loops [BM05].

Let us assume we have detected a cycle of mappings $m_0, m_1, \ldots, m_{n-1}$ connecting n distinct peers $p_0, p_1, \ldots, p_{n-1}, p_0$ in a circle through equivalence mappings. Cycles of mappings can easily be discovered by the peers in the PDMS network, either by proactively flooding their neighborhood with probe messages with a certain Time-To-Live (TTL) or by examining the trace of queries routed through the network as explained in Chapter 4. We take advantage of transitive closures of mapping operations in

the cycle to compare a query q posed against the schema of p_0 to the corresponding query q' reformulated through all the n mappings along the cycle: $q' = m_{n-1} \circ m_{n-2} \circ \ldots \circ m_0(q)$. q and q' can be compared on an equal basis since they are both expressed in terms of the schema of p_0. In an ideal world, $q' \equiv q$ since the transformed query q' is the result of n equivalence mappings applied on the original query q. In a distributed setting, however, this might not always be the case, both because of the lack of expressiveness of the mappings and of the fact that mappings can be created in (semi) automatic ways.

As discussed in Section 4.5.1, three subcases can occur when comparing attribute A_i in an operation $op(A_i)$ appearing in the original query q to the attribute A_j from the corresponding operation $op'(A_j)$ in the transformed query q':

$A_j = A_i$: this occurs when the attribute, after having been reformulated n times through the mappings, still maps to the original attribute when returning to the semantic domain of p_0. Since this indicates a high level of semantic agreement along the cycle for this particular attribute, we say that this represents positive feedback f^+ on the soundness of the mappings in the cycle.

$A_j \neq A_i$: this occurs when the attribute, after having been transformed n times through the mappings, maps to a different attribute when returning to the semantic domain of p_0. As this indicates some disagreement on the semantics of A_i along the cycle of mappings, we say that this represents negative feedback f^- on the soundness of the mappings in the cycle.

$A_j = \perp$: this occurs when some intermediary schema does not have any representation for the attribute in question, i.e., cannot map the attribute onto one of its own attributes. This does not give us any additional (feedback) information on the level of semantic agreement along the cycle, but can still represent some valuable information in other contexts, for example when analyzing query forwarding on a syntactic level (see Section 4.4).

Also to be taken into account, the fact that series of erroneous mappings on A_i can accidentally *compensate* their respective errors and actually create a correct composite mapping $m_{n-1} \circ m_{n-2} \circ \ldots \circ m_0$ in the end. Assuming a probability δ of two or more mapping errors being compensated along a cycle in this way, we can determine the conditional probability of a cycle producing positive feedback $f^+_{\circlearrowleft}$ given the soundness of its constituting mappings $m_0, \ldots, m_{n-1}$:

$$P(f^+_{\circlearrowleft} | m_0, \ldots, m_{n-1}) = \begin{cases} 1 & \text{if all mappings sound} \\ 0 & \text{if one mapping unsound} \\ \delta & \text{if two or more mappings unsound.} \end{cases}$$

This conditional probability function allows us to create a factor-graph from a network of interconnected mappings. We create a global factor-graph by linking local variables representing the soundness of the mappings to conditional probability functions. The algorithm to derive the factor-graph is given in Algorithm 6.1.

Algorithm 6.1 Deriving a global factor-graph from a PDMS network

```
/*create a variable and a factor for each mapping*/
for all mapping m in PDMS do
  add m.factor to global-factor-graph;
  add m.variable to global-factor-graph;
  connect m.factor to m.variable;
end for
for all mapping-cycle c in PDMS do
  /*create a variable and a factor for each cycle*/
  add c.feedback.factor to global-factor-graph;
  add c.feedback.variable to global-factor-graph;
  connect c.feedback.factor to c.feedback.variable;
  for all mapping m in mapping-cycle c do
    /*connect the mappings to the corresponding cycles*/
    connect c.feedback.factor to m.variable;
  end for
end for
```

Figure 6.3 illustrates the derivation of a factor-graph from a simple semantic network of four peers $p_1, \ldots, p_4$ (left-hand side of Figure 6.3). The peers are interconnected through five mappings $m_{12}, m_{23}, m_{34}, m_{41}$ and m_{24}. One may attempt to obtain feedback from three different mapping cycles in this network:

$$\boldsymbol{f}^{\mathbf{1}}_{\circlearrowleft} : m_{12} - m_{23} - m_{34} - m_{41}$$
$$\boldsymbol{f}^{\mathbf{2}}_{\circlearrowleft} : m_{12} - m_{24} - m_{41}$$
$$\boldsymbol{f}^{\mathbf{3}}_{\circlearrowleft} : m_{23} - m_{34} - m_{24}.$$

The right-hand side of Figure 6.3 depicts the resulting factor-graph, containing from top to bottom: five one-variable factors for the prior probability functions on the mappings, five mapping variables m_{ij}, three factors linking feedback variables to mapping variables through conditional probability functions (defined as explained above), and finally three feedback variables f_k. Note that feedback variables are usually not independent: two feedback variables are correlated as soon as the two mapping cycles they represent have at least one mapping in common (e.g., in Figure 6.3, where all three feedbacks are correlated).

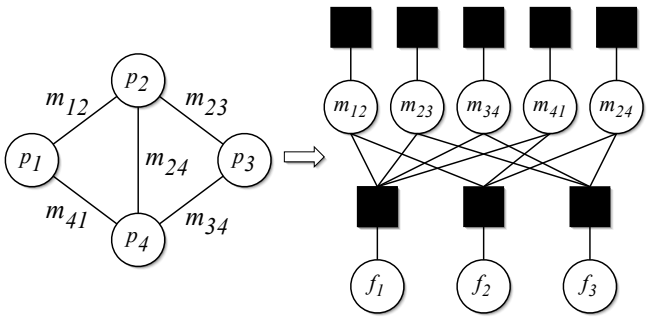

Figure 6.3: Modeling an undirected network of mappings (left-hand side) as a factor-graph (right-hand side); the nodes of the factor-graph represent, from top to bottom, probability functions encapsulating *a priori* information on the mappings, variables for the attribute mappings, probability functions relating attribute mapping variables to feedback variables, and finally variables for the feedback that can be gathered from the network.

6.3.3 On Factor-Graphs in Directed PDMSs with Containment Mappings

One may derive similar factor-graphs in directed PDMSs with subsumption, focusing this time on directed mapping cycles and parallel mapping paths. Factors from directed mapping cycles $p_0 \rightarrow p_1 \rightarrow, \ldots \rightarrow p_{n-1} \rightarrow p_0$ are defined in exactly the same way as explained above for the undirected case. Those cycles can contain zero, one or several containment mappings. When comparing attribute A_i in an operation $op(A_i)$ appearing in the original query q to the attribute A_j from the corresponding operation $op'(A_j)$ in the transformed query q', four cases can occur:

$A_j = A_i$: this occurs when the attribute, after having been transformed n times through the mappings, still maps to the original attribute when returning to the semantic domain of p_0. This represents positive feedback f^+.

$A_j \sqsubseteq A_i$: this occurs when the attribute, after having been transformed n times through the mappings, maps to a sub-attribute of the original attribute A_i. This is also considered as positive feedback f^+.

$A_j \neq A_i$ and $A_j \not\sqsubseteq A_i$: this occurs when the attribute, after having been transformed n times through the mappings, maps to an an attribute that is neither subsumed by the original attribute nor identical to the original attribute when returning to the semantic domain of p_0. This indicates some disagreement on the semantics of A_i and is treated as negative feedback f^-.

$A_j = \perp$: this occurs when some intermediary schema does not have any representation for the attribute in question, i.e., cannot map the attribute onto one of its own attributes. This does not give us any additional (feedback) information on the level of semantic agreement along the cycle, but can still represent some valuable information in other contexts, for example when analyzing query forwarding on a syntactic level (see Section 4.4).

Note that we consider the feedback as negative only when we are certain that an unsound reformulation has occurred. Also, note that in settings with strict subsumption relations (i.e., $A_j \sqsubset A_i$ instead of $A_j \sqsubseteq A_i$), one may actually introduce stricter tests on the feedbacks (e.g., $A_j = A_i$ yields negative feedback as soon as a strict contained mapping is used along the cycle).

Containment mappings introduce a new type of compensating errors, when an incorrect subsumption gets compensated in subsequent subsumptions, e.g., if $A_{p_1} \rightarrow_{\sqsupseteq} B_{p_2}$, $B_{p_2} \rightarrow_{\sqsupseteq} C_{p_1}$ with $A_{p_1} \not\sqsupseteq B_{p_2}$, $B_{p_2} \sqsupseteq C_{p_1}$ and $A_{p_1} \sqsupseteq C_{p_1}$. The probability of correcting an erroneous cycle in that way can be estimated locally (see for example Section 6.4). Taking into account those compensations $\delta_{\sqsubseteq}$, the conditional probability of a cycle containing subsumption mappings producing positive feedback is:

$$P(f^+_{\circlearrowright} | m_0, \ldots, m_{n-1}) = \begin{cases} 1 & \text{if all mappings sound} \\ \delta_{\sqsubseteq} & \text{if one mapping unsound} \\ \delta + \delta_{\sqsubseteq} & \text{if two or more mappings unsound.} \end{cases}$$

Parallel mapping paths occur when two different series of mappings $\boldsymbol{m'}$ and $\boldsymbol{m''}$ share the same source and destination, e.g., $m'_0 \rightarrow m'_1 \rightarrow \ldots \rightarrow m'_{n'}$ and $m''_0 \rightarrow m''_1 \rightarrow \ldots \rightarrow m''_{n''}$, with m'_0 and m''_0 departing from the same peer and $m'_{n'}$ and $m''_{n''}$ arriving at the same peer, without any other common peer in-between. Those two parallel paths would be considered as forming an undirected cycle in an undirected network, but cannot be considered as such here due to the restriction on the direction of the mapping operations in a network of directed mappings.

If a query q is forwarded through both parallel paths, the destination peer p_{dest} can compare both $q' = m'_{n'}(\ldots(m'_0(q)))$ and $q'' = m''_{n''}(\ldots(m''_0(q)))$.

In this setting, four cases can occur when comparing attribute A'_i appearing in operation $op'(A'_i)$ in q' to attribute A''_j appearing in the corresponding operation $op''(A''_j)$ in q'':

$A''_j = A'_i$: this occurs when the original attribute, after having been transformed through both mapping paths, map to the same attribute at the destination. This is treated as positive feedback f^+.

$A''_j \sqsubseteq A'_i$ or $A'_i \sqsubseteq A''_j$: this occurs when the target attribute obtained by one of the series of mappings $\boldsymbol{m'}$ subsumes the attribute obtained by the other series of mappings $\boldsymbol{m''}$. This is treated as positive feedback f^+.

$A''_j \neq A'_i$ and $A''_j \not\sqsubseteq A'_i$ and $A'_i \not\sqsubseteq A''_j$: this occurs when the attribute, after having been transformed through both series of mappings, maps to two different attributes that are not related by any subsumption relation at the destination. This indicates some disagreement on the semantics of the attribute and is treated as negative feedback f^-.

$A''_j = \perp$ or $A'_i = \perp$: this occurs when some intermediary schema does not have a representation for the attribute in question, i.e., cannot map the attribute onto one of its own attributes. This does not give us any additional (feedback) information on the level of semantic agreement along the paths.

The conditional probability function for receiving positive feedback $f^+_{\rightrightarrows}$ through parallel paths knowing the soundness of the sets of mappings $\boldsymbol{m'} = \{m'_0, \ldots, m'_{n'}\}$ and $\boldsymbol{m''} = \{m''_0, \ldots, m''_{n''}\}$ with at least one containment mapping along one of the paths is:

$$P(f^+_{\rightrightarrows}|\boldsymbol{m'}, \boldsymbol{m''}) = \begin{cases} 1 & \text{if all mappings sound} \\ \delta_{\sqsubseteq} & \text{if one mapping unsound} \\ \delta + \delta_{\sqsubseteq} & \text{if two or more mappings unsound.} \end{cases}$$

Figure 6.4 shows an example of a directed mapping network with four peers and six mappings. Feedback from three directed cycles and three pairs of parallel paths might be gathered from the network:

$$\begin{aligned} \boldsymbol{f^1_{\circlearrowleft}} &: m_{12} \rightarrow m_{23} \rightarrow m_{34} \rightarrow m_{41} \\ \boldsymbol{f^2_{\circlearrowleft}} &: m_{12} \rightarrow m_{24} \rightarrow m_{41} \\ \boldsymbol{f^3_{\circlearrowleft}} &: m_{12} \rightarrow m_{21} \\ \boldsymbol{f^4_{\rightrightarrows}} &: m_{21} \| m_{24} \rightarrow m_{41} \\ \boldsymbol{f^5_{\rightrightarrows}} &: m_{24} \| m_{23} \rightarrow m_{34} \\ \boldsymbol{f^6_{\rightrightarrows}} &: m_{21} \| m_{23} \rightarrow m_{34} \rightarrow m_{41}. \end{aligned}$$

As for the undirected case, the right-hand side of Figure 6.4 represents the factor-graph derived from the directed mapping network of the left-hand side. Since undirected mapping networks and directed mapping networks result in structurally similar factor-graphs in the end, we treat them on the same basis in the following.

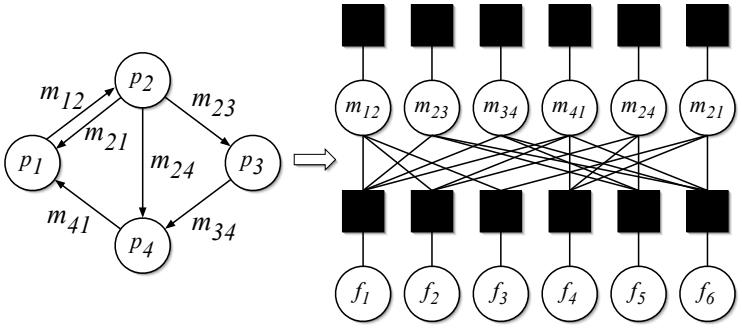

Figure 6.4: Modeling a directed network of mappings (left-hand side) as a factor-graph (right-hand side); the nodes of the factor-graph represent, from top to bottom, probability functions encapsulating *a priori* information on the mappings, variables for the attribute mappings, probability functions relating attribute mapping variables to feedback variables, and finally variables for the feedback that can be gathered from the network.

6.4 Embedded Message Passing

So far, we have developed graphical probabilistic models capturing the relations between schema mappings and network feedback in a PDMS. To take advantage of these models, one would have to gather *all* information pertaining to *all* mappings, cycles and parallel paths in a system. However, adopting this centralized approach makes no sense in our context, as PDMSs were precisely invented to avoid such centralization. Instead, we devise below a method to embed message passing into normal operations of a Peer Data Management System. Thus, we are able to get globally consistent mapping quality measures in a scalable, decentralized and parallel manner while respecting the autonomy of the peers.

Looking back at the factor-graphs introduced in Section 6.3.2 and 6.3.3, we make two observations: i) some (but not all) nodes appearing in the factor-graphs can be mapped back onto the original elements of the PDMS graph, and ii) the factor-graphs contain cycles.

6.4.1 On Feedback Variables in PDMS Factor-Graphs

Going through one of the figures representing a PDMS factor-graph from top to bottom, one may identify four different kinds of nodes: factors for the prior probability functions on the mappings, variable nodes for the soundness of the attribute mappings, factors for the probability functions linking mapping and feedback variables, and finally variable nodes for the feedback information. Going one step further, one can make a distinction between nodes representing local information, i.e., mapping factors

and mapping variables, and nodes pertaining to global information, i.e., feedback factors and feedback variables.

Mapping back local information nodes onto the PDMS is easy, as only the node from which a mapping is departing needs to store information about that mapping (see Semantic Gossiping algorithm in Section 4.6). Luckily, we can also map the other nodes rather easily, as they either contain global but *static* information (a function for feedback factors), or information gathered around the local *neighborhood* of a node (δ, observed values for $f^i_{\circlearrowleft}$ and $f^j_{\rightrightarrows}$, see preceding section). Hence, each peer p only needs to store a fraction of the global factor-graph, fraction composed of the factors and variables for its outgoing mappings and their corresponding feedback factors and variables. The algorithm for constructing a local factor-graph is given in Algorithm 6.2. A local factor-graph contains the factor and variable for all local mappings, the related feedback factors and feedback variables, and *virtual peers* p_i representing the other peers related to mappings appearing in the local feedback cycles or parallel paths. Figure 6.5 shows how p_1 from Figure 6.4 would store its local factor-graph.

Algorithm 6.2 Constructing a local factor-graph for a peer

```
/*create a variable and a factor for each outgoing mapping*/
for all outgoing mapping m do
  add m.factor to local-factor-graph;
  add m.variable to local-factor-graph;
  connect m.factor to m.variable;
  /*add a variable and a factor for each feedback*/
  for all feedback f pertaining to m do
    add f.factor to local-factor-graph;
    add f.variable to local-factor-graph;
    connect f.factor to f.variable;
    connect m.variable to f.factor;
    /*create a virtual node for all foreign peers*/
    for all mapping m' in feedback f except m do
      add virtual-peer m'.peer to local-factor-graph;
      connect m'.peer to f.factor;
    end for
  end for
end for
```

6.4.2 On Cycles in PDMS Factor-Graphs

Cycles appear in PDMS factor-graphs as soon as two mappings belong to two identical cycles or parallel paths in the PDMS. See for example the PDMS in Figure 6.3, where m_{12} and m_{41} both appear in cycles $p_1 -$

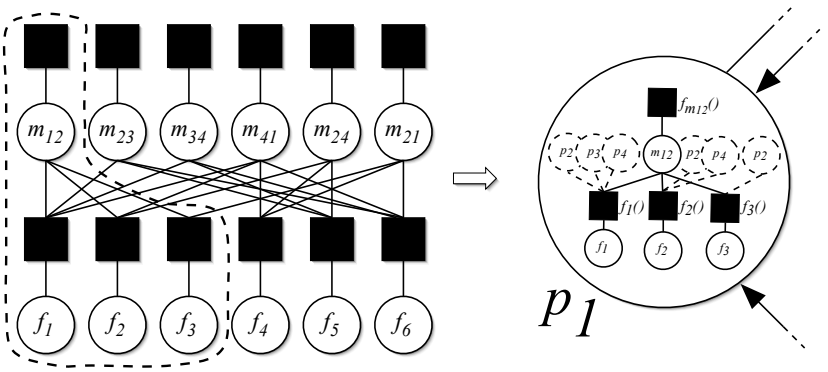

Figure 6.5: Creating a local factor-graph in the PDMS (here for peer p_1); the local factor-graph contains variables and factors for the local mappings departing from a peer, variables and factors for the feedback information related to those mappings, and virtual peers representing the other nodes involved in the cycles and parallel paths taken into consideration locally.

$p_2 - p_4 - p_1$ and $p_1 - p_2 - p_3 - p_4 - p_1$, hence creating a cycle $m_{12} - factor(f_1) - m_{41} - factor(f_2) - m_{12}$ in the factor-graph. As mentioned above, the results of the sum-product algorithm operating in a factor-graph with cycles cannot (in general) be interpreted as exact function summaries.

One well-known approach to circumvent this problem is to transform the factor-graph by regrouping nodes (*clustering* or *stretching* transformations) to produce a factor tree. In our case, this would result in regrouping all mappings having more than one cycle or parallel path in common; this is obviously inapplicable in practice, as this would imply introducing central components in the PDMS to regroup (potentially large) sets of independent peers (see also Section 6.6 for a discussion on this topic). Instead, we rely on iterative, decentralized message passing schedules (see below) to estimate marginal functions in a concurrent and efficient way. We show in Section 6.5 that those evaluations are sufficiently accurate to make sensible decisions on the mappings in practice.

6.4.3 Embedded Message Passing Schedules

Given its local factor-graph, a peer can locally update its belief on the mappings by reformulating the sum-product algorithm (Section 6.3.1); messages from variables to factors can now be divided into two types: local messages sent to local factors, and remote messages sent to distant peers appearing as virtual peers in the local factor-graph:

local message from mapping m_i to factor $f_j() \in n(m_i)$:

$$M_{m_i \to f_j()}(m_i) = \prod_{f() \in n(m_i) \setminus \{f_j()\}} M_{f() \to m_i}(m_i)$$

remote message for factor $f_k()$ from peer p_0 to peer $p_j \in n(f_k())$:

$$M_{p_0 \to f_k()}(m_i) = \prod_{f() \in n(m_i) \setminus \{f_k()\}} M_{f() \to m_i}(m_i).$$

Messages for the mapping variables can then simply be computed by combining both local messages and remote messages received from distant peers:

local message from factor $f_j()$ to mapping variable m_i:

$$M_{f_j() \to m_i}(m_i) = \sum_{\sim\{m_i\}} \left(f_j() \prod_{p_k \in n(f_j())} M_{p_k \to f_j()}(p_k) \prod_{m_l \in n(f_j()) \setminus \{m_i\}} M_{m_l \to f_j()}(m_l) \right)$$

posterior soundness of local mapping m_i:

$$P(m_i | \boldsymbol{f}) = \alpha \left(\prod_{f() \in n(m_i)} M_{f() \to m_i}(m_i)) \right)$$

where α is a normalizing constant ensuring that the probabilities of all events sum to one (i.e., making sure that $P(m_i = \mathit{sound}) + P(m_i = \mathit{unsound}) = 1$).

In cycle-free PDMS factor-graphs (i.e., trees), exact messages can be propagated from mapping variables to the rest of the network in at most two iterations (due to the specific topology of our factor-graph). Thus, all inference results will be exact in two iterations.

For the more general case of PDMS factor-graphs with cycles, we are stuck at the beginning of the computation since every peer has to wait for messages from other peers. We resolve this problem in a standard manner by considering that all peers virtually received a unit message (i.e., a message representing the unit function) from all other peers appearing in their local factor-graphs prior to starting the algorithm. From there on, peers derive probabilities on the soundness of their local mappings and send messages to other peers as described above. We show in Section 6.5 that for PDMS factor-graphs with cycles, the algorithm converges to very good approximations of the exact values obtained by a standard global inference process. Peers can decide to send messages according to different schedules depending on the PDMS; we detail below two possible schedules with quite different performance in terms of communication overhead and convergence speed.

Periodic Message Passing Schedule

In highly dynamic environments where databases, schemas and schema mappings are constantly evolving, appearing or disappearing, peers might wish to act proactively in order to get results on the semantic correctness of their mappings in a timely fashion. In a Periodic Message Passing Schedule, peers send remote messages to all peers p_i appearing in their local factor-graph every time period T. This corresponds to a new round of the iterative sum-product algorithm. This periodic schedule induces some communication overhead (a maximum of $\sum_{c_i}(l_{c_i}-1)$ messages per peer every T, where c_i represent all mapping cycles passing through the peer and l_{c_i} the length of the cycles) but guarantees our methods to converge within a given time-frame dependent on the topology of the network (see also Section 6.5). Note that T should be chosen according to the network churn in order to guarantee convergence in highly dynamic networks. Its exact value may range from a couple of seconds to weeks or months depending on the setting.

Lazy Message Passing Schedule

A very nice property of the iterative message passing algorithm is that it is tolerant to delayed or lost messages. Hence, we do not actually require any kind of synchronization for the message passing schedule; peers can decide to send a remote message whenever they want without endangering the global convergence of the algorithm (the algorithm will still converge to the same point, simply slower, see also Section 6.5). We may thus take advantage of this property to totally eliminate any communication overhead (i.e., number of additional messages sent) induced by our method by piggybacking on query messages. The idea is as follows: every time a query message is sent from one peer to another through a mapping, we append to that query message all sum-product messages pertaining to the mapping being used. In that case, the convergence speed of our algorithm is directly related to the query load of the system. This may be the ideal schedule for query-intensive or relatively static systems.

6.4.4 Prior Belief Updates

Our computations always take into account the mapping factors (top layer of a PDMS factor-graph). These factors represent any local, prior knowledge the peers might possess on their mappings. For example, if the mappings were carefully checked and validated by a domain expert, the peer might want to set all prior probabilities on the soundness of the mappings to one to ensure that these mappings will always be treated as semantically correct.

In most cases, however, the peers only have a vague idea (e.g., presupposed quality of the alignment technique used to create the mappings) on

the priors related to their mappings initially. As the network of mappings evolves and time passes, however, the peers start to accumulate various posterior probabilities on the soundness of their mappings thanks to the iterative message passing techniques described above. Actually, the peers get new posterior probabilities on the soundness of the mappings as long as the network of mappings continues to evolve (e.g., as cycles and parallel mapping paths get created, modified or deleted). Thus, peers can decide to modify their prior belief taking into account the evidences accumulated in order to get more accurate results in the future. This corresponds to learning parameters in a probabilistic graphical model when some of the observations are missing. Several techniques might be applied to that type of problem (e.g., Monte Carlo methods, Gaussian approximations). We propose in the following a simple Expectation-Maximization [DLR77] process, which works as follows:

- Initialize the prior probability on the soundness of the mapping taking into account any prior information on the mapping. If no information is available for a given mapping, start with $P(m = sound) = P(m = unsound) = 0.5$ (maximum entropy principle).

- Gather posterior evidences $P_k(m = sound|\boldsymbol{f_k})$ on the soundness of the mapping thanks to cycle analyses and message passing techniques. Treat those evidences as new observations for every change on the local factor-graphs (i.e., new, modified or lost cycle or parallel path)

- After each change affecting the local factor-graph, update the prior belief on the soundness of the mapping m given previous evidences $P_k(m = sound|\boldsymbol{f_k})$ in the following way:

$$P(m = sound) = \sum_{i=1}^{k} P_i(m = sound|\boldsymbol{f_i})k^{-1}$$

Hence, we make the prior values slowly converge to a local maximum likelihood to reflect the fact that more and more evidences are being gathered about the mappings as the mapping network evolves.

6.4.5 Introductory Example Revisited

Let us now come back to our introductory example and describe in more detail what happened. Imagine that the network of databases was just created and that the peers have no prior information on their mappings or the network. By sending probe queries with $TTL \geq 4$ through its two mapping links, p_2 detects two cycles and one parallel path, and gets all related feedback information. For the attribute *Creator*:

$$\boldsymbol{f}^{\mathbf{1+}}_{\circlearrowleft} : m_{12} \rightarrow m_{23} \rightarrow m_{34} \rightarrow m_{41}$$
$$\boldsymbol{f}^{\mathbf{2-}}_{\circlearrowleft} : m_{12} \rightarrow m_{24} \rightarrow m_{41}$$
$$\boldsymbol{f}^{\mathbf{3-}}_{\rightrightarrows} : m_{24} \| m_{23} \rightarrow m_{34}$$

δ, the probability that two or more mapping errors get compensated along a cycle, is here estimated to 1/50; if we consider that the schema of p_2 contains fifty-one attributes, and that mapping errors map to a randomly chosen attribute (but obviously not the correct one), the probability of the last mapping error compensating any previous error is 1/50, thus explaining our choice.

Let us suppose that p_2's schema is made of seven subsumption hierarchies organized as binary trees of seven attributes each (the two remaining attributes are not part of any hierarchy). p_2 estimates $\delta_{\sqsubseteq}$, the probability of an erroneous subsumption mapping on a super attribute being compensated along the cycle by another subsumption, to 10/343. This corresponds to the probability of a random mapping relating an attribute to one of its super attributes in the hierarchies. Note that both of the aforementioned estimations should take into account the (semi) automatic algorithms used to create the mappings if they are known. Also, note that the exact values of those estimated parameters have little influence on the performance of our approach (see Section 6.5.2).

p_2 constructs a local factor-graph based on the feedback information it has gathered and starts sending remote messages and calculating posterior probabilities for its local mappings according to the schedule in place in the PDMS. After a handful of iterations, the posterior probabilities on the soundness of p_2's mappings towards p_3 and p_4 converge to 0.62 and 0.14 respectively. The second mapping has been successfully detected as unsound for the given attribute, and will thus not be used to forward query q_1. The query will however reach all the databases by being correctly forwarded through $p_2 \rightarrow p_3$, $p_3 \rightarrow p_4$ and finally $p_4 \rightarrow p_1$. As the PDMS network evolves, p_2 will update its prior probabilities on the mapping toward p_3 and p_4 to 0.56 and 0.32 respectively to reflect the knowledge gathered on the mappings so far.

6.5 Performance Evaluation

We present below series of results pertaining to the performance of our approach. We proceed in two phases: We first report on sets of simulations to highlight some of the specificities of our approach before presenting results obtained on a set of real-world schemas.

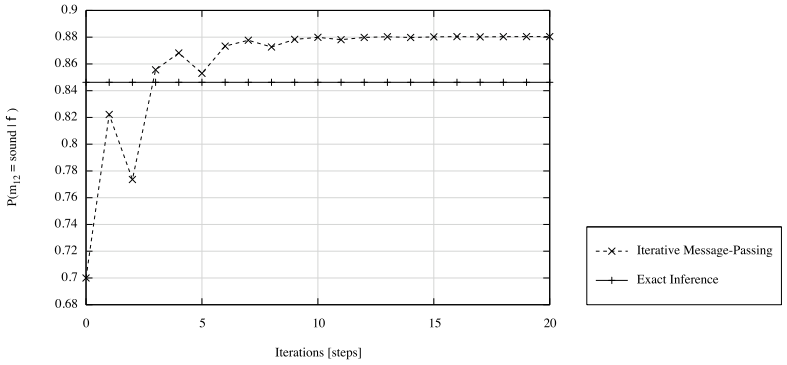

Figure 6.6: Convergence of iterative message passing algorithm (example graph, priors at 0.7).

6.5.1 Performance Analyses

Convergence

As previously mentioned, our approach is exact for cycle-free PDMS factor-graphs. For PDMS factor-graphs with cycles, our embedded message passing scheme converges to approximate results in a handful of iterations. Figure 6.6 below illustrates a typical convergence process for the example PDMS factor-graph of Figure 6.3 for small schemas of about ten attributes (i.e., δ set to 0.1), prior beliefs at 0.7 and cycle feedback as follows: f_1^+, f_2^-, f_3^-.

Note that the accuracy does not suffer from longer cycles: Figure 6.8 shows the relative error between our iterative and decentralized scheme and a global inference process. The relative error is computed for our example graph ($\delta = 0.1$, priors at 0.8, f_1^+, f_2^-, f_3^-, 10 iterations), and by successively adding a new peer as depicted in Figure 6.7. The relative error is bigger for very short cycles but never reaches 6% (those results are quite typical of iterative message passing schemes).

Cycle Length Impact

As cycles in the PDMS get bigger, so does the number of variables appearing in the feedback factors. Shorter cycles provide more precise evidences on the soundness (or unsoundness) of a mapping than longer cycles due to the inherent uncertainty pertaining to each mapping variable. Figure 6.9 demonstrates this claim for simple cyclic PDMS networks of two to twenty nodes (priors at 0.5, positive feedback, 2 iterations [cycle-free factor-graph]). The results are quite naturally influenced by the value of δ, but in the end, cycles greater than ten mappings always end up by providing very little evidence on the soundness of the mappings, even for

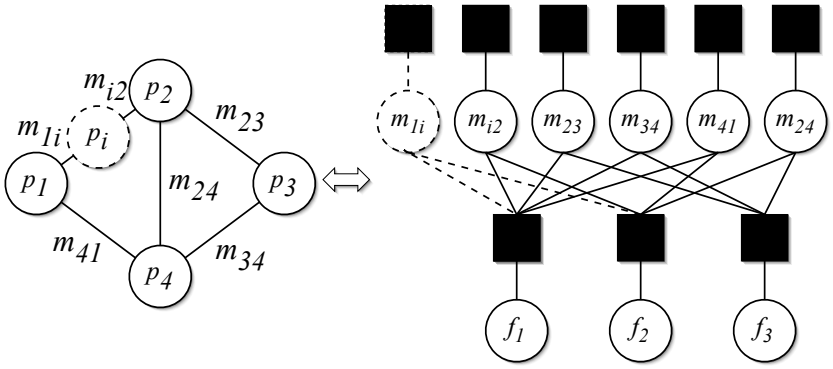

Figure 6.7: Adding nodes iteratively to increase the cycle length.

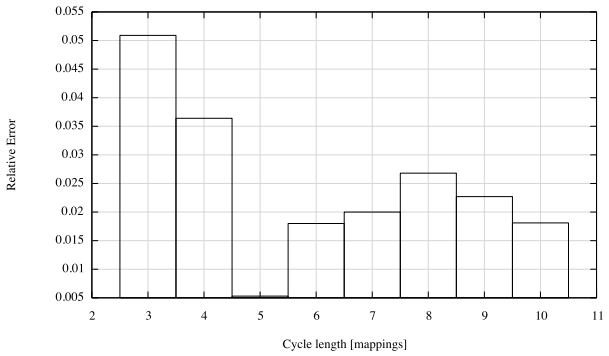

Figure 6.8: Relative error of iterative message passing algorithm for various cycle lengths (10 iterations, example graph, priors at 0.8).

bigger schemas for which compensating errors are statistically infrequent (e.g., Figure 6.9 for δ=0.01).

Scale-free networks are known to have a number of large loops growing exponentially with the size of the loops considered [BM05]. Generally speaking, this would imply exponentially growing PDMS factor-graphs as well for large-scale networks. However, as we have just seen, the impact of cycles on the posterior probabilities diminishes rapidly with the size of the cycles (see also below Section 6.5.2).

Fault-Tolerance

As mentioned earlier for the lazy message passing schedule, our scheme does not requires peers to be synchronized to send their messages. To simulate this property, we randomly discard messages during the iterative message passing schedule and observe the resulting effects. Figure 6.10

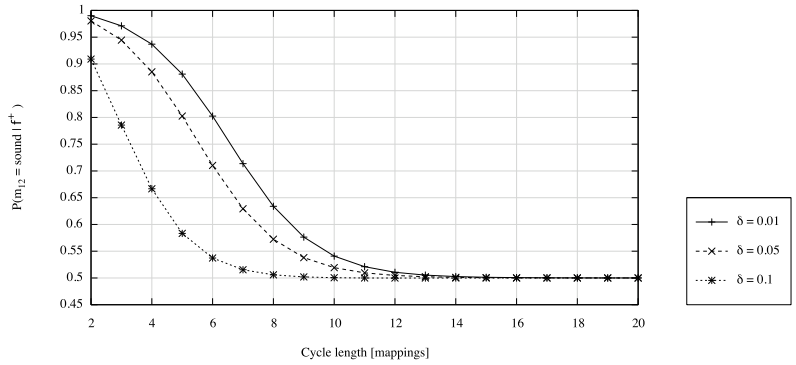

Figure 6.9: Impact of the cycle length on the posterior probability, here for a simple positive cycle graph of a varying number of mappings and for three values of δ.

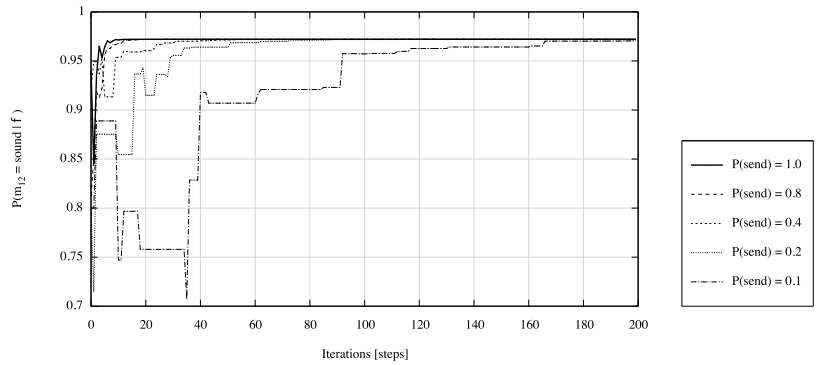

Figure 6.10: Robustness against faulty links (lost messages).

shows the results if we consider, for every message to be sent, a probability $P(send)$ to send it only (example network, $\delta = 0.1$, priors at 0.8, f_1^+, f_2^-, f_3^-). We observe that our method always converges, even for cases where 90% of the messages get discarded, and that the number of iterations required in order for our algorithm to converge grows linearly with the rate of discarded messages.

6.5.2 Performance Evaluation on Random PDMS Networks

To test our heuristics on larger networks, we create individual nodes (schemas) interlinked with edges (schema mappings) by randomly choosing a distinct pair of nodes for each undirected mapping we wish to include. We obtain irreflexive, non redundant and undirected Poisson-

distributed graphs in this manner. We randomly pick a certain proportion of mappings and create erroneous links. Also, we randomly select a given percentage of cycle feedback for which two or more errors get compensated. Finally, we run our iterative message passing heuristics on the resulting graphs and determine for each mapping whether it is sound or not (most probable value of $P(m_i|\boldsymbol{f})$). The results are given in terms of *precision* values, where precision is defined as the ratio of the number of correctly evaluated mappings over the total number of mappings evaluated. As our analysis takes into account the correlation of the cycles in a graph, the results can vary substantially between two random graphs generated using the same parameters. In the following, each result is given as an average calculated over twenty consecutive runs, with a confidence interval corresponding to a confidence level of 95%.

Performance with an Increasing Proportion of Unsound Mappings

Figure 6.11 provides results corresponding to networks of 50 schemas and 200 mappings, with an increasing percentage of unsound mappings and for relatively small schemas ($\delta = 5\%$). Our methods work surprisingly well for low densities of unsound mappings, with 98% or more of correct decisions for networks with less than 30% of erroneous mappings. For networks with a larger proportions of unsound mappings, the results are less spectacular but still satisfying with precision values above 60%. Note that those values are obtained automatically, via a totally decentralized process and without any prior information on the mappings. Compensating errors make it difficult to detect all errors in networks with many erroneous mappings (cycles which should be treated as negative are in fact seen as positive). This fact is highlighted by a second curve in Figure 6.11 ($\delta = 0$), corresponding to very large schemas, where compensating errors can be neglected and where it is much easier to make sensible decisions on networks with very high proportions of unsound mappings.

Precision with an Increasing Number of Mappings

Figure 6.12 provides results corresponding to networks of 50 schemas and an increasing number of mappings between the schemas. For sparse networks (e.g., 50 mapping links, corresponding to one mapping per schema on average), few cycles can be detected and thus little feedback information is available. As more and more mappings are created, more feedback information gets available, thus making it easier to take sensible decisions on the soundness of the mappings (which corroborates the results obtained in Section 5.2). Social networks have a very high number of long cycles (e.g., in scale-free networks, where the number of large loops grows exponentially with the size of the loops considered [BM05]); the longer the cycle, however, the less interesting it is from an inference point of

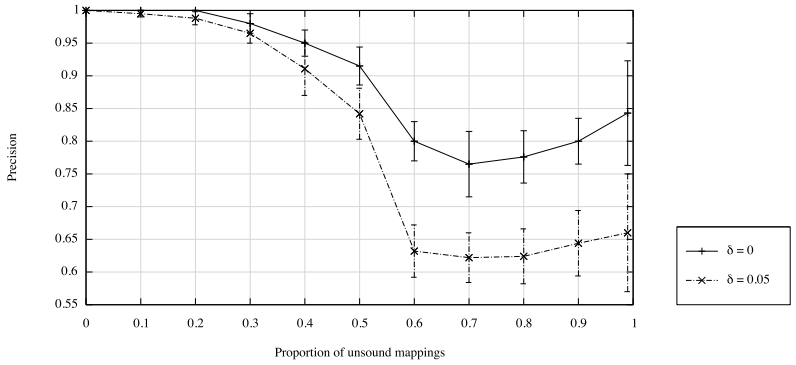

Figure 6.11: Precision of mapping evaluation on random networks of 50 schemas and 200 mappings, with a varying proportion of erroneous mappings, two values of δ, TTL = 5 to detect cycles and without any a priori information

view as it is related to a higher number of mapping variables (and hence represents less precise information, see above Figure 6.9).

Thus, peers should always be cautious to analyze the most pertinent feedback information only, pertaining to cycles or parallel paths as small as possible, and to keep their TTL for detecting cycles and parallel paths relatively small; to highlight this fact, Figure 6.12 shows two curves: one for a fixed TTL of 5 and one with an adaptive TTL (6 for 50 to 100 mappings, 5 for 150 to 200 mappings and 4 from 250 mappings). Adapting the TTL value is important in two respects: first, in sparse networks where peers should try to detect longer cycles in order to get more feedback information (e.g. for 100 mappings in Figure 6.12, where a TTL of 6 leads to better results than a TTL of 5). In very sparse networks, however, there are simply too few mappings to detect a sufficient number of cycles, even for large TTL values (e.g., for 50 mappings in Figure 6.12). Second, in dense networks, where precious information given by short cycles can rapidly be diluted by taking into account longer cycles (e.g., for 300 mappings in Figure 6.12, where more than 20000 cycles of length 5 can be discovered, leading to poorer results if all taken into account). From a local perspective, peers should thus start with low TTL values and increase their TTL only when very few cycles are discovered. This also ensures the scalability of our approach: peers can concentrate on their direct vicinity and do not need to analyze the network as a whole.

Performance with an Increasing Proportion of Verified Mappings

We expect a fraction of the peers to have some *a priori* information on their mappings in practice. As we take into account the correlation of the

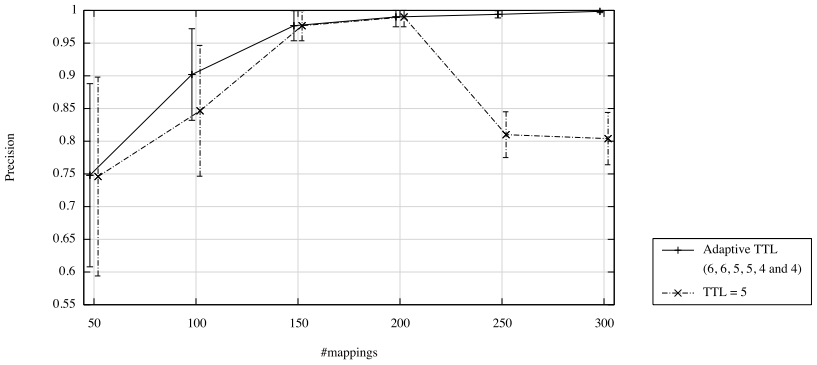

Figure 6.12: Precision of mapping evaluation on random networks of 50 schemas and a varying number of mappings, with a proportion of 20% of erroneous mappings, $\delta = 0.05$, without any a priori information but with different TTL values for detecting the cycles

various cycles, prior information on a few mappings can have quite some influence on the performance of the whole decision process. Figure 6.13 shows the evolution of the precision of our approach for a random network of 50 nodes and 150 mappings ($\delta = 0.05$), and for a varying proportion of verified mappings. All verified mappings have their prior probability set to one if they are sound, or to zero if they are unsound. We produce two curves for the figure: one with 20% and on with 40% of unsound mappings on average. As shown in the figure, even a small proportion of verified mappings can have a substantial impact on the performance of our algorithm, e.g., for a proportion of 40% of unsound mappings and 10% of verified mappings reducing the number of wrong decisions by more than 50%.

Sensitivity to Estimated Parameter

Finally, note that our approach is rather insensitive to the estimation of δ, the probability of having multiple errors to compensate each other in a series of mappings. Generally speaking, it is important to consider a small and positive value of δ to take into account all possible combinations of sound/unsound mappings in the cycles. The exact value of δ is however of little significance for relatively large schemas, as shown in Figure 6.14. The figure shows the evolution of the precision of our approach in a network of 50 schemas and 150 mappings, for various estimations of δ. The figure was generated with a real δ of 0.05, and for two proportions of erroneous mappings. As shown in the figure, considering a δ of zero leads to suboptimal results. On the other hand, approximated values of δ lead to similar results up to errors of 100% on the real value.

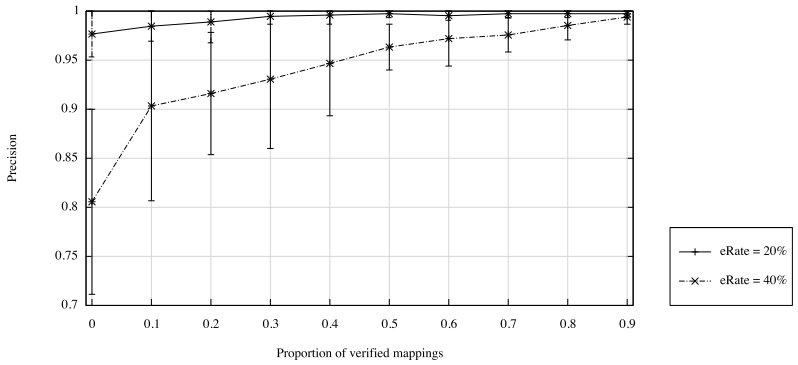

Figure 6.13: Precision of mapping evaluation on random networks of 50 schemas and 150 mappings, with a varying proportion of mappings verified by the peers, $\delta = 0.05$, and two initial error rates on the mappings.

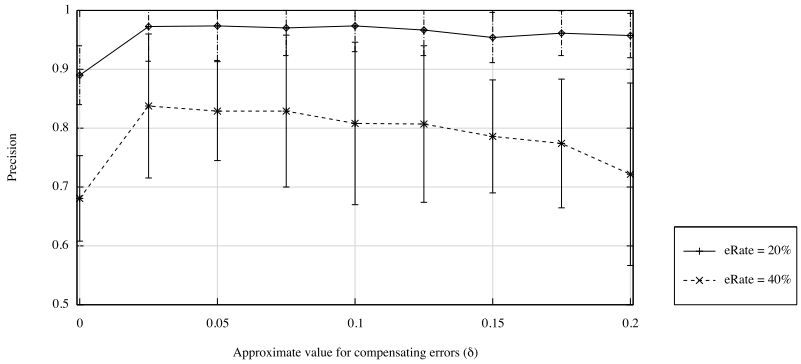

Figure 6.14: Precision of mapping evaluation on random networks of 50 schemas and 150 mappings, with various estimations of δ, a real δ of 0.05, and three initial error rates on the mappings.

6.5.3 Applying Message Passing on Real-World Schemas

We have developed a tool to test our methods on real-world schemas and mappings. This tool can import OWL schemas (serialized in RDF/XML) and simple RDF mappings (following the format introduced in the context of the Knowledge Web project [Bou04]). Our tool can also automatically create new mappings using some generic alignment techniques based on object and attribute names, edit/substring distances and aggregation [Euz04a]. Once schemas and alignments have been defined, the tool creates peers, automatically detects cycles, transforms the setting into a PDMS factor-graph and starts sending messages in order to get posterior quality values on the mappings as described above.

We report on controlled experiments based on a set of standard schemas taken from the EON Ontology Alignment Contest[1]. The setting is as follows: we first several schemas related to bibliographic data: the so-called reference schema (called 101 in the EON contest), a similar schema but translated into French (221), the Bibtex schema from M.I.T., the Bibtex schema from UMBC, and another bibliographic schemas from INRIA. Each of those schemas is composed of about thirty concepts (attributes). We then start our automatic alignment techniques to create mappings between this set of schemas. We create series of mappings as follows: $101 \rightarrow 221 \rightarrow MIT \rightarrow UMBC \rightarrow Karlsruhe \rightarrow 101$, $101 \rightarrow Karlsruhe \rightarrow UMBC \rightarrow MIT \rightarrow 221 \rightarrow 101$, $221 \rightarrow UMBC$, $UMBC \rightarrow 101$ and finally $Karlsruhe \rightarrow MIT$. We launch the iterative message passing algorithm on the resulting PDMS factor-graph and try to detect erroneous mappings automatically.

The resulting PDMS contains 396 generated attribute mappings. The attribute mappings each map one attribute of the target schema onto one attribute of the source schemas. We have at our disposal correct (i.e., sound and complete) mappings between the so-called reference schema and the other schemas[2]. Furthermore, we manually created correct mappings for the other pairs of schemas related in our setting. Comparing our reference mappings with the results of our message-passing process, we observe that our message passing algorithm evaluates 82% of the mappings correctly. Figure 6.15 shows the precision of our approach for various thresholds τ used to determine whether a mapping is sound or not (i.e., m_i is considered as being sound if $P(m_i = sound \geq \tau)$). We had no prior information on the mappings and set δ to 1/10. For all threshold values, our method is significantly better than random guesses. Those preliminary results are quite encouraging, considering that no prior information was provided on the mappings, and that only one complete round of the algorithm could be completed on this static network (i.e.,

[1] *http://co4.inrialpes.fr/align/Contest/*

[2] see *http://oaei.ontologymatching.org/2004/Contest/301/refalign.html* for an example of mapping

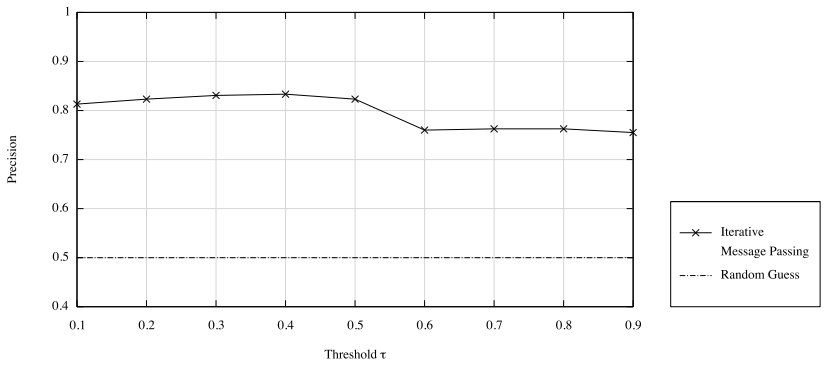

Figure 6.15: Precision results of the Message Passing approach on a real set of schemas with 396 automatically generated mappings, with a varying threshold τ to determine the set of sound mappings.

no update on prior beliefs to iteratively differentiate the sound from the unsound mappings).

6.6 Conclusions

As distributed database systems move from static, controlled environments to highly dynamic, decentralized settings, we are convinced that correctly handling uncertainty and erroneous information is becoming a key challenge for improving the overall quality of query answering schemes. The vast majority of approaches are today centered around local and deductive methods which seem quite inappropriate to maximize the performance of systems that operate without any form of central coordination. Contrary to these approaches, we considered in this chapter an abductive, non-monotonic reasoning scheme which reacts to observations or inconsistencies by propagating belief in a decentralized way. Our approach is computationally efficient as it is solely based on sum-product operations.

As future work involving probabilistic message passing, we are particularly interested in understanding the relation between exact inference and our approximate results. We are currently analyzing the computational overhead and scalability properties of other inference techniques (e.g., generalized belief propagation [YFW00], or techniques constructing a junction tree in a distributed way [PG04a]). Also, as global interoperability is always challenged by the dynamics of the mapping network, we plan to analyze the tradeoff between the efforts required to maintain the probabilistic network in a coherent state and the potential gain in terms of relevance of results. Probabilistic message passing is currently being implemented in our semantic overlay network (GridVine, see Chapter 8).

As part of this implementation effort, we are also extending our method to handle result analysis (see Chapter 4) in a similar manner, by attaching additional feedback variables to the factor-graphs.

Chapter 7

Analyzing Semantic Interoperability in the Large

In previous chapters, we introduced series of methods to capture and foster interoperability by analyzing transitive closures of mapping operations and modeling semantics as dynamic agreements between sets of heterogeneous parties. In this chapter, we take a step back and analyze the semantic mediation layer in a holistic way using graph theoretic tools. We model PDMSs as graphs, derive a necessary condition to foster semantic interoperability in the large [CMA04] and test our heuristics in the context of an existing bioinformatic portal with hundreds of schemas.

7.1 Introduction

Information systems are undergoing profound changes with the wide adoption of semi-structured data standards like XML or RDF. Those specifications aim at providing machine-processable information and should underpin the creation of systems where data are given well-defined semantics. In Chapter 3 and 4, we introduced the Peer Data Management System architecture and Semantic Gossiping as new ways of reconciling semantically heterogeneous domains in an evolutionary and completely decentralized manner. We have shown (Chapters 5 and 6) that sets of uncertain, pair-wise, and local schema mappings can be sufficient for creating a global self-healing semantic network where semantically correct translations get reinforced.

Even if much effort has recently been devoted to the creation of sophisticated schemes to relate pairs of schemas or ontologies through mappings [RB01], it is however still far from being clear how such large-scale semantic systems evolve or how they can be characterized. For example, even if a lack of schema mappings clearly limits the quality of the overall semantic consensus in a given system, the exact relationships between the former and the latter are unknown. Is there a minimum number

of mappings required to foster semantic interoperability in a network of information sharing parties? Given a large set of schemas and schema mappings, can we somehow predict the impact of a query issued locally?

This chapter represents a first attempt to look at the problem from a macroscopic point of view. Our contribution is two-fold: first, we develop a model capturing the problem of semantic interoperability in large scale decentralized environments. Second, we identify recent graph theoretic results and show how they can be extended to be applicable to our problem. More specifically, we derive a necessary condition to foster semantic interoperability in the large and present a method for evaluating the degree of propagation of a query issued locally. Also, we give an evaluation of our methods applied on a real graph representing several hundreds of interconnected bioinformatic schemas. The rest of this chapter is organized as follows: we start by introducing a general layered representation for distributed semantic systems. Section 7.3 is devoted to the formal model with which we analyze semantic interoperability in the large. The main theoretical results related to semantic interoperability and semantic component sizes are detailed in Section 7.4 and Section 7.5. Section 7.6 explores weighted graphs, while Section 7.7 describes our findings related to the analysis of a real bioinformatic semantic network. Finally, we discuss practical applications of our main results from a decentralized perspective before concluding.

7.2 The Model

Large-scale networks are traditionally represented by a graph. In our case, however, a single graph is insufficient to accurately model the relationships between both the systems and their schemas. In Section 3.2, we introduced a three-layer model for semantic overlay networks. Below, we present a set of graphs capturing the organization of the two upper layers of that model.

As for the previous chapters, we model information parties as peers that are related to each other physically (denoted as the Overlay Layer in Figure 3.4). The *Peer-to-Peer* model described hereafter captures the organization of the peers. Peers use various schemas or ontologies to structure their data. We define a *Peer-to-Schema model* to represent the relation between the peers and the schemas. Finally, schemas themselves can be related through schema mappings captured by the *Schema-to-Schema model* (also called Semantic Mediation Layer in Chapter 3). Each of those models represents a distinct facet of the overall Peer Data Management System and can be decorrelated from the other two (as, for example, in the *GridVine* system described in Chapter 8).

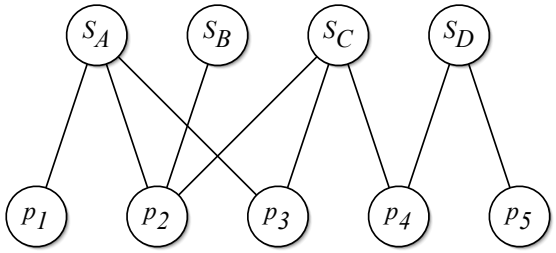

Figure 7.1: The Peer-to-Schema model is a bipartite graph representing the various schemas used by the peers to structure their data.

7.2.1 The Peer-to-Peer Model

Peers represent autonomous parties producing and consuming information in a system. Each peer $p \in \mathcal{P}$ has a basic communication mechanism that allows it to establish connections with other peers. The Peer-to-Peer model captures the organization of the peers through that communication mechanism. The communication mechanism is such that it should allow any peer to contact any other peer in the system – either by broadcasting (Gnutella) or by using a central (Napster), hierarchical (DNS) or decentralized (P-Grid) registry. Furthermore, we assume that the information and meta-information (i.e., metadata, schemas and schema mappings) available in the system are all indexed in a similar way, allowing a peer to retrieve any resource independently of its exact nature.

7.2.2 The Peer-to-Schema Model

We assume that peers store annotations (or metadata) related to the resources available on their system. Each peer $p \in \mathcal{P}$ organizes its local database DB_p according to a set of schemas $\boldsymbol{S}_p \in \mathcal{S}$. When a peer p organizes (part of) its database following a schema S_i, we say that p is in the *semantic domain* of S_i: $p \leftrightarrow S_i$. Individual schemas are uniquely identified throughout the network and may be used by different peers. We represent the relation between the peers and the schemas through a bipartite graph where the vertices are divided into two disjoints sets $\mathcal{V}_1 \equiv \mathcal{P}$ and $\mathcal{V}_2 \equiv \mathcal{S}$ representing respectively the set of peers and the set of schemas. An edge in that graph denotes the fact that a peer uses a certain schema to structure (part of) its data. Figure 7.1 represents such a bipartite Peer-to-Schema graph where p_3 annotates data according to schemas S_A and S_C.
We do not make any assumption on the languages used to express the meta-data or schemas. Peers can for example use different mechanisms (e.g., XML Schema elements or RDFS/OWL classes) for categorizing resources. However, all peers should be able to locally issue queries $q_i \in \mathcal{Q}$

against their databases using standard query operators in order to retrieve sets of specific resources.

7.2.3 The Schema-to-Schema Model

Finally, we allow peers to create mapping links between schemas. We do not put any constraint on the origin of the mappings: they might be automatically generated, written by domain experts, partially wrong, and may be provided by any peer, regardless of the schemas it uses for its own database. As for the previous chapters, we use $\mu_{S_1 \rightarrow S_2}$ to denote a mapping μ relating two schemas S_1 and S_2. Using a mapping $\mu_{S_1 \rightarrow S_2}$, a peer $p_1 \leftrightarrow S_1$ may reformulate a local query q on its database DB_{p_1} into a transformed query q' applicable to a second semantic domain S_2:

$$\mu_{S_1 \rightarrow S_2}(q(S_1)) = q'(S_2), p_1 \leftrightarrow S_1 \wedge p_2 \leftrightarrow S_2$$

Note that multiple transformations may be applied to a single query q. The composition of multiple transformations $\mu_1, \ldots, \mu_n$ is given by using the associative composition operator (specific to a given approach) $\circ$ as follows:

$$\mu_n \circ \ldots \circ \mu_1(q)(S).$$

From a graph modeling perspective, schema mappings may be viewed as edges interconnecting schema nodes. Figure 7.2 depicts a Schema-to-Schema graph. Note that the edges have to be directed in order to capture the peculiarities of the mapping operations, since mapping functions may not be invertible and since the properties of even the most basic translations can be dependent on the direction with which they are applied (e.g., relations between subclasses and super-classes, see the preceding chapter). Also, note that a growing number of schemes use a metric to characterize the quality of the various mapping operations encapsulated by the translation links [CFMR04, ZLFH04] (see also Chapters 4 and 6). The resulting graph is therefore a weighted directed multigraph, i.e., a directed graph with (possibly) multiple, weighted edges (mapping links) between two vertices (schemas).

7.3 Semantic Interoperability In the Large

The rest of this chapter is devoted to the study of interoperability in our PDMS setting, mainly through the analysis of a derived version of the Schema-to-Schema graph. A peer $p_i \leftrightarrow S_j$ may send a query to any peer in its own semantic domain, i.e., to any peer $p_k \in \mathcal{P} \mid p_k \leftrightarrow S_j$ in the Peer-to-Schema model (supposing, again, that the Peer-to-Peer model allows peers to contact any other peer in the network). The query may also be forwarded to peers in foreign semantic domains $S_l \neq S_j$ as long as there

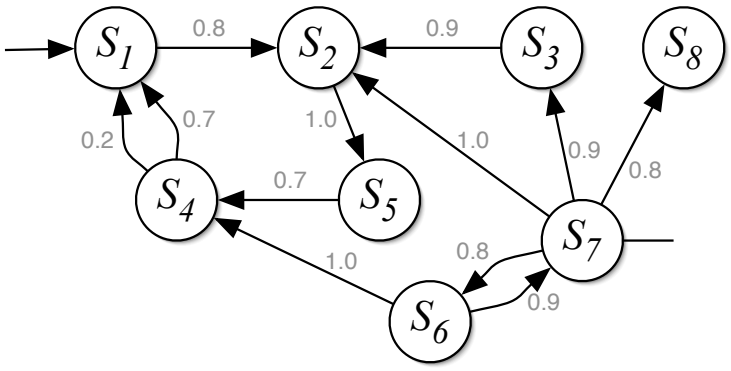

Figure 7.2: The Schema-to-Schema model is a weighted directed multigraph connecting nodes (schemas) through edges (schema mappings).

exists a mapping $\mu_{S_j \to S_l}(q)$ or a series of mappings $\mu_{S_{n-1} \to S_n} \circ \ldots \circ \mu_{S_j \to S_l}$ to reformulate the query adequately. Generalizing that statement, we introduce the notion of semantic interoperability for our context:

Definition (Semantic Interoperability) Two peers are said to be *semantically interoperable* if they can forward queries to each other, potentially through series of schema mappings.

Note that the aforementioned definition does not characterize the quality of the semantic interoperability in any way; it simply acknowledges the existence of some semantic relationship between two peers on the basis of a mapping path. If no semantic path exists to forward the query, we say that the two peers in question are *semantically unreconcilable*.

7.3.1 Semantic Connectivity

Analogously to physical network analysis, we define an intermediary layer accounting for the semantic connectivity of the system. Indeed, considering the definition given above, we can slightly relax our Schema-to-Schema model when analyzing semantic interoperability:

Unweighted model: Since our definition of semantic interoperability is based on the presence or absence of mapping links, we ignore the weights in the Schema-to-Schema model.

No duplicate edges: From a vertex strong connectivity point of view, duplicate edges between two vertices play no role. Thus, directed multigraphs may be replaced by their corresponding digraphs.

However, when analyzing semantic connectivity graphs, one has to account for two important specificities of large-scale semantic systems:

High clustering: Sets of schemas related to a given domain of expertise tend to organize themselves tightly and thus share many mapping links, while being largely disconnected from schemas describing other domains. The clustering coefficient measures the degree to which the neighbors of a vertex are neighbors of each other. Hence, we expect clustering coefficients in large-scale semantic graphs to be particularly high.

Bidirectional edges: Even if mappings used in the translation links are essentially unidirectional, we can expect domain experts to create translations in both directions (to and from a given schema) in order to foster semantic interoperability. Thus, a (potentially high) fraction of the mappings can be considered as bidirectional in our connectivity analysis.

Taking into account the points exposed above, we can finally propose our formal model for the study of semantic interoperability:

Definition (Semantic Connectivity Graph) A *Semantic Connectivity Graph* is a pair $(\boldsymbol{S}, \boldsymbol{M})$ where

- $\boldsymbol{S}$ is the set of schemas in a large-scale semantic system
- $\boldsymbol{M}$ is a non-redundant, irreflexive set of ordered pairs $(S_i, S_j) \mid i \neq j \wedge S_i, S_j \in \boldsymbol{S}$, each denoting a mapping between two schemas.

Using this formalism, semantic systems can be represented by digraphs where $\boldsymbol{S}$ is a set of vertices and $\boldsymbol{M}$ a set of directed edges. A couple of statistical properties derived from the semantic connectivity graphs will be of particular interest in our upcoming analysis:

- The probabilities p_{jk} that a randomly chosen vertex has in-degree j and out-degree k.
- The *clustering coefficient cc* defined as the average number of edges of a node's neighbors connecting to other neighbors of the same node.
- The *bidirectional coefficient bc* defined as the average fraction of edges that can be considered as bidirectional, i.e., the fraction of mappings $(S_i, S_j) \in \boldsymbol{M} \mid \exists (S_j, S_i) \in \boldsymbol{M}$.

Remembering that a directed graph is strongly connected if it has a path from each vertex to every other vertex, one can easily determine wether or not a set of peers is semantically interoperable by inspecting the semantic connectivity graph:

Theorem 7.1. *Peers in a set* $\boldsymbol{P}_s \subseteq \mathcal{P}$ *cannot be* semantically interoperable *if* $\boldsymbol{S}_s \subseteq \mathcal{S}$ *is not strongly connected, with* $\boldsymbol{S}_s \equiv \{S \mid \exists p \in \boldsymbol{P}_s, p \leftrightarrow S\}$.

Proof If $\boldsymbol{S}_s$ is not strongly connected, there exists at least one vertex $S_l \in \boldsymbol{S}_s$, which cannot be reached from another vertex $S_j \in \boldsymbol{S}_s$. This means that a peer $p_i \in \boldsymbol{P}_s$, $p_i \leftrightarrow S_j$ is semantically unreconcilable with a second peer $p_k \in \boldsymbol{P}_s$, $p_k \leftrightarrow S_l$, and thus the set of peers is not semantically interoperable.

■

As a corollary, a network of peers is *globally* semantically interoperable if its semantic connectivity graph is strongly connected. This property may be satisfied in a wide variety of topologies. Introducing $|\boldsymbol{S}_s|$ and $|\boldsymbol{M}_s|$ as (respectively) the number of vertices and edges in the semantic connectivity subgraph related to a set of peers $\boldsymbol{P}_s \subseteq \mathcal{P}$, we can immediately derive two bounds on the number of mapping links affecting the semantic interoperability:

Observation 7.1. *A set of peers $\boldsymbol{P}_s \subseteq \mathcal{P}$ cannot be semantically interoperable if $|\boldsymbol{M}_s| < |\boldsymbol{S}_s|$.*

Observation 7.2. *A set of peers $\boldsymbol{P}_s \subseteq \mathcal{P}$ is semantically interoperable if $|\boldsymbol{M}_s| > |\boldsymbol{S}_s|(|\boldsymbol{S}_s| - 1) - (|\boldsymbol{S}_s| - 1)$.*

The proofs of those two observations are immediate.

7.4 A Necessary Condition for Semantic Interoperability

7.4.1 Undirected Model

Real world graphs usually develop by following preferential attachment laws and exhibit properties (e.g., small-world, scale-free) specific to their statistical distribution. Thanks to recent advances in graph theory, it is now possible to study arbitrary large graphs based on their degree distribution. However, there exists no model taking into account all the specificities of our semantic connectivity graph. In the following, we derive new results from the framework introduced by Mark Newman [NSW01] to account for those specificities. Since we do not assume readers to be generally familiar with *generatingfunctionologic* graph theory, we start by introducing a simpler, undirected model before presenting the directed one. Note that this undirected setting can be applied in many practical settings where mappings are bidirectional (see for example Section 6.3.2 or Chapter 8).

Our approach is based on generating functions [Wil94]; first, we introduce a generating function for the degree distribution of a semantic connectivity graph:

$$G_0(x) = \sum_{k=0}^{\infty} p_k x^k \tag{7.1}$$

where p_k is the probability that a randomly chosen vertex has degree k. This function encapsulates all the information related to the degree distribution of the graph, since

$$p_k = \frac{1}{k!} \frac{d^k G_0}{dx^k}\bigg|_{x=0}. \tag{7.2}$$

Theorem 7.2. *Peers in a set* $\boldsymbol{P}_s \subseteq \mathcal{P}$ *cannot be* semantically interoperable *if* $\sum_k k(k-2-cc)p_k < 0$*, with* p_k *the probability that a node has degree* k *in the undirected semantic connectivity graph of the set and* cc *the clustering coefficient.*

Proof The average number of neighbors of a node is

$$z_1 =< k >= \sum_k k p_k = G_0'(1).$$

If we follow a randomly chosen edge, we arrive at a vertex with probability proportional to the degree of that vertex, i.e., proportional to kp_k. The correctly normalized degree distribution of the node we arrive at is

$$\frac{\sum_k k p_k x^k}{\sum_k k p_k} = x \frac{G_0'(x)}{G_0'(1)}. \tag{7.3}$$

If we start at a randomly chosen vertex and follow all the edges from that vertex to get to the set of direct neighbors, each of those first-order neighbors will have a degree distribution given by Equation 7.3. Now, if we want to count the number of second-order neighbors from the original node we started at, we can consider the first-order neighbors as being one degree lower, since we do not want to take into account the edge connecting our original node to the first-order neighbor. Similarly, we can subtract on average cc degrees of the first-order neighbors to account for those links which connect first-order neighbors together. In the end, the distribution of the number of second-order neighbors we get from a first-order neighbor is

$$G_1(x) = \frac{1}{x^{cc}} \frac{G_0'(x)}{G_0'(1)} = \frac{1}{z_1} \frac{1}{x^{cc}} G_0'(x). \tag{7.4}$$

The probability distribution of the number of second-order neighbors is then obtained by multiplying Equation 7.4 by the probability of the original node having k first-order neighbors and by summing over those k neighbors. Remembering that the distribution of a generating function summed over m realizations is generated by the m^{th} power of that generating function [Wil94], we get

$$\sum_k p_k [G_1(x)]^k = G_0(G_1(x)).$$

The average number of second order neighbors is

$$z_2 = \left[\frac{d}{dx} G_0(G_1(x))\right]_{x=1} = G_0'(G_1(1))G_1'(1)$$
$$= G_0'(1)G_1'(1) = \sum_k k(k-1-cc)p_k$$

since $G_1(1) = 1$.

A necessary condition for a graph to be strongly connected is the emergence of a giant component connecting most of its vertices. It has been shown [NSW01] that such a component can only appear if the number of second-order neighbors of a graph is on average greater or equal than the number of first-order neighbors. Presently, if

$$z_2 \geq z_1 \Leftrightarrow \sum_k k(k-1-cc)p_k \geq \sum_k kp_k \Leftrightarrow \sum_k k(k-2-cc)p_k \geq 0. \quad (7.5)$$

If the condition in Equation 7.5 is not satisfied, the undirected semantic connectivity graph cannot be strongly connected, and thus the set of peers cannot be semantically interoperable.

■

We term $\sum_k k(k-2-cc)p_k$ *connectivity indicator ci*. Figure 7.3 below compares this indicator (a) with the size of the biggest connected component in a random, undirected semantic connectivity graph of 10000 vertices with a variable number of edges (b). Edges are generated randomly (each pair of distinct vertices has the same probability of being connected). We notice that *ci* is a very good indicator of the overall connectivity of a semantic graph, i.e., the indicator reaches zero precisely at the percolation threshold: the graph is in a *sub-critical* phase when $ci < 0$ (no giant connected component) while it is in a super-critical phase when $ci > 0$ (when a giant connected component suddenly starts to appear).

7.4.2 Directed Model

We now turn to the full-fledged, directed model based on the semantic connectivity graph. Our methodology is similar to the one used above for the undirected case. Remember that p_{jk} is the probability that a randomly chosen vertex has in-degree j and out-degree k in our semantic connectivity graph. We introduce $\mathcal{G}(x, y)$, a generating function for the joint probability distribution of in and out-degrees:

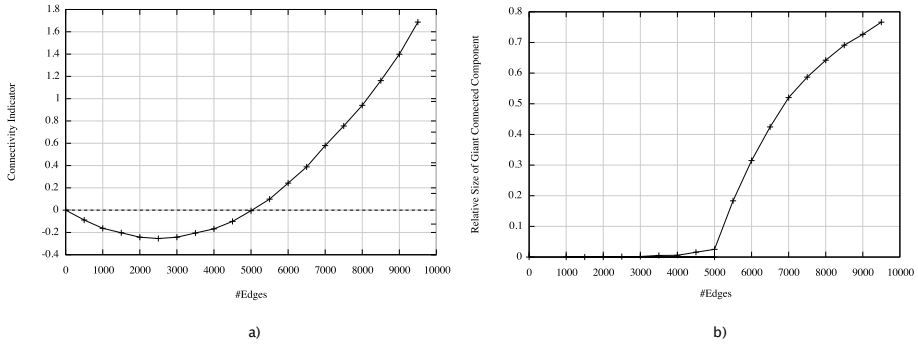

Figure 7.3: Connectivity Indicator (a) and maximal connected cluster size (b) for a random network of 10000 vertices and a varying number of edges.

$$\mathcal{G}(x,y) = \sum_{j,k} p_{jk} x^j y^k$$

which has to satisfy

$$\sum_{jk} (j-k) p_{jk} = 0$$

since every edge leaving some vertex has to enter another. This also implies that the average degree (both in and out) z_1 of vertices in the graph is

$$z_1 = \sum_{jk} j p_{jk} = \sum_{jk} k p_{jk} = \left.\frac{\delta \mathcal{G}}{\delta x}\right|_{x,y=1} = \left.\frac{\delta \mathcal{G}}{\delta y}\right|_{x,y=1}. \tag{7.6}$$

The joint probability p_{jk} is given by

$$p_{jk} = \left.\frac{1}{j!k!} \frac{\delta^{j+k} \mathcal{G}}{\delta^j x \delta^k y}\right|_{x=0,y=0}.$$

Again, the generating function encapsulates all the information contained in the discrete probability distribution p_{jk}.

Theorem 7.3. [Necessary condition for semantic interoperability] *Peers in a set $\boldsymbol{P}_s \subseteq \mathcal{P}$ cannot be* semantically interoperable *if $\sum_{j,k}(jk - j(bc+cc) - k)p_{jk} < 0$, with p_{jk} the probability that a node has in-degree j and out-degree k in the semantic connectivity graph of the set, bc the bidirectional coefficient and cc the clustering coefficient.*

Proof The function generating the number of outgoing edges leaving a randomly chosen vertex is

$$G_0(y) = \mathcal{G}(1, y).$$

If we follow an edge chosen randomly, we arrive at a vertex with a probability proportional to the in-degree of that vertex. Normalizing on the degree distribution of that vertex, we obtain:

$$\frac{\sum_{jk} j p_{jk} y^k}{\sum_{jk} j p_{jk}} = x \frac{\delta \mathcal{G}}{\delta x}\bigg|_{x=1} \left(\frac{\delta \mathcal{G}}{\delta x}\bigg|_{x,y=1} \right)^{-1}. \tag{7.7}$$

If we start at a randomly chosen vertex and follow each of the edges at that vertex to reach the k nearest, first-order neighbors, then the vertices we arrive at have a distribution of outgoing edges generated by 7.7, less one power of x to account for the edge that we followed. Thus, the distribution of outgoing edges after having followed a random edge is generated by the function

$$\mathcal{G}_1(y) = \frac{\delta \mathcal{G}}{\delta x}\bigg|_{x=1} \left(\frac{\delta \mathcal{G}}{\delta x}\bigg|_{x,y=1} \right)^{-1} = \frac{1}{z_1} \frac{\delta \mathcal{G}}{\delta x}\bigg|_{x=1}$$

where z_1 is, as above, the average vertex degree. We can now determine the distribution of second-order neighbors by summing this expression over the probabilities of a node to have k outgoing edges, but we have to be careful of two facts:

1. Some of the edges leaving a first-order neighbor connect to other first-order neighbors (clustering effect). In our model, this occurs on average cc times for a given vertex. We should not to take these nodes into account when counting the number of second-order neighbors.

2. The edge going from our initial node to a first-order neighbor might be bidirectional. This happens with a probability bc in our model. We must subtract this edge from the number of outgoing edge of a first-order neighbor when it occurs.

Consequently, the distribution of outgoing edges from first to second-order neighbors is

$$G_1(y) = (1 - bc) \frac{1}{y^{cc}} \mathcal{G}_1(y) + bc \frac{1}{y^{cc+1}} \mathcal{G}_1(y).$$

As for the undirected case, the average number of second-order neighbors is

$$z_2 = G_0'(1) G_1'(1).$$

Finally, the condition $z_2 > z_1$ yields to

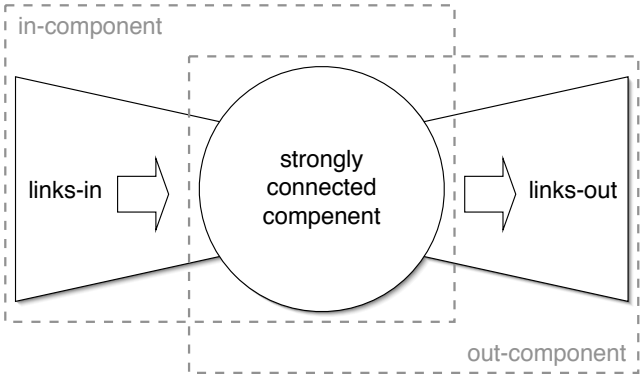

Figure 7.4: The "bow-tie" diagram representing the giant component of a directed graph.

$$\sum_{j,k}(jk - j(bc + cc) - k)p_{jk} > 0. \tag{7.8}$$

■

Equation 7.8 marks the phase transition at which a giant component appears in a semantic connectivity graph. By neglecting the bidirectional and the clustering coefficient ($bc, cc = 0$) and reorganizing the terms using Equation 7.6 we fall back on the equation for the appearance of a giant component in a directed graph derived by Mark Newman [NSW01]. Neglecting these two terms has of course a negative influence on the precision of our method (e.g., in highly clustered settings, where links connecting first-order neighbors should not be taken into account for deriving the phase transition).

In a directed graph, the giant component can be represented using a "bow-tie" diagram [BKM+] as in Figure 7.4: The *strongly connected component* represents the portion of the graph in which every vertex can be reached from each other, while the *links-in* and *links-out* respectively stand for those vertices which can reach the strongly connected component but cannot be reached from it and those which can be reached from the strongly connected component but cannot reach it. We call the union of the links-in and of the strongly connected component the *in-component* and the union of the links-out and of the strongly connected component the *out-component*.

Figure 7.5 compares the evolution of the size of the biggest out-component in a random network of 10000 vertices with the value of our new Connectivity Indicator $ci' = \sum_{j,k}(jk - j(bc + cc) - k)p_{jk}$ as the number of directed edges varies. The directed edges are added successively by choosing ordered pairs of vertices. At each step, we make sure

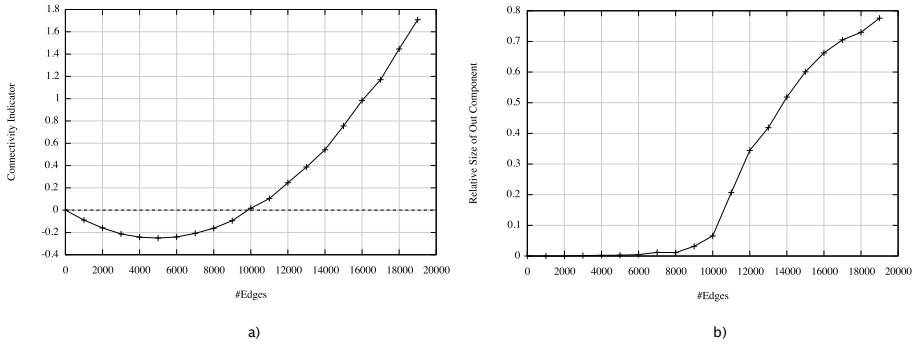

Figure 7.5: Connectivity Indicator (a) and maximal out-component (b) for a random network of 10000 vertices and a varying number of edges.

that the graph remains non-redundant and irreflexive. As expected, the Connectivity Indicator becomes positive at the phase transition when a giant-component emerges and grows then with the size of that component.

7.5 Semantic Component Size

Even in a network where parties are not all semantically interoperable, a given peer can be tempted to send a query and observe how it gets propagated through the different semantic domains. We can get a very good approximation of the degree of semantic diffusion of a query from our model.

Using a similar approach as described in recent graph theoretic works [NSW01] and taking advantage of our specific generating functions, we can calculate the relative size s of the subgraph that can be reached from the strongly connected component of the semantic connectivity graph (*out-component*):

$$s = 1 - G_0(u), \tag{7.9}$$

where u is the smallest non-negative real solution of

$$u = G_1(u). \tag{7.10}$$

Figure 7.6 shows the size of the out-component in a randomly generated digraph of 10000 vertices with a varying number of edges. The two curves represent the relative size of the component (i) as measured in the graph and (ii) as evaluated using the degree distribution, the clustering coefficient and the bidirectional coefficient of the graph with the method described above. As the figure shows, the theory and practice are in good agreement (less than one percent of difference in the super-critical phase).

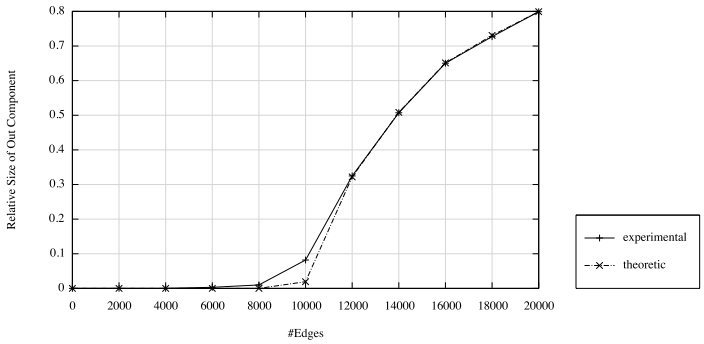

Figure 7.6: Size Comparison of the out-component in a random network of 10000 vertices.

7.6 Weighted Graphs

So far, we only analyzed the presence and size of a giant connected component in order to determine which portion of a semantic network could potentially be semantically interoperable. In large-scale decentralized networks, however, one should not only look into the giant semantic component itself, but also analyze the *quality* of the mappings used to propagate queries from one schema to the other (see Chapters 4 and 6 for a discussion on that topic). Indeed, in any large, decentralized network, it is very unlikely that all schema mappings could *correctly* map queries from one schema to the other, because of the lack of expressivity of the mapping languages, and of the fact that some (most?) of the mappings might be generated automatically.

Thus, as considered by more and more semantic query routing algorithms, we introduce weights for the schema mappings to capture the quality of a given mapping. Weights range from zero (indicating a really poor mapping unable to semantically keep any information while translating the query) to one (for perfect mappings, keeping the semantics of the query intact from one schema to the other). We then iteratively forward a query posed against a specific schema to other schemas through schema mappings if and only if a given mapping has a weight (i.e., quality) greater than a predefined threshold τ. $\tau = 0$ corresponds to sending the query through any schema mapping, irrespective of its quality. On the contrary, when we set τ to one, the query only gets propagated through semantically perfect mappings, while even slightly faulty mappings are ignored. Previous works in statistical physics and graph theory have looked into percolation for weighted graphs. We present hereafter an extension of our heuristics for weighted semantic networks inspired by recent works in graph theory [CNSW00].

7.6.1 Connectivity Indicator

As before, we consider a generating function for the degree distribution

$$G_0(x) = \sum_{k=0}^{\infty} p_k x^k$$

where p_k is the probability that a randomly chosen vertex has degree k in the network. We then define t_{jk} as the probability that an edge has a weight above τ given that it binds vertices of degree j and k. Thus, $w_k = \sum_{j=0}^{\infty} t_{jk}$ is the probability that an edge *transmits*, given that it is attached to a vertex of degree k. The generating function for the probability that a vertex we arrive at by following a randomly chosen edge is of degree k *and* transmits is

$$G_1(x) = \frac{\sum_{k=0}^{\infty} w_k k p_k x^{k-1}}{x^{cc} \sum_{k=0}^{\infty} k p_k}$$

where cc is the clustering coefficient. We know [CNSW00] that the generating function for the probability that one end of a randomly chosen *edge* on the graph leads to a percolation cluster of a given number of vertices is

$$H_1(x) = 1 - G_1(1) + x G_1 \left[H_1(x) \right]. \tag{7.11}$$

Similarly, the generating function for the probability that a randomly chosen *vertex* belongs to a percolation cluster of a given number of vertices is

$$H_0(x) = 1 - G_0(1) + x G_0 \left[H_1(x) \right] \tag{7.12}$$

such that the mean component size corresponding to a randomly chosen vertex is

$$\langle s \rangle = H_0'(1) = G_0(1) + \frac{G_0'(1) G_1(1)}{1 - G_1'(1)}$$

which diverges for $G_1'(1) \geq 1$. However,

$$G_1'(1) = \frac{\sum_{k=0}^{\infty} w_k p_k k (k - 1 - cc)}{\sum_{k=0}^{\infty} k p_k}$$

such that a giant connected component appears if

$$ci = \sum_{k=0}^{\infty} k p_k \left[w_k (k - 1 - cc) - 1 \right] \geq 0.$$

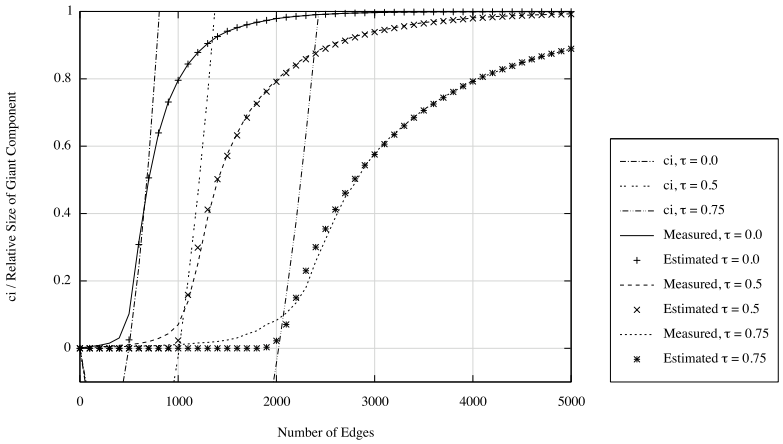

Figure 7.7: Giant component size and connectivity indicator for undirected weighted random graphs of 1000 vertices, with various thresholds τ limiting the propagation of the query.

7.6.2 Giant Component Size

As seen above, $H_0(x)$ represents the distribution for the cluster size which a randomly chosen vertex belongs to, *excluding* the giant component. Thus, $H_0(1)$ is equal to the fraction of the nodes which are not in the giant component. The fraction of the nodes which are in the giant component is hence $S = 1 - H_0(1)$. Using Equation 7.12 we can write

$$S = 1 - H_0(1) = G_0(1) - G_0\left[H_1(1)\right].$$

with $H_1(1) = 1 - G_1(1) + G_1\left[H_1(1)\right]$. Thus $H_1(1) = u$ where u is the smallest non-negative solution of

$$u = 1 - G_1(1) + G_1(u).$$

The relative size of the giant component reached by the query in a weighted semantic graph follows as

$$S = G_0(1) - G_0(u).$$

Equations for the directed case can be derived in a similar manner. Figure 7.7 compares the connectivity indicator and the relative size of the giant component as derived using our heuristics and as measured in the random graphs. We consider purely random graphs, and edge weights – representing the mapping qualities – as realizations of a continuous random variable uniformly distributed on $[0, 1]$. All the results are averaged over 20 consecutive runs and are given for different thresholds τ limiting the propagation of a query.

7.7 Semantic Interoperability in a Bioinformatic Database Network

Even if Semantic Web technologies and semi-structured representations have recently gained momentum, their deployment on the wide-scale, structured Internet is still in its infancy. Only a very small portion of websites have so far been enriched with machine-processable and queryable data encoded in RDF or XML. Thus, the difficulty to analyze semantic networks due to the very lack of realistic data one can gather about them. We gave a couple of results pertaining to our approach above by generating large-scale, random topologies. Below, we test our heuristics on a real semantic network, namely on a collection of schemas registered within the proprietary Sequence Retrieval System (SRS)[1].

We start below by giving a short introduction to SRS. We then present our approach, which boils down to an analysis of the component sizes in a graph of schemas interconnected through schema mappings. We report on the statistical properties of the SRS network we consider and on the performance of our heuristics applied to that network. Finally, we report on the performance of our approach on larger and weighted networks mimicking the statistical properties of the SRS network.

7.7.1 The Sequence Retrieval System (SRS)

The *Sequence Retrieval System* SRS is a commercial indexing and retrieval system designed for bioinformatic libraries such as the *EMBL* nucleotide sequence databank, the *SwissProt* protein sequence databank or the *Prosite* library of protein subsequence consensus patterns. It started as a data management system initially developed by the European Molecular Biology Laboratory in Heidelberg. As such, it allows the querying of one or several databases simultaneously, regardless of their format. SRS repositories typically contain a central index for one hundred or more databases, whose data are saved as unstructured (so-called *flat*) files mainly.

Administrators wishing to register new databases within an SRS repository first have to define *structure files*, which detail on a syntactic level the schema according to which data has been organized in the flat files. Once their schemas have been defined, administrators can export schema instances (i.e., flat text files) whose data will be centrally parsed, indexed and processed thanks to the corresponding schema definitions. Entries in bioinformatic databases often contain explicit or implicit references to each other; for example, information about elements related to a nucleic acid segment can be available in a protein databank. Taking advantage of this fact, SRS lets administrators manually define relationships between their database schema and similar schemas using a proprietary language.

[1] *http://www.lionbioscience.com/*

7.7.2 Graph analysis of an SRS repository

Conceptually, the model described above is very close to the semantic networks we have been discussing throughout this chapter, i.e., a collection of related schemas (or ontologies) linked one to another through pairwise mappings. The graph that can be extracted from an SRS repository has two main advantages over those which can be built from current RDFS / XML repositories: i) it is based on a real-world collection of schemas, which are being used on a daily basis by numerous independent parties and ii) it is of a reasonable size (several hundreds of semantically related complex schemas). Thus, after having been rather unsuccessful at finding reasonable semantic networks from the Web itself, we decided to build a specialized crawler to analyze the semantic graph of an SRS repository and to test our heuristics on the resulting network.

We chose to analyze the semantic network from the European Bioinformatics Institute SRS repository[2]. We built a custom crawler, which traverses the entire network of databases and extracts schema mapping links stored in the schema definition files. The discussion below is based on the state of the SRS repository as of May 2005.

The graph resulting from our crawling process is an undirected graph of 388 nodes (database schemas) and 518 edges (pairwise schema mappings). We chose to represent links as undirected edges since they are used in both directions in SRS (they basically represent cross-references between two entries of two database schemas). We identified all connected components in the graph (two nodes are in the same connected component if there is a path from one to the other following a series of edges). The analysis revealed one giant connected component, i.e., a relatively large set of interconnected schemas, of 187 nodes, which represents roughly half of the nodes and 498 edges. Besides the giant connected component, the graph also has two smaller components, each consisting of two vertices. The rest of the nodes are isolated, representing mostly result databases or databases for which no link to other databases was defined.

The average degree of the nodes is 2.2 for the whole graph and 4.6 for the giant component. Real networks differ from random graphs in that often their degree distribution follows a power law, or has a power law tail, while random graphs have a Poisson distribution of degrees [AB02]. Unsurprisingly, our semantic network is no exception as can be seen in Figure 7.8, which depicts an accurate approximation of the degree distribution of our network by a power-law distribution $y(x) = \alpha x^{-\gamma}$ with $\alpha = 0.21$ and $\gamma = 1.51$.

Another interesting property we explored was the tendency of the schemas to form clusters, quantified by our clustering coefficient. The network we considered has a high average clustering coefficient of 0.32 for the whole graph and of 0.54 for the giant component. The diameter

[2]publicly available at *http://srs.ebi.ac.uk/*

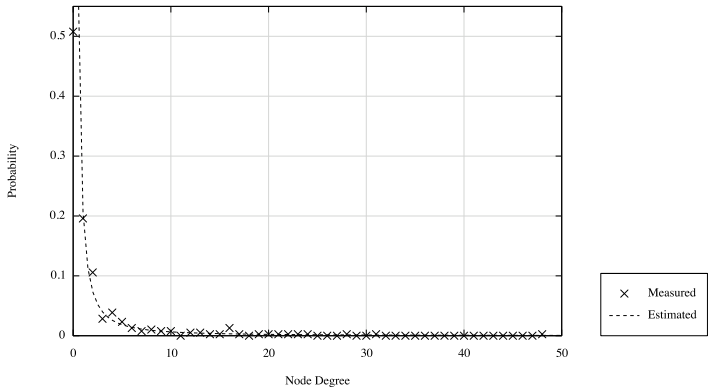

Figure 7.8: An approximation of the degree distribution of our semantic network by a power-law distribution $y(x) = \alpha x^{-\gamma}$ with $\alpha = 0.21$ and $\gamma = 1.51$.

(maximum shortest paths between any two vertices) of the giant connected component is 9. Those data indicate that our network can be characterized as *scale-free* (power-law distribution of degrees) or *small-world* (small diameter, high clustering coefficient).

7.7.3 Applying our Heuristics to the SRS Graph

We applied our heuristics to the SRS graph we obtained from the crawling process. The results are as follows: we get a connectivity indicator *ci* of 25.4, indicating that the semantic network is clearly in a super-critical phase and that a giant component interlinking most of the databases has appeared. The size of this giant component as estimated by our heuristics (see above) is of 0.47, meaning that 47% of the schemas are part of the giant connected component. This is surprisingly close to the real value of 0.48 as observed in the graph.

7.7.4 Generating a Graph with a given Power-Law Degree Distribution

Going slightly further, we want to analyze the dynamics of semantic graphs with varying numbers of edges. Our aim is to generate graphs with the same statistical properties as the SRS graphs, that is, graphs following a power-law degree distribution:

$$P(k) = \alpha k^{-\gamma} \tag{7.13}$$

but with a varying number of edges. We take an existing [CCRM02] graph-building algorithm yielding a power-law degree distribution with a given exponent γ. It goes as follows: (1) create a (large) number N of

vertices, (2) to each vertex i, assign an *importance* x_i, which is a real number taken from a distribution $\rho(x)$, and (3) for each pair of vertices, draw a link with probability $f(x_i, x_j)$, which is function of the importance of both vertices.

Now if $f(x_i, x_j) = (x_i x_j / x_M^2)$ (where x_M is the largest value of x in the graph), we know [CCRM02] that the degree distribution of the graph is

$$P(k) = \frac{x_M^2}{N\langle x\rangle}\rho\left(\frac{x_M^2}{N\langle x\rangle}k\right) \tag{7.14}$$

where $\langle x\rangle$ is the expected value of the importance x, such that $P(k)$ follows a power-law if $\rho(x)$ does so.

We then choose a power-law distribution

$$\rho(x) = \frac{\gamma - 1}{(m^{1-\gamma} - Q^{1-\gamma})}x^{-\gamma} \tag{7.15}$$

defined over the interval $[m, Q]$. However, we still have to find values for m and Q such that the scale of the resulting degree distribution equals to α. Using Equation 7.15, we find the expected importance value as

$$\langle x\rangle = \frac{(\gamma - 1)(m^2 Q^\gamma - m^\gamma Q^2)}{(\gamma - 2)(mQ^\gamma - m^\gamma Q)}.$$

Replacing $\rho(x)$ in Equation 7.14, the degree distribution of the resulting graph becomes

$$P(k) = \frac{x_M^2}{N\langle x\rangle}\frac{\gamma - 1}{(m^{1-\gamma} - Q^{1-\gamma})}\left(\frac{x_M^2}{N\langle x\rangle}k\right)^{-\gamma} \tag{7.16}$$

such that, equating with Equation 7.13, we get

$$\alpha = \frac{x_M^2}{N\langle x\rangle}\frac{\gamma - 1}{(m^{1-\gamma} - Q^{1-\gamma})}\left(\frac{x_M^2}{N\langle x\rangle}\right)^{-\gamma}. \tag{7.17}$$

We can then arbitrarily choose $m > 0$ and find Q by numerical approximation, since the right-hand side of Equation 7.17 is defined and continue for values of $Q > m$.

Figures 7.9 and 7.10 show the results of our heuristics on networks of respectively 388 (i.e., mimicking the original SRS graph) and 3880 edges (i.e., 10 times bigger than the original SRS graph but with the same statistical properties) constructed in the way presented above with a varying number of edges. The curves are averaged over 50 consecutive runs. As for the original SRS network, we see that we can accurately predict the size of the giant semantic component, even for very dense graphs.

Figures 7.11 and 7.12 show the results of our heuristics on weighted networks of respectively 388 and 3880 nodes, for a varying number of edges

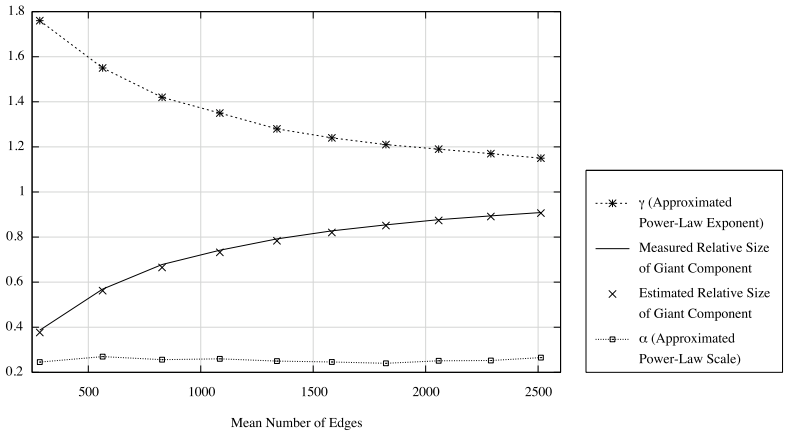

Figure 7.9: Estimation of the giant component size of a scale-free semantic network of 388 nodes with a varying number of edges.

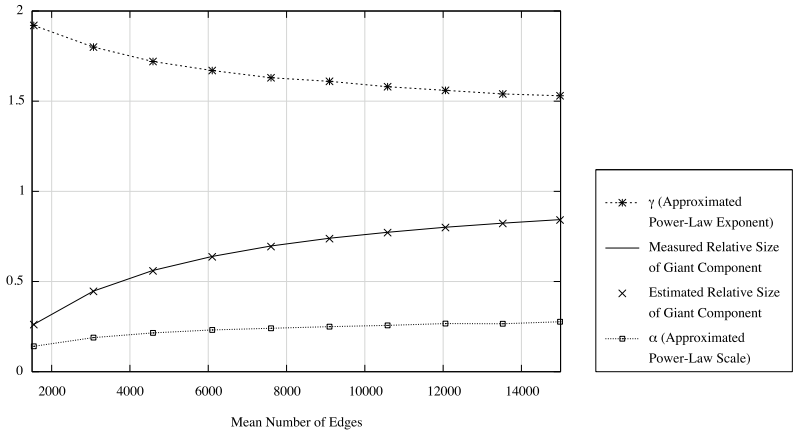

Figure 7.10: Estimation of the giant component size of a scale-free semantic network of 3880 nodes with a varying number of edges.

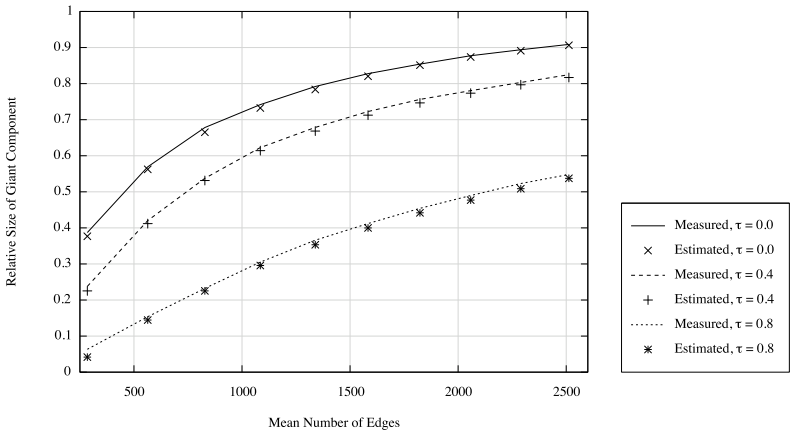

Figure 7.11: Fraction of the graph a local query gets forwarded to, for a weighted network of 388 nodes with a varying number of edges and various forwarding thresholds τ.

and various values of τ. The curves are averaged over 50 consecutive runs, and the weights of individual schema mappings are randomly generated using a uniform distribution ranging from zero to one. We see that our heuristics can quite adequately predict the relative size of the graph which a given query gets forwarded to. Also, as for the unweighted analysis, we observe similar behaviors for the two graphs; this is rather unsurprising as we are dealing with scale-free networks whose properties are basically independent of their size.

7.8 Use Case Scenarios

The methods described so far can readily be applied to study semantic interoperability of large-scale semantic systems in a global manner. Besides, we also believe in their high utility when used locally, e.g., by individual peers in the system. Peers can determine the statistical properties (degree distribution, clustering and bidirectional coefficients) of a semantic network in several ways:

- In a structured P2P system, they can lookup the different values in the common registry of the system (see for example the following chapter). This of course requires the different peers to insert their own local values in the repository beforehand.
- They can query a third-party tool (e.g., a semantic search engine) that regularly crawls the semantic graph to gather its statistical properties.

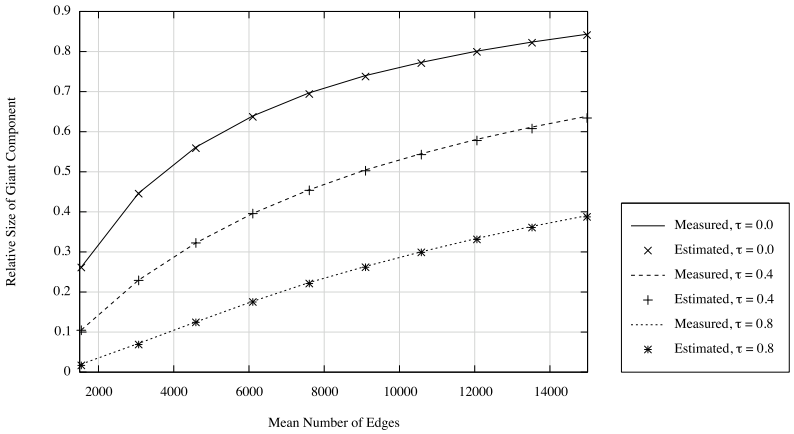

Figure 7.12: Fraction of the graph a local query gets forwarded to, for a weighted network of 3880 nodes with a varying number of edges and various forwarding thresholds τ.

- They can approximate the statistical properties themselves, by gathering information from queries routed randomly through the semantic network.

In an unstructured P2P network, the approximation of the statistical properties of the semantic connectivity graph can for example be made locally, using *ping* queries flooding the neighborhood with a certain TTL and analyzing *pong* answers containing the degree of each node responding. Figure 7.13 gives the approximated degree distribution of a graph using flooding with a TTL of 5 for a directed random graph of 1000 vertices. As the degree distribution is a two-dimensional variable in a directed, graph, we decided to focus on the out-degree distribution for ease of presentation. If $D(x, y) = \sum_{jk} p_{jk} x^j y^k$ is the generating function for the in-out degree distribution – with p_{jk} the probability that a vertex has in-degree j and out-degree k – then the out-degree distribution is simply

$$D_{out}(y) = \sum_j D(1, y) = \sum_{jk} p_{jk} y^k.$$

The experiment was conducted for 50 consecutive randomly generated graphs; for each graph, we randomly select initially a reference vertex we name *root* vertex and always estimate the degree distribution from that vertex as the graph grows. It might happen that the root vertex is not initially part of the giant component of the graph; in that case, the size of the giant connected component is naturally underestimated. Figure 7.13 compares the resulting evaluation of the degree distribution with the real degree distribution as measured in the graphs. Notice that the more

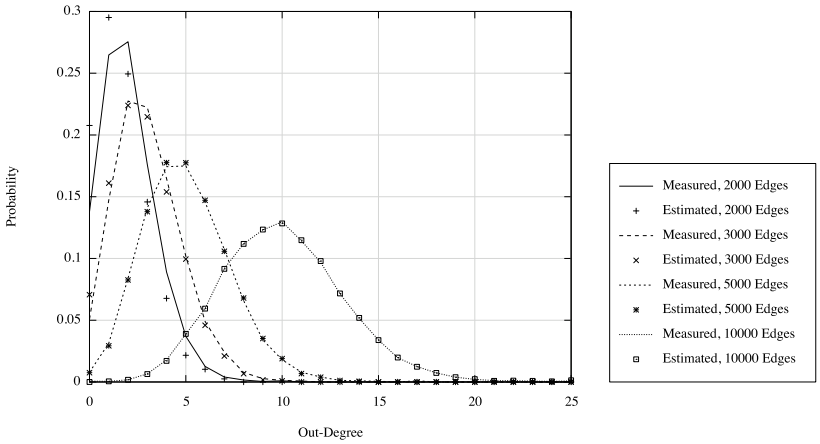

Figure 7.13: Estimation of the out-degree distribution with a flooding of TTL 5 for different number of edges in a directed graph of 1000 vertices.

edges, the better the estimation: with few edges, it is more likely that the root vertex, taken as the origin for the flooding, is part of a small isolated component or even not connected at all. Figure 7.14 gives results pertaining to the estimation of the out-component size using a degree distribution estimated with flooding and various semantic thresholds τ.

The precision of the estimation is related to the amount of degree information gathered throughout the graph. Figure 7.15 shows the (Pearson's product-moment) correlation coefficient between the measured and estimated degree distributions of the graphs, averaged over 50 directed graphs of 1000 vertices and 4000 edges. The estimations are based on flooding with varying TTLs and on semantic random walkers, which represent queries routed randomly in the semantic network from a root vertex with a given TTL and gathering the degree distributions along its path. Floodings with large TTLs lead to the best results, but also impose a higher network load ($\sum_{i=1}^{TTL} outdegree^i$ messages for the flooding versus $\#walkers * TTL$ messages for the random walkers). For the setting of Figure 7.15, flooding is more expensive than random walkers for TTLs greater of equal to three (84 messages for flooding with a mean out-degree of four versus 60 messages for the random walkers). Figure 7.16 shows the relative error on the estimation of the out-component of the graphs, using our methods and the estimated degree distributions with flooding and random walkers.

Once gathered, the relevant data can be exploited in order to foster semantic interoperability in the large: when joining a semantic network, peers can determine wether the semantic network in question is semantically interoperable. If it is not, they can trigger the (automated or manual) creation of new mapping links until the semantic connectivity

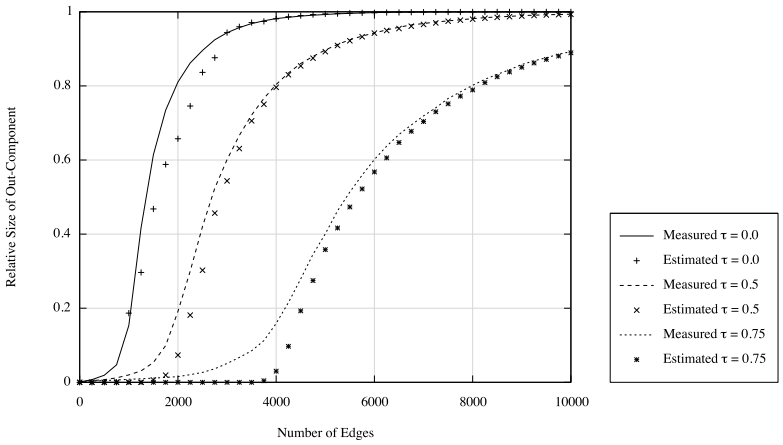

Figure 7.14: Out-component size for a weighted directed network of 1000 nodes and 4000 edges, with the degree distribution estimated by a flooding of TTL 5.

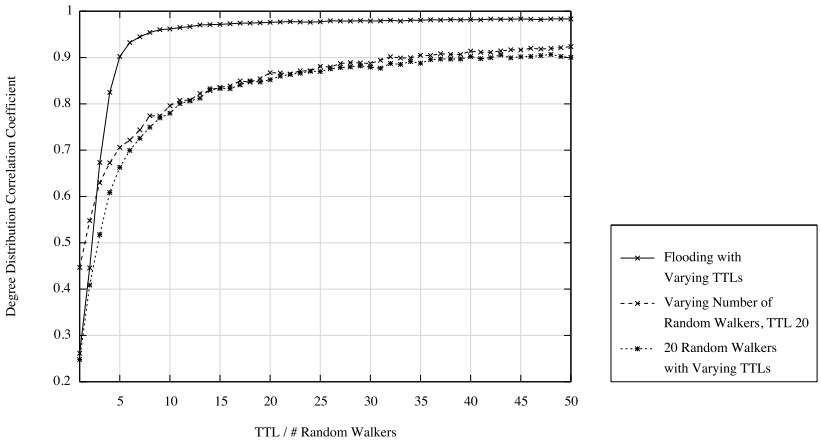

Figure 7.15: Correlation coefficient between the measured degree distribution and estimated degree distributions for different estimation techniques; the estimations were performed on directed random graphs of 1000 vertices and 4000 edges.

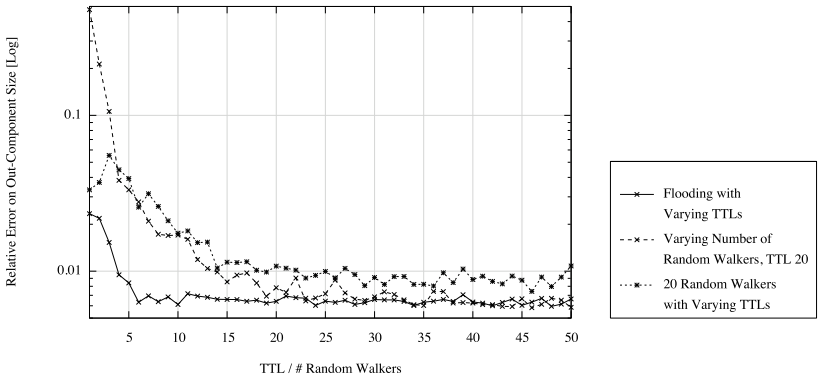

Figure 7.16: Relative error (logarithmic scale) on the estimation of the out-component size for different estimation techniques; the estimations were performed on directed random graphs of 1000 vertices and 4000 edges.

subgraph moves to a super-critical phase ($ci > 0$). Such heuristics may have to be used periodically in environments where schemas and mappings appear or disappear dynamically. Moreover, peers can evaluate the potential impact of a query based on a given schema: once a network is semantically interoperable, peers can predict the degree to which a query will be forwarded through the Schema-to-Schema graph, thanks to the component size analysis. Finally, note that our heuristics have been applied on schema mappings above, but could in a similar manner be applied at a finer granularity on attribute mappings to determine to which extent a given attributes A_i is known – in some form or another – throughout the network.

7.9 Conclusions

So far, there exists little research on semantic interoperability in the large. Current approaches typically analyze a handful of schemas or ontologies at a time only, and from a purely local perspective. Research on large-scale systems (e.g., works on Web dynamics or social networks) cannot be directly applied to our problem because of its specificities. We believe that new frameworks have to be developed in order to rightfully model the upcoming large-scale semantic systems. This chapter pointed to one possible, and in our opinion promising, avenue by taking advantage of a recent graph-theoretic framework to analyze and (potentially) iteratively realize semantic interoperability in a large network of information-sharing parties. We evaluated our approach on a real semantic network, with results confirming the validity of our heuristics beyond our initial hopes.

Also, we extended our analysis to apply our heuristics on larger networks enjoying similar statistical properties and on weighted semantic networks. It was for us quite important to test our heuristics using real-world data, as semantic network analyses mostly consider artificial networks today, due to the current lack of semantically enriched websites or deployed semantic infrastructures.

Our techniques can readily be used in real networks to determine whether or not the system is interoperable from a decentralized perspective. Also, they can be used to locally predict the degree of diffusion of a query in the network given a semantic threshold. This first work opens a whole range of extensions and improvements. In the future, we plan to extend our analyses to take into account the dynamicity (churn) of the network of schema mappings, and to consider more accurate query forwarding schemes based on transitive closures of mapping operations. Also, we plan to integrate some of the heuristics presented above in our own semantic P2P system, GridVine, presented in the following chapter. Finally, note that Jiang, Cybenko and Hendler [JCH06] recently developed an interesting and related framework to analyze semantic interoperability in the large – called *information fluidity* in their context. Their framework, however, is much more limited than ours as it only deals with undirected, unweighed networks analyzed from a global perspective.

Part III

Systems

Chapter 8

GridVine: Building Internet-Scale Semantic Overlay Networks

We present hereafter two systems designed following some of the principles we have proposed in the preceding chapters. The first system, GridVine, is a DHT-based Semantic Overlay Network fostering semantic interoperability through Semantic Gossiping. The second system, called PicShark and described in Chapter 9, builds on GridVine to offer picture sharing functionalities in environments where picture annotations are scarce and heterogeneous.

8.1 Introduction

In the preceding chapters, we focused on analyzing schemas and schema mappings to foster semantic interoperability in large scale structured settings. In this chapter, we come back to the vision of a three-layered Semantic Overlay Network we originally depicted in Figure 3.4. We present a system called GridVine [ACMHvP04] fostering semantic interoperability at the semantic mediation layer while organizing peers in structured manner thanks to a DHT at the overlay layer.

Structured overlay networks like Chord [SMK+01] or P-Grid [Abe01] have been developed as new infrastructures to route requests for resources that are distributed over large populations of peers using application-specific keys in an efficient manner. Those networks can be employed in order to efficiently respond to simple keyword-based queries. Structured overlay networks clearly also have the potential to support the efficient operation of a semantic overlay network. In the following, we introduce an architecture and implementation leveraging on the potential for scalability offered by structured overlay networks in the realization of interoperable, large-scale semantic overlay networks. We propose a semantic P2P system

where end-users can freely annotate the data items they want to share using semi-structured schemas, and search for data items using expressive queries. A key aspect of the approach we take is to apply the principle of data independence [Hel03] by separating a logical from a physical layer. This principle is well-known from the database area and has largely contributed to the success of modern database systems. At the logical layer, we support various operations to maintain the semantic overlay network and to support semantic interoperability, including attribute-based search, schema management, schema inheritance and schema mapping. We also provide support for specific schema reconciliation techniques, including Semantic Gossiping, which we have introduced earlier in Chatper 4. We provide those mechanisms within the standard syntactic framework of RDF/OWL. At the physical layer, we provide efficient realizations of the operations exploiting a structured overlay network, namely P-Grid. This requires mappings of semantic data and operations to the physical layer. Important aspects of this mapping are:

- The mapping of data and metadata to routable keys.
- The introduction of a specific namespace for resources present in the peer space, such that the infrastructure can resolve resource requests.
- The implementation of traversals of the semantic mediation layer taking advantage of intermediate schema mappings. An interesting aspect is the possibility to use different strategies to implement such traversals at the structured overlay network layer, in ways that are substantially different from naive solutions. We analyze two types of processing strategies, iterative and recursive. As in standard database query processing, the data independence principle thus opens up the possibility of optimization using different query processing strategies.

The rest of this chapter is structured as follows: we start with an overview of our approach in Section 8.2. Our architecture and implementation use P-Grid as a physical layer, which is briefly described in Section 8.3. Section 8.4 presents the mechanisms used to index metadata or schemas and to resolve queries. Section 8.6 describes semantic interoperability while Section 8.7 is dedicated the implementation of our approach. Finally, we discuss related work in Section 8.8 and conclude.

8.2 Overview of our Approach

8.2.1 Data Independence

Following the principle of data independence introduced above, our approach revolves around a two-layer model: a physical layer based on the

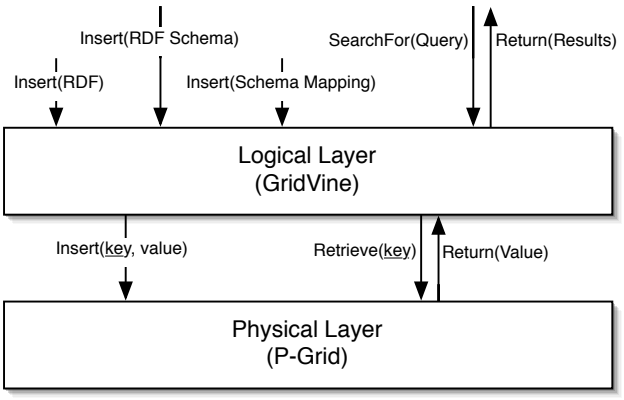

Figure 8.1: The principle of data independence: the GridVine Semantic Overlay Network is built on top of the P-Grid P2P access structure.

P-Grid access structure underpinning GridVine, and a logical semantic overlay layer (see Figure 8.1). P-Grid (Section 8.3) is an efficient, self-organizing and fully decentralized access structure based on a Distributed Hash Table (DHT). GridVine uses two of P-Grid's basic functionalities: the $Insert(\underline{key}, value)$ primitive for storing data items based on a key identifier and the $Retrieve(\underline{key})$ primitive for retrieving data items given their key.

Taking advantage of those two rather limited primitives, we build a full-fledged semantic overlay network on top of P-Grid. The system exposes a new set of primitives (depicted on top of Figure 8.1) allowing end-users to insert metadata, schemas and schema mappings as well as retrieve semantic information using expressive query languages. Capitalizing on recent developments, we chose the RDF / RDFS pair as languages to encode metadata and vocabulary definitions in GridVine. Those two languages represent the fundamental building blocks of the emerging Semantic Web (see Figure 2.4) and are predestined to become *de facto* standards for encoding metadata as well as their corresponding schematic information.

The exact mechanisms we choose for inserting metadata into the P-Grid are naturally of utmost importance, since they directly influence the query resolution capabilities of the overall system, and are extensively discussed in the following (Section 8.4). In order to support the processing of schema-specific information, we introduce a meta-schema specifying common characteristics for all custom schemas derived by the users. Also, we introduce new addressing spaces, i.e., URI schemes, to identify resources both in the physical (P-Grid data items) and logical (semantic information) layers.

8.2.2 Decentralized Semantics

Classification of resources and definition of vocabularies are essential for leveraging metadata creation and fostering semantic interoperability through reuse of conceptualizations. Legacy information sharing systems typically support static sets of centrally imposed, predefined schemas. We consider such a monolithic approach as far too rigid for adequately capturing information sources in a network of autonomous and heterogeneous parties. Not only is this not desirable from an ideological perspective, it also misses out on the power of P2P. Seeing users as experts in the information they share, they themselves are most competent in coming up with a proper schema to describe their data. However desirable it may be to let users come up with their own schemas in a bottom-up manner, it also severely endangers global semantic interoperability and search capabilities: how could one ensure optimal precision and recall when searching for data items that might be referred to by a large variety of terms? In the context of GridVine, our answer to this question if twofold, including both schema inheritance and Semantic Gossiping mechanisms.

Schema inheritance provides GridVine with basic schema reusability and interoperability capabilities. As for other social networks [AB02], we expect the popularity of schemas in GridVine to follow scale-free preferential attachment laws, such that a small subset of schemas gain unparalleled popularity while the others remain mainly confidential. By allowing users to derive new schemas from well-known base schemas, we implicitly foster interoperability by reusing sets of conceptualizations belonging to the base schemas.

Semantic Gossiping (see Chapter 4) is a semantic reconciliation method that can be applied to foster semantic interoperability in decentralized settings. The method aims at establishing global forms of agreement starting from a graph of purely local mappings among schemas. Following this approach, we allow peers in GridVine to create, and possibly index, schema mappings linking schemas one to another. Those mappings can be used to propagate queries in such a way that relevant data items annotated according to different schemas can also be retrieved. Query forwarding can be implemented using several approaches. In the following, we identify two radically different strategies for forwarding queries: iterative forwarding, where peers process series of mapping links repeatedly, and recursive forwarding, where peers delegate the forwarding to other peers. Schema inheritance and Semantic Gossiping are further described in Section 8.6.

8.3 The P-Grid P2P system

GridVine takes advantage of the P-Grid P2P access structure at the physical layer. P-Grid is based on a DHT [PRR97]. As any DHT approach, P-Grid associates peers with data keys from a key space, i.e., partitions of the underlying distributed data structure. Each peer is responsible for some part of the overall key space and maintains additional (routing) information to forward queries and requests. Without constraining its general applicability, we use binary keys in the following. P-Grid peers refer to a common underlying tree structure in order to organize their routing tables. In the following, we assume that the tree is binary. This is not a fundamental limitation as a generalization of P-Grid to k-ary structures has been introduced [AP03], but will simplify the following presentation.

Each peer $p \in \mathcal{P}$ is associated with a leaf of the binary tree. Each leaf corresponds to a binary string $\pi \in \Pi$. Thus, each peer p is associated with a path $\pi(p)$. For search, the peer stores for each prefix $\pi(p,l)$ of $\pi(p)$ of length l a set of references $\rho(p,l)$ to peers q with property $\overline{\pi(p,l)} = \pi(q,l)$, where $\overline{\pi}$ is the binary string π with the last bit inverted. This means that for each level of the tree, the peer has references to some other peers that do not pertain to the peer's subtree at that level, which enables the implementation of prefix routing for efficient search. The cost for storing the references (in routing tables) and the associated maintenance cost are scalable as they are proportional to the depth of the underlying binary tree.

Each peer stores a set of data items $\delta(p)$. For $d \in \delta(p)$ the binary key $key(d)$ is generated using an order-preserving hash function $Hash()$. $key(d)$ has $\pi(p)$ as prefix but we do not exclude that temporarily also other data items are stored at a peer, that is, the set $\delta(p, \pi(p))$ of data items whose key matches $\pi(p)$ can be a proper subset of $\delta(p)$. In addition, peers also maintain references $\sigma(p)$ to peers having the same path, i.e., their replicas.

P-Grid supports two basic operations: $Retrieve(\underline{key})$ for searching a certain key and retrieving the associated data item and $Insert(\underline{key}, value)$ for storing new data items. Since P-Grid uses a binary tree, $Retrieve(\underline{key})$ intuitively is efficient, i.e., $\mathcal{O}(log(|\Pi|))$, measured in terms of messages required for resolving a search request, in a balanced tree. For skewed data distributions, it has been shown [Abe02] that due to the probabilistic nature of the P-Grid approach, the expected search cost measured by the number of messages required to perform the search remains logarithmic, independently of how the P-Grid is structured. This is important as it allows us to apply simple order-preserving hashing functions to metadata annotations, which may lead to non-uniformly distributed key distributions in practice. As P-Grid uses an order-preserving hash function to compute keys and define their association with peers, it pro-

cesses prefix and range queries of arbitrary granularity efficiently, i.e., in $\mathcal{O}(log(|\Pi|) + |\{p \mid \kappa \subseteq \pi(p)\}|$ messages, where κ denotes the common prefix of the borders of the range query [DHS+05]. Prefix queries will be an important constituent in the generic implementation of metadata queries. $Insert(\underline{key}, value)$ is based on P-Grid's more general update functionality [DHA03] which provides probabilistic guarantees for consistency and is efficient even in highly unreliable, replicated environments.

8.4 Semantic Support

In the following, we elaborate on how GridVine handles the creation and management of RDF triples (Section 8.4.1) and schemas (Section 8.4.2). We then discuss query resolution mechanismss in Section 8.5.

8.4.1 Metadata Storage

In GridVine, end-users and generated annotations are stored as RDF triples and refer to the data items shared in the P-Grid infrastructure. RDF stores meta-data statements as *triples* $t \in \mathcal{T}$; they always take the form of

$$t_i = \{t_{subject}, t_{predicate}, t_{object}\}$$

where $t_{subject}$(the *subject*) is the resource about which the statement is made, $t_{predicate}$ (the *predicate*) represents a specific property in the statement and t_{object} (the *object*) is the value (resource or literal) of the predicate in the statement.

In the present case, we make statements about files shared in the underlying P-Grid infrastructure. A structured overlay network allows to implement an application specific addressing space. In our case, we introduce the specific URI schemes *pgrid* : for resources, and *pgrids* :, for schema-elements. This does not exclude the use of other URI schemes in conjunction with P-Grid's specific ones.

In the case were all resources are identified by P-Grid URIs, a typical situation would be a statement where the subject is identified by a P-Grid key, i.e., a binary string such as *11110101*, whereas the predicate and object refer to P-Grid's specific RDF schemas (or literals). This allows us to constrain the applicability of the schema constructs. An example of such a statement could be *the P-Grid resource 11110101* (*subject*) *is entitled* (*predicate*) *Rain, Steam and Speed* (*object*), which, encoded using the XML serialization of RDF, would result in a file like the one transcribed in Figure 8.2.

We now exploit the underlying P-Grid infrastructure in order to enable sharing of data items in a large-scale distributed environment. The granularity of triple storage as well as the exact mechanism we use for inserting meta-data into the P-Grid are naturally of utmost importance since they both directly influence the query processing capabilities of the

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
  <rdf:Description rdf:about="pgrid://11110101">
    <Title xmlns="pgrids://01001101:Image#">Rain, Steam and Speed</Title>
  </rdf:Description>
</rdf:RDF>
```

Figure 8.2: A statement encoded using the XML serialization of RDF.

overall system. Since we want to support search capabilities of individual statements, we index each triple separately. Since most RDF query languages [PG04b] are based on constraint searches on the triples' *subject*, *predicate* or *object*, we have to reference each individual triple three times, generating separate keys based on their *subject*, *predicate* and *object* values. Thus, the insertion operation of a triple t is performed as follows:

$$Insert(t) \equiv Insert(t_{subject}, t), Insert(Hash(t_{predicate}), t), Insert(Hash(t_{object}), t).$$

Each peer p maintains a local database DB_p to store the triples it is responsible for. Since all RDF and RDFS statements can be written as ternary relations, the physical schemas of the local databases can all be identical and consist of three attributes $S_{DB} = (subject,\ predicate,\ object)$. The local databases support three standard relational algebra operators: projection π, selection σ and (self) join $\bowtie$.

In that way, each triple triggers three *Insert()* operations in GridVine. Prefix searches, e.g., on the beginning of a string representing an object value, are inherently supported by P-Grid's routing mechanisms. Supporting substring searches – for example on the literal *objects* of the triples – imposes to index all the suffixes of a generated key as well. Thus, if we introduce l as the average length of the strings representing *subjects*, *objects* or *predicates*, $3l$ *Insert()* operations are required to support substring searches on the triples.

8.4.2 Schema Definition And Storage

Classification of resources and definition of vocabularies are essential for leveraging meta-data creation and fostering semantic interoperability through reuse of conceptualizations. Modern information sharing systems usually support static sets of predefined schemas (e.g., taxonomies based on ID3[1] or EXIF[2]). We consider such monolithic approaches as far too rigid for adequately capturing contemporary information sources.

[1] http://www.id3.org/
[2] http://www.exif.org/

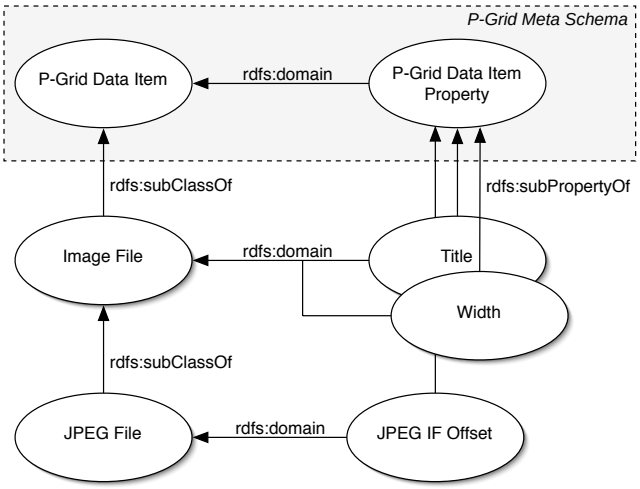

Figure 8.3: A user-defined schema, a second user-defined schema deriving from the first user-defined schema and their respective relation to the P-Grid meta schema.

Instead, we detail below an approach supporting on-the-fly schema creation, redefinition and extension by the end-user.

Schema information in GridVine is encoded using the most elementary schematic layer from the Semantic Web, i.e., RDF Schema (RDFS). Note that this does not preclude the use of more advanced ontological layers like OWL. RDFS is an extension of RDF providing mechanisms for describing groups of resources and their relationships. Among other capabilities, it allows to define classes of resources (*classes*) and predicates (*properties*) and to specify constraints on the subject (*domain*) or the object (*range*) of a given class of predicates.

GridVine schemas allow to declare new *categories* to describe application-specific resources. A meta-schema defines the base class and property from which all user-defined schemas are then derived. User-defined categories are all defined by a class subclassing a generic RDF super-class called *P-Grid Data Item* representing any P-Grid addressable resource. Properties referring to that class as *domain* allow to declare application-specific vocabularies (i.e., metadata attributes) with arbitrary values as *ranges*. In GridVine, all properties derive from the *P-Grid Data Item Property* super-property. A category class and its related properties are linked by the *domain* definition of the property. Multi-level inheritance is supported and follows the standard semantics of RDFS: instances of a subclass can act as *subject* whenever the *domain* of the property references the parent class.

We create distinct RDFS files for every category, regrouping the definitions of a subclass as well as all its affiliated properties. We create a

unique identifier for the category by concatenating the path $\pi(p)$ of the peer creating a category to the category itself. We then insert it into P-Grid as any other file:

$$Insert(RDF\_Schema) \equiv Insert(\pi(p) : Hash(Class\_Name), RDF\_Schema).$$

We give an example of a category definition in Figure 8.4. Note that since we can support substring searches, schemas can also be searched using the name of their representative class.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
 <rdfs:Class rdf:ID="Image" rdfs:comment="Image category">
   <rdfs:subClassOf rdf:resource=
    "http://www.p-grid.org/p-grid.rdfs#PGridDataItem"/>
 </rdfs:Class>

 <rdf:Property rdf:ID="Title">
   <rdfs:domain rdf:resource="#Image"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#string"/>
   <rdfs:subPropertyOf rdf:resource=
    "http://www.p-grid.org/p-grid.rdfs#PGridDataItemProperty">
 </rdf:Property>

 <rdf:Property rdf:ID="Width">
   <rdfs:domain rdf:resource="#Image"/>
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/>
   <rdfs:subPropertyOf rdf:resource=
    "http://www.p-grid.org/p-grid.rdfs#PGridDataItemProperty">
 </rdf:Property>
</rdf:RDF>
```

Figure 8.4: A user-defined category defined as an RDF Schema class with a series of RDF Schema properties representing the category attributes.

8.5 Resolving Queries in GridVine

We detail below how structured queries can be resolved taking advantage of both the metadata and the schematic information that have been indexed. We start by presenting how simple triple pattern queries are handled before providing a mapping to resolve more complex conjunctive queries.

8.5.1 Resolving Atomic Queries

A *triple pattern* [Sea04] is an expression of the form (s, p, o) where s and p are URIs or variables, and o is a URI, a literal or a variable. In GridVine, a native query is a conjunctive triple pattern query taking the following form:

$$x_1?, \ldots, x_m? \ : \ (s_1,\ p_1,\ o_1) \wedge \ldots \wedge (s_n,\ p_n,\ o_n)$$

where $x_1?, \ldots, x_m?$ are distinguished variables appearing in the triple patterns $(s_i,\ p_i,\ o_i)$. Note that joins can be expressed by multiple occurrences of the same variable in that notation. A triple pattern query retrieves (parts of) RDF triples based on the value of their *subject*, *predicate* or *object*.

Let us define $\boldsymbol{T}_{GV} \subseteq \mathcal{T}$ the set of all triples indexed in the system, and $\mathcal{D}$ the set of all URIs and RDF literals. A valuation v over a set of variables $\boldsymbol{v}$ is a total function from $\boldsymbol{v}$ to $\mathcal{D}$. We extend the valuations to map triple patterns $(s,\ p,\ o)$ to triples $t \in \boldsymbol{T}_{GV}$, and conjunction of triple patterns to conjunction of triples. The answer to a triple pattern consists of all m-tuples $v(x_1?), \ldots, v(x_m?)$ where v is a valuation over the conjunction of triple patterns in the query. For instance, the following triple pattern query

$$x_2? \ : \ (x_1?,\ pgrids://01001101 : Image\#Title,\ x_2?)$$

retrieves the $Title$ of all the instances of the $pgrids://01001101 : Image$ class. We call such a query an *atomic* query since it only contains one triple pattern.

In GridVine, atomic queries can be resolved by look-ups using a slightly modified version of the $Retrieve(\underline{key})$ primitive of P-Grid. A peer issuing an atomic query q first has to determine the address space $\underline{key}$ where it can find the answers. Once discovered, it simply forwards the query to the peer(s) responsible for that space using $Retrieve(\underline{key}, q)$. The resolution is dependent on the number of unbound variables in the query:

Three unbound variables atomic queries retrieve all triples $t \in \boldsymbol{T}_{GV}$, implying $\mathcal{O}(|\Pi| log(|\Pi|))$ messages and are not allowed in GridVine.

Two unbound variables atomic queries contain only one constant term c_0. We define $pos(term)$ as the position of a term (variable or constant) in a triple pattern, i.e., $pos(term)$ either takes *subject*, *predicate* or *object* as value. As all triples are indexed on their *subject*, *predicate* and *object* in GridVine, these queries can be resolved by sending the query to the corresponding address space: $Retrieve(c_0, q)$ if $pos(c_0) = subject$, $Retrieve(Hash(c_0), q)$ otherwise. The query resolution boils down to a standard P-Grid look-up generating $\mathcal{O}(log(|\Pi|))$ messages. Once arrived at its final destination(s) p_{dest}, the query is resolved with a local relational query on the local database DB_{dest}:

$$Results = \pi_{pos(distinguished\_variables)}\ \sigma_{pos(c_0)=c_0}\ (DB_{dest}).$$

One unbound variable atomic queries can be resolved in a similar manner. Since they contain two constant terms c_1 and c_2, the peer issuing the query has the choice of resolving the query by retrieving values based on either c_1 or c_2 (or even both). The most selective predicate should always be preferred. The corresponding local query used to retrieve the results is:

$$Results = \pi_{pos(distinguished\_variables)}\ \sigma_{pos(c_1)=c_1 \wedge pos(c_2)=c_2}\ (DB_{dest}).$$

Zero unbound variable triple patterns are constant terms and require no further resolution.

Once retrieved by the destination peer, the results are sent back to the original issuer of the query.

8.5.2 Resolving Conjunctive Queries

Conjunctive queries are resolved in a similar manner, by iteratively resolving each triple pattern contained in the query. Algorithm 8.1 gives the complete algorithm for conjunctive query resolution in GridVine. The peer issuing the query starts by resolving the first triple pattern. As the successive triple patterns are resolved, a set of intermediary results *resultSet*, containing the valuations of the variables in the query, is maintained. New results are combined with the set of intermediary results using a join operation on the existing valuations of the variables. Note that the algorithm proposed here can be optimized in several ways: by joining the result sets locally at the destination peers, by parallelizing the algorithm using multiple peers to answer parts of the query simultaneously [LIK06], or by selecting the most effective order of processing for the triple patterns.

8.6 Semantic Interoperability

As previously mentioned, supporting the extensions of existing schemas and the introduction of new schemas by end-users is essential to maximize the utility of information sharing systems. Introducing new vocabulary terms poses however a severe threat to the interoperability of the overall system. We detail below the mechanisms we take advantage of in order to foster semantic interoperability in GridVine.

8.6.1 Schema Inheritance

We let users derive new categories from the existing ones. However, we impose that the new class representing the subcategory subclasses the base category class. RDFS enforces monotonic inheritance of properties

Algorithm 8.1 Conjunctive Query Resolution

```
/*create a empty set of answers initially*/
resultSet = null;
/*resolve each triple pattern iteratively*/
for all triple-pattern t in query q do
  constants = getConstantTerms(t);
  key = getMostSelectiveConstant(constants);
  valuations = Retrieve(key, t);
  if answerSet == null then
    answerSet = valuations;
  else
    /*only keep the valuations compatible with the other valuations retrieved so far*/
    answerSet = answerSet ⋈ valuations;
  end if
end for
answerSet = π_{pos(distinguished_variables)} (answerSet);
```

through category hierarchies; in our case, the subcategory automatically inherits the properties defined in the base category through the *domain* definitions. Additionally, subcategories may introduce sets of new properties specific to the subclass. The process can of course be applied recursively in the sense that a subcategory may in turn serve as a supercategory for a new derivation, creating complex hierarchies of categories and classes from the most popular base schemas.

Figure 8.3 provided an example where a category for annotating JPEG files is derived from a more generic category of image files. All searches on a property of the base class $Image$ (such as $(x_1?,\ ImageTitle,\ x_2?)$) propagate naturally to all subclasses (e.g., $JPEG$) through the monotonic inheritance of the base properties. Thus, we create sets of semantically interoperable schemas through properties inherited by all descendants of a (potentially very popular) base schema.

Additionally, when inserting an instance of a derived class, we materialize the subsumption hierarchies in the system by inserting type assertions on the superclass: inserting $(x,\ rdf:type,\ z)$, triggers the insertion of $(x,\ rdf:type,\ z')\ \forall z' \mid z' \sqsupseteq z$. Thus, all the searches on instances of a base class (e.g., $(x_1?,\ rdf:type,\ Image)$) retrieves all instances of the subclasses as well (e.g., instances of the $JPEG$ class).

8.6.2 Semantic Gossiping

In Chapter 4, we introduced Semantic Gossiping to foster semantic interoperability in decentralized settings. Semantic gossiping aims at establishing global forms of agreement starting from a graph of purely local

```
<?xml version="1.0"?> <?xml version="1.0" encoding="ISO-8859-1" ?>
<rdf:RDF xmlns:owl ="http://www.w3.org/2002/07/owl#"
  xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

  <Image_Description xmlns="pgrids://10000101:exif#">
    <owl:equivalentProperty rdf:ID="mapping1"
     rdf:resource="pgrids://01001101:bmp#Title"/>
  </Image_Description>
  <#mapping1>
    <pgrids://owl#CycleAnalysis "1.0"/>
  </#mapping1>

  <Exif_Image_Width xmlns="pgrids://10000101:exif#">
    <owl:equivalentProperty rdf:ID="mapping2"
     rdf:resource="pgrids://01001101:bmp#Width"/>
  </Exif_Image_Width>
  <#mapping2>
    <pgrids://owl#ResultAnalysis "0.9"/>
  </#mapping2>

</rdf:RDF>
```

Figure 8.5: An example of schema mapping in GridVine, where the two attribute mappings are reified in order to introduce semantic similarity values.

schema mappings. Peers that have annotated their data according to the same schema are said to belong to the same *semantic neighborhood*. Each peer has the possibility to create (either manually or automatically) a mapping between two schemas, in effect creating a link between two semantic neighborhoods. The network as such can be seen as a directed graph of translations.

Following this approach, we allow peers in GridVine to create schema mappings mapping one schema onto another. A mapping $\mu_{S_1 \to S_2}$ can be used to propagate a query from a source semantic domain S_1 to a target semantic domain S_2 (see Section 4.6). Since RDFS does not support schema mapping, we encode schema mappings using OWL [Mv04]. Our mappings consist of series of *owl:equivalentProperty* statements, which characterize the correspondences between the two categories at the property level. Figure 8.5 gives an example of schema mapping in GridVine. Individual property equivalence statements are reified (i.e., equivalence statements are treated as resources) in order to account for partially-overlapping properties and subsumption mappings: peers can thus refer to the various equivalence statements and qualify the attribute mappings with semantic similarity values as introduced in Chapter 4. Both cycle and result analyses are implemented in GridVine. Result analysis is handled directly by the end-user, who can tag each result received to express whether or not the result was relevant.
Schema mappings are inserted at the address space corresponding of the

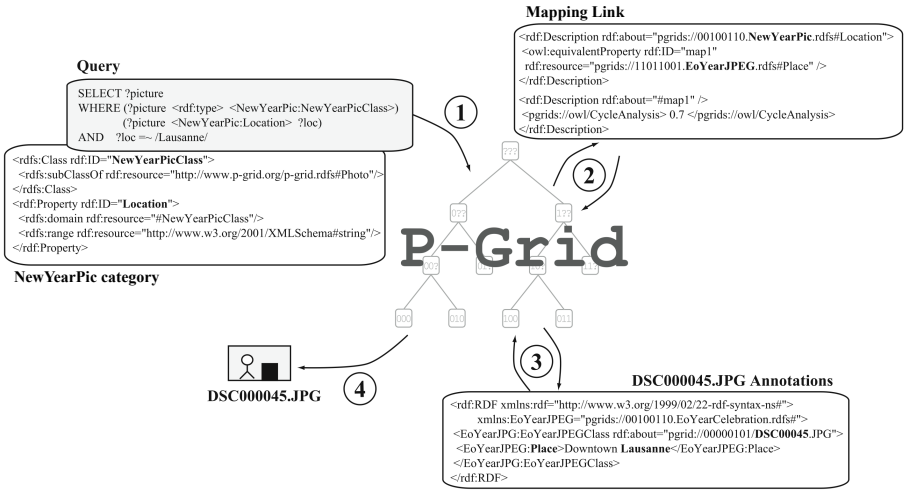

Figure 8.6: A simple example of Semantic Gossiping in GridVine.

source schema S_1 (or both schemas if the mapping is bidirectional):

$$Insert(Schema\_Mapping) \equiv Insert(\pi(p) : Hash(Source\_Class\_Name), Schema\_Mapping).$$

Thus, we can reformulate queries iteratively by looking-up all schemas mapped to the current schema of the query. Forwarding a query is then logically handled using gossiping as described in Section 4.6: starting from a given semantic domain, the query gets reformulated and iteratively traverses other semantic domains following mapping links until it is considered as being too different (either from a syntactic or a semantic point of view) from the original query. A new query is issued after each reformulation. Figure 8.6 shows a simplified example of Semantic Gossiping in GridVine. In the following section, we show that different physical implementations of Semantic Gossiping can be realized using the P-Grid overlay network.

8.7 Implementation

8.7.1 Architectural Overview

GridVine was implemented by extending our existing Java-based P-Grid library[3]. Figure 8.7 shows the architecture of the implementation as a UML class diagram.

The left hand-side shows the P-Grid library with the Gridella GUI, while the right hand-side shows GridVine's semantic extensions. The ar-

[3]available at *http://www.p-grid.org/*

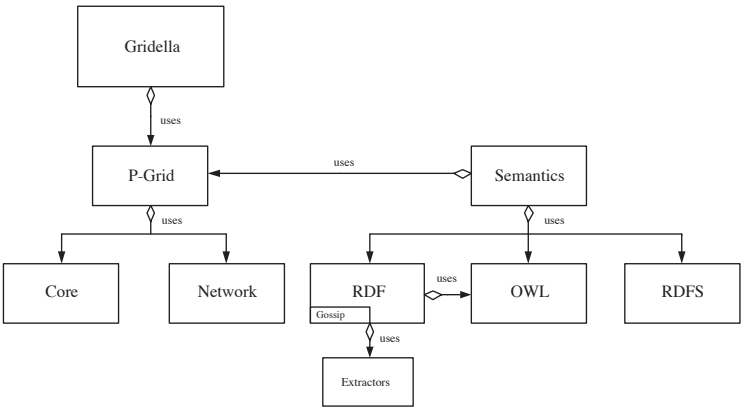

Figure 8.7: The GridVine component model

rows in the figure denote *uses* relationships. The *Gridella* component provides a GUI and uses the *P-Grid* component to issue queries; the *Semantics* component uses the *P-Grid* component to issue and receive queries and uses the *RDF*, *RDFS*, and *OWL* components to handle incoming requests. The *RDF* component is responsible for creating and managing RDF metadata and provides the gossiping functionalities. The *Extractors* subcomponent facilitates automatic metadata extraction to leverage the burden of manual annotation (e.g., automatic extraction of EXIF information from images). Functionalities related to schemas are provided by the *RDFS* component, while the *OWL* component handles all issues regarding schema mappings. The *P-Grid*, *Semantics*, *RDF*, *RDFS*, and *OWL* components are implemented as *Singletons*, i.e., only a single instance of each of these classes exists at runtime and handles all the requests.

8.7.2 Querying

Figure 8.8 shows the initiator's side of a query in GridVine. The user initiates a query via the Gridella GUI, which hands it over to the P-Grid component to perform the actual search. The parameter type defines the type of data to search for (GUID, File, RDF, RDFS, OWL), and is implicitly assigned by the system as the user interacts with the front-end. The query is then routed to other peers storing (some of) the results as described in Section 8.5. If a peer receives a query, it checks whether it can answer the query, i.e., wether it is responsible for the partition of the key space corresponding to one of the patterns in the query, otherwise the query gets forwarded as shown in Figure 8.8.

Once arrived at its destination, the query is processed locally by the destination peer as shown in Figure 8.9. The peer checks the type of

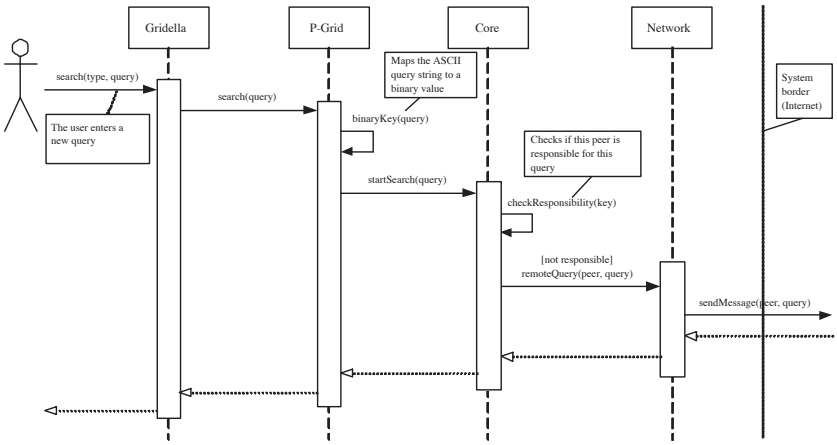

Figure 8.8: Initiating a query.

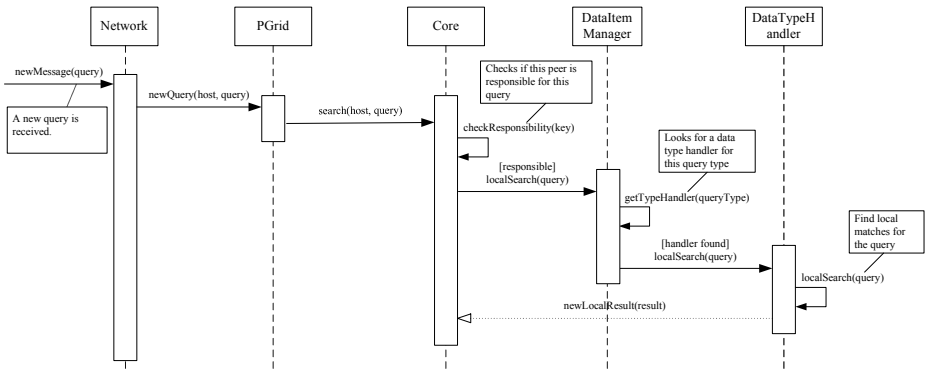

Figure 8.9: Handling an incoming query.

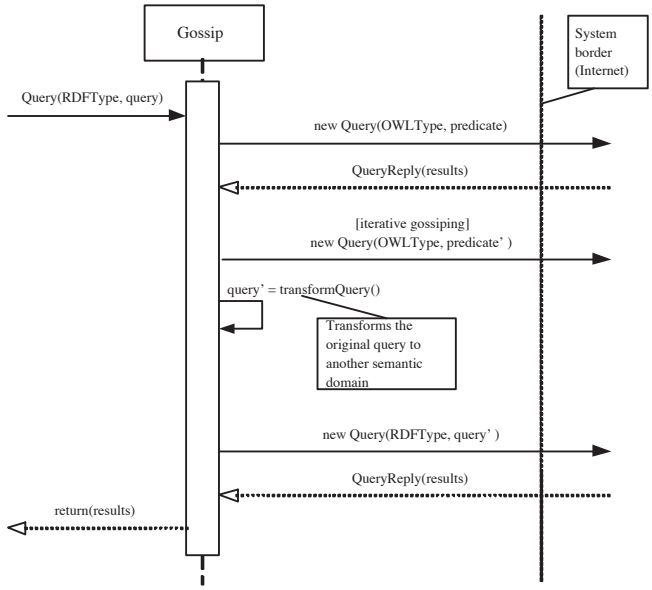

Figure 8.10: Flow of the Gossiping mechanism.

the query (GUID, File, RDF, RDFS, OWL) and hands it over to the corresponding datatype handler, which processes the query according to its type. Datatype handlers are defined and registered within the P-Grid library by the users of the library, i.e., the *Gridella* and *Semantics* components.

8.7.3 Query Reformulation

The introduction of RDF type queries enables Semantic Gossiping, which is explicitly activated by the issuer of a query through a special flag in the query. The degree of propagation of a given query can be set by the user, by defining similarity thresholds as explained in Section 4.6. Figure 8.10 sketches how gossiping is implemented. In the figure, we just show the flow relevant to the forwarding itself and omit the preceding flow of control (incoming message – P-Grid – Datatype handler [Semantics]) for simplicity.

When reformulating queries through schema mappings, we support two approaches: iterative and recursive. In iterative resolution, the peer issuing the RDF query tries to find and process all the mapping links by itself; it first issues a query to retrieve the mappings capable of transforming the category used in the original query; upon finding a translation, it reformulates the original query into a transformed query (Query') and issues a search for the reformulated query. Furthermore, the gossiping peer issues a query for a translation of the reformulated query (Predicate').

This continues until no more translation is available or the transformed query is considered as being too different from the original query following syntactic and semantic similarity measures.

In recursive resolution, the issuing peer tries to reformulate the query by delegating it rather than doing it itself: first, it looks for mappings for the predicates used in the query and reformulates the query upon finding an appropriate translation. The transformed query is issued and results for the query are returned to the issuer of the query. The receiver of the transformed query follow the same procedure recursively. In case of a conjunctive query, each receivers only reformulates the triple patterns it is currently considering.

8.7.4 Experimental Evaluation

We briefly discuss below an initial performance evaluation of the two Semantic Gossiping techniques GridVine implements. The tests were performed using the current implementation on a Fast Ethernet network of 60 SUN Ultra10 stations (Solaris 8). We first created 15 different semantic domains (i.e., 15 different categories C_0 to C_{15}) related to each other through 15 mappings as depicted in Figure 8.11 a). We chose to organize the mappings in a very regular way (i.e., a tree) in order to get a better grasp on the results obtained; note however that our approach and implementation work equally well on arbitrarily complex mapping graphs (see the preceding chapters).

We launched 15 peers, each on a separate computer and each locally storing a annotated document related to a different category. By issuing a query from the peer using C_0, we can retrieve results from all the 15 semantic domains by forwarding the query through the mapping hierarchy. A second setting was created by replicating this first setting four times, running 60 peers using the same category setting (i.e., we then had 4 peers for each category).

The results, time elapsed versus quantity of results (up to 15/60 results) received by the peer issuing the query, for both settings and for iterative and recursive forwarding are displayed in Figure 8.11 b). As expected, iterative forwarding works in a fairly linear manner. Also, note the initial delay incurred by letting one peer process and send all the queries for iterative forwarding with 60 peers. Our recursive approach proceeds in a succession of stages, as it delegates the whole process of query forwarding to intermediary peers. This second approach proves to be particularly scalable with the number of peers: results are rather independent of the number of peers or results returned, since the number of peers processing and forwarding the query increases with the network size. Note that the evaluation performed above was based an initial version of the P-Grid library; we are currently redeploying GridVine using a new, more efficient version of the P-Grid access structure (see Section 8.9).

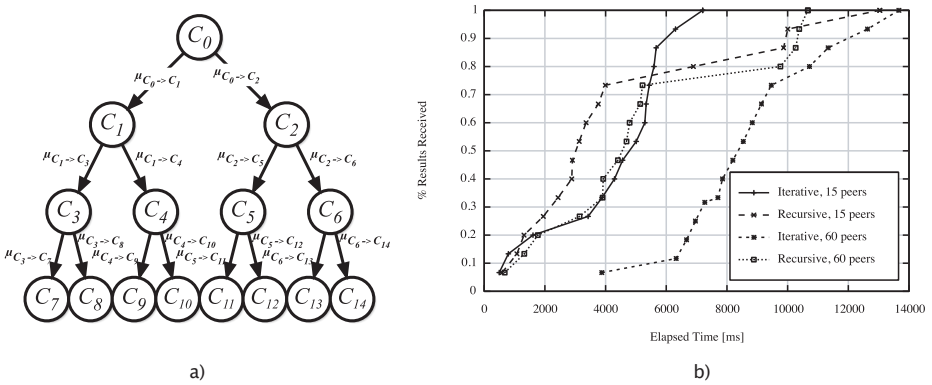

Figure 8.11: Semantic gossiping evaluation on a semantic hierarchy (a) of 15 semantic categories ($C_0, \ldots, C_{14}$) and for 15/60 peers using (b) iterative and recursive forwarding.

8.8 Related Work

Several systems [ACM05] were recently developed to enrich P2P systems with structured or semi-structured data. SWAP [EHS+03] is an approach combining P2P and Semantic Web techniques. It relies on an RDF(S) model and on structure extraction for handling queries in a P2P setting. Edutella [NaCQD+02] employs a super-peer topology and facilitates the clustering of data based on ontology, rule, or query. In PeerDB [OST03], each peer holds a set of locally available metadata (*Local Dictionary*) and a set of metadata that can be accessed by other nodes in the network (*Export Dictionary*). Metadata can be added through an SQL query facility. The system if based on BestPeer [NOT03] which employs mobile agents to satisfy queries. No global schema is imposed but the process is not fully automated, as the user has to decide which mappings are actually meaningful. The Piazza peer data management project [TIaAH+03] takes an approach to semantic heterogeneity that is similar to Semantic Gossiping. Unlike our approach, Piazza does not provide any measures to judge the (in)correctness of mappings. The indexing and query reformulations are centralized in Piazza. OntoBuilder [GMJ04] is a centralized system focusing on the automatic integration of semi-structured information extracted from the Web. It enables fully-automatic ontology matching on ontologies extracted from Web forms, and query reformulation mechanisms to propagate search from one form to the others.

All the above approaches address semantic interoperability but offer limited scalability. Other approaches address scalability but do not deal with semantic interoperability. For example, Peer-to-Peer Information Exchange Retrieval (PIER) [HHL+03] is a database-style query engine built on top of a DHT. Its main focus is to provide database query

processing facilities to widely distributed environments. One of PIER's restrictions is that it imposes global, standard schemas following the rational that some schemas become *de facto* standards. RDFPeers [CF04b] builds on the Multi-Attribute Addressable Network (MAAN), which extends Chord, to efficiently answer multi-attribute and range queries on RDF triples. RDFPeers is a scalable RDF store, but does not provide any schematic support (e.g., to handle user-defined schemas or to address semantic interoperability issues).

8.9 Conclusions

To the best of our knowledge, GridVine is the first semantic overlay network based on an scalable, efficient and totally decentralized access structure supporting the creation of local schemas while fostering global semantic interoperability. Following the principle of data independence, our approach separates the logical and physical aspects such that it can be generalized to any physical infrastructure that provides functionalities similar to our P-Grid P2P system. GridVine continues to evolve today: we are currently working on an improved version of the application in the context of the Social Semantic Desktop project[4]. The new version is based on the latest P-Grid library supporting faster indexing, dynamic load-balancing, identity management and more efficient query handling.

[4]see *http://nepomuk.semanticdesktop.org*

Chapter 9

PicShark: Sharing Semi-Structured Annotations in the Large

With the ubiquitous availability of communication devices, personal information and media sharing is becoming a killer application on the pervasive Web. In such a context, publishing and searching media content heavily relies on the availability of meaningful metadata. Metadata scarcity and heterogeneity are among the key obstacles preventing meaningful media sharing in large communities. However, participating in a community also opens new perspectives for addressing metadata scarcity and heterogeneity through sharing of metadata and semantic knowledge.

Taking advantage of that opportunity, we extend in this chapter some of the result developed in the context of this thesis to tackle the lack of metadata in large-scale collaborative systems. We develop a community-based and self-organizing system call PicShark, in which information entropy – in terms of missing metadata – is gradually alleviated through decentralized instance and schema matching. Our information recontextualization process focuses on semi-structured metadata and confines computationally expensive operations to the edge of the network, while keeping distributed operations as simple as possible to ensure scalability. PicShark builds on GridVine (see previous chapter) for distributed look-up operations, but extends the application of self-organization principles to the bootstrapping of schema mappings and the creation of annotations. We demonstrate the practical applicability of our heuristics in an image sharing scenario and provide experimental evidences illustrating the validity of our approach.

9.1 Introduction

With the explosion of digital communications, the sheer size of information individuals have to cope with is rapidly becoming overwhelming. Metadata have long been recognized as an efficient way to help manage resources and are today widely used by operating systems, personal information managers or media libraries. The general idea is simple: adding a set of keywords or series of attributes to resources in order to facilitate information categorization and retrieval.

While the use of unstructured metadata drew considerable attention in the recent years – e.g., through keyword annotation of images or HTML pages – the focus recently shifted back to more structured metadata formats. Unstructured metadata such as tags are ambiguous by nature and lack precise semantics, making it very difficult to support structured searches *a la* SQL. Structured representations such as relational tables are much easier to process automatically, as they constrain the representation of data through complex data structures and schemas. However, creating those schemas is typically a a complex task left to expert users such as database administrators. As for the preceding chapters, we focus here on novel formats that let end-users freely define and extend their own schemas according to their needs (e.g., XML, RDF/S). We qualify those formats as *semi-structured* formats since they tend to blur the separation between the data and schemas and to impose looser constraints than the relational model to the data.

Semi-structured formats are today gaining momentum; they are flexible enough to allow easy definition and extension of schemas, while structured enough to support automated processing and complex searches (e.g., through languages such as XQuery [BCF+06] or SPARQL [PS06]). More and more applications (see below) take advantage of semi-structured metadata to organize pieces of information *locally*, and the picture annotation domain is a relevant example of that trend. The problem we want to tackle lies in the fact that those applications do not allow to meaningfully share the semi-structured metadata in order to enable global search capabilities in large scale distributed settings. Exploiting semi-structured metadata in distributed environments is intrinsically difficult, as the metadata have to be extracted from their original context and integrated, i.e., *recontextualized*, into the distributed infrastructure.

The two problems we tackle through our recontextualization process are metadata scarcity, i.e., lack of annotations, and semantic heterogeneity, i.e., lack of interoperability. We show in the following that the resolution of each of those two problems influences the other, and thus propose a bottom-up process to solve both of them by iteratively propagating information. Following our approach, global semantics incrementally emerge from the system as a consequence of multiple local interactions. As a result, we show how we can gradually foster global searches on heteroge-

neous metadata in decentralized network, even when most of the metadata are missing initially.

The rest of this chapter presents an approach and an architecture to minimize metadata scarcity and semantic heterogeneity in very large scale, collaborative media sharing environments. Our approach confines computationally expensive operations to the edge of the network and keeps distributed operations as simple as possible to ensure scalability. The contributions of this chapter include:

- the formalization of the problem of sharing semi-structured metadata in distributed settings, in terms of lack of metadata (metadata scarcity) and semantic heterogeneity
- the introduction of metadata entropy to capture uncertainty related to semi-structured metadata
- a bottom-up, emergent semantics recontextualization process to enable global searches in distributed settings through the minimization of both metadata scarcity and semantic heterogeneity
- a system architecture supporting the recontextualization process through a Peer-to-Peer architecture
- an experimental evaluation of our recontextualization process on a large sample set of 300 images.

We start with a general description of the problem in Section 9.2. We give a formalization of the problem in Section 9.3. Our recontextualization approach is presented in Section 9.4. We describe the architecture of our prototype, called PicShark, and experimental findings in Section 9.5. Finally, we give a survey of related work is in Section 9.6 before presenting our conclusions.

9.2 Collaboratively Sharing Semi-Structured Metadata

9.2.1 On the Difficulty of Sharing Semi-Structured Metadata

We focus on semi-structured metadata formats that take advantage of simple schemas to define and organize the metadata. Such formats are today sprouting from various contexts and encompass quite a variety of data models; some of them, such as the Extensible Markup Language (XML), rely on hierarchies of elements to organize metadata. Ontological metadata tie metadata to formal descriptions where classes of resources (and properties) are defined and interrelated. This class of metadata standards is currently drawing a lot of attention with the advent of the

Semantic Web and its associated languages (e.g., RDF/S, XMP[1] or OWL, see Chapter 2). Type-based metadata are rooted in object-oriented technologies, where data (e.g., named attributes, relationships) are bound to strongly-typed class instances. The next generation of the Windows File System (WinFS) is a singular representative of this recent trend.

In the following, we adopt some of the terminology and syntax defined in the context of the RDF/S family of languages. However, we do not constraint the applicability of our approach to those languages and always consider the broader class of semi-structured metadata formats discussed above, which are used by many popular tools today (see also Section 9.5 for concrete examples). We call *resources* the various digital assets that might be shared by the end-users, and *semi-structured metadata* the statements providing semi-structured information on those resources. Users organize themselves in *communities of interest*, which define *vocabulary terms* used in the semi-structured metadata. We define a *schema* as the collection of vocabulary terms defined in a given community of interest.

Our goal is to enable global searches on the resources based on semi-structured metadata shared in large-scale, heterogeneous and decentralized settings. Semi-structured metadata are intrinsically difficult to share, since their values only make sense in a given community of interest – as opposed to keyword metadata or textual tags, which supposedly convey predefined, global semantics. Although semi-structured metadata formats are getting increasingly used, no support is yet provided when it comes to sharing the semi-structured metadata outside of their original community of interest and users are often compelled to export semi-structured metadata as simple unstructured keyword lists.

Some might advocate a straightforward solution to our problem: using a common format, like RDF, for all metadata. Though necessary, we argue that this syntactical alignment step only represents the tip of the iceberg in our case. Even with a global, common format, fundamental problems remain: users would still have a hard time sharing their local metadata, which might be incomplete and totally unrelated to metadata coming from other users.

In the end, two fundamental hurdles prevent semi-structured metadata from being shared:

Metadata scarcity: Though more and more tools rely today on some semi-automatic annotation schemes to add metadata to resources (see Haystack [KBH+05] for an example), fully-automated solutions remain impractical. Most of the time, human attention, which is today considered as one of the scarcest resources, is still required for producing high-quality, meaningful metadata. Realistically, a

[1] http://www.adobe.com/products/xmp/

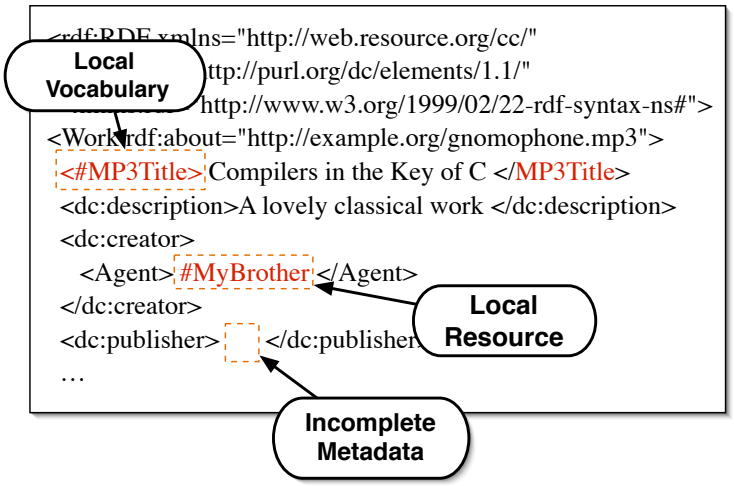

Figure 9.1: The two fundamental hurdles preventing semi-structured metadata from being shared in decentralized settings: metadata scarcity caused by incomplete metadata, and semantic heterogeneity attributable to local vocabulary terms and local resources.

(potentially large) fraction of the shared resources will not be annotated by the user, leaving some (most) of the related semi-structured metadata incomplete. This lack of metadata severely hampers any system relying on the semi-structured metadata to retrieve the resources.

Semantic heterogeneity: Some of the vocabulary terms introduced by end-users to annotate content locally may not make sense on a larger scale. New vocabulary terms – new tag categories or properties used locally by some community – should somehow be related to equivalent vocabulary terms coming from different communities to guarantee interoperability. This is a semantic heterogeneity issue requiring a decentralized integration paradigm, as we have to deal with large-scale, decentralized communities of users without any central authority to enforce vocabulary terms globally (see also Chapter 2). A similar issue arises when a user makes an explicit reference to a local resource in the collaborative setting: the reference can be totally irrelevant to most of the other users who are not aware of the resource in question.

An RDF document exhibiting concrete examples of those two hurdles is reproduced in Figure 9.1.

9.2.2 Opportunities for Reducing Metadata Scarcity and Semantic Heterogeneity Collaboratively

In the rest of this chapter, we tackle the two aforementioned problems in a large scale resource-sharing context. We focus on methods to *re-contextualize* the metadata, i.e., to minimize both metadata scarcity and semantic heterogeneity for the semi-structured metadata attached to the shared resources. Even if sharing semi-structured metadata is intrinsically difficult, we argue that by simultaneously sharing both the resources and their associated metadata in large scale communities, we open the door to new opportunities for supporting global searches on the shared resources.

Assuming that we can relate shared resources semantically inside a given community of interest, e.g., by a low-level analysis of their content or by a semantic analysis of their metadata, metadata can be propagated within the community of interest to reduce metadata scarcity. By taking into account other resources shared in a community, we can thus augment individual metadata by combining local metadata attached to a resource with other metadata originating from similar resources. We call this process *metadata imputation* in Figure 9.2. Data imputation is a field aiming at replacing missing values in a data set by some plausible values (see Farhangfar *et al.* [FKP04] for a recent survey of the field).

By relating resources and metadata coming from different communities, we can further enhance the process by creating schema mappings between semantically related communities of interest. As previously discussed in this thesis, those schema mappings associate vocabulary terms of one community to related terms coming from another community. They allow the reformulation of a query posed against a given schema into a semantically similar query written in terms of another schema. TWe refer to this process as *query propagation* in Figure 9.2, where straight arrows represent mappings between the schemas of two given communities. Schema mappings can reduce semantic heterogeneity by enabling the propagation of a local query across the whole network of communities by following series of mapping links iteratively.

In this chapter, we additionally take advantage of schema mappings to propagate existing metadata across semantically heterogeneous communities, and thus to reduce metadata scarcity. Metadata imputation is this time contingent on the availability of the schema mappings relating the schemas of heterogeneous communities. We refer to this process as *metadata propagation* in Figure 9.2.

In turn, the metadata that have been propagated through a schema mapping can be exploited in order to infer new mappings or verify existing mappings and to increase the accuracy of metadata propagation. This shows a clear correlation between metadata scarcity and semantic heterogeneity, as minimizing metadata scarcity through metadata propagation takes advantage of schema mappings used to minimize semantic

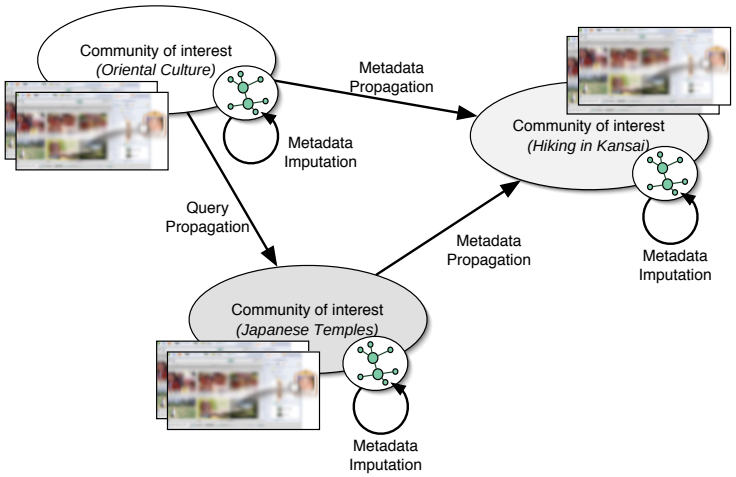

Figure 9.2: Recontextualizing metadata: metadata scarcity is minimized by imputing metadata for similar resources inside a community of interest, while semantic heterogeneity is gradually alleviated thanks to pairwise schema mappings.

heterogeneity, and vice-versa.

In the following, we propose a recontextualization process that reduces the overall scarcity and heterogeneity of the metadata in an autocatalytic process, where both metadata and mappings get reinforced recursively by putting local metadata into a global community-based context. Taking a global view on the system, we observe that the global semantics are not fixed *a priori*, but evolve as users interact with the system and guide the recontextualization process by sharing new resources, by adding new metadata, or by providing positive or negative feedback based on the results retrieved following their queries. The way the semantics of the system dynamically evolve in a bottom-up manner following local interactions is typical of an emergent semantics system (see also the introduction of the thesis).

9.3 Formal Model

In this section we extend our formal model to capture both metadata scarcity and semantic heterogeneity in an information theoretic framework. The problem we want to tackle can be formally introduced as follows: a large set of autonomous information parties we name *peers* $p \in \mathcal{P}$ store *resources* (e.g., calendar entries, pictures, or video files) $r \in \mathcal{R}_p$ locally. Peers take advantage of *schemas* $S \in \mathcal{S}$ to describe their resources with semi-structured metadata. Schemas S_i can be shared by several peers forming a community of interest, and consist of a finite set

of vocabulary terms $t \in \mathcal{T}$. We mainly focus on vocabulary terms representing properties, which invariably exist in one way or another in all the semi-structured metadata formats we have encountered, but classes of resources can be taken into account by our process as well. In the setting we consider, we assume that the number of shared resources is typically significantly higher than the number of peers, which is itself significantly higher than the number of communities: $|\mathcal{R}| >> |\mathcal{P}| >> |\mathcal{S}|$.

Peers store the semi-structured metadata attached to their resources in local databases of triadic relations we call metadata *statements* (r, t, v). A statement (r, t, v) associates a value v to a local resource r through a vocabulary term t. Values v appearing in the statements can either represent literals $l \in \mathcal{L}$ or local resources. Hence, statements can be seen at the syntactical level as RDF triples with constraints on the values of their subject (r), predicate (t) and object (v). A statement evaluates to *true* if it exists in one of the databases of the peers, to false otherwise.

Peers can pose queries locally in order to retrieve specific resources based on vocabulary terms, literals and other local resources. As for the preceding chapter, queries take the form of conjunctions of triple patterns:

$$r? \ : \ (r_1, t_1, v_1), \ldots, (r_n, t_n, v_n)$$

where r_k, t_k, v_k are local resources, vocabulary terms, literals or variables and $i?$ is a distinguished variable appearing in at least one of the triple pattern (r_k, t_k, v_k). Note that joins can be expressed by multiple occurrences of the same variable in that notation. We say that a resource $r_0 \in \mathcal{R}$ is an *answer* to the query q, and write $q \models r_0$, if, when substituted for the distinguished variable, there exists a valuation of all other variables in the conjunction of triple patterns such that the valuation evaluates to *true*.

Now, let us assume that some of the statements are incomplete and that the peers have a means to export resources through some common infrastructure (e.g., the World Wide Web or a Distributed Hash Table). Our goal is to *recontextualize* the local statements in the common infrastructure to support global search capabilities: we create additional statements in such a way that any peer posing a query q against its local schema can retrieve a maximal number of relevant resources $r \mid q \models r$ from the global set of shared resources $\mathcal{R}$ while minimizing false positives and user's involvement under the following restrictions:

Metadata scarcity: Some values v_k appearing in the statements are replaced by null-values $\perp_k$ inducing incomplete statements $(r_k, t_k, \perp_k)$. Null-values are equivalent to the values they replace but cannot be distinguished by the peers.

Semantic Heterogeneity: Each local resource and vocabulary term is assigned a set of fixed interpretations r^I from an abstract and global domain of interpretations Δ^I with $r^I \subseteq \Delta^I$. Arbitrary peers are not

aware of such assignments (i.e., they are not aware of the global semantics of the system). We define two resources r_i and r_j as *equivalent*, expressed by $r_i \equiv r_j$, if and only if $r_i^I = r_j^I$. We define that a resource r_i *subsumes* another resource r_j, expressed by $r_j \sqsubseteq r_i$, if and only if $r_j^I \subseteq r_i^I$. Trueness of statements is relative to the equivalence and subsumption relation, in the sense that if a statement (r, t, v) evaluates to *true*, then all statements $(r', t', v') \mid r' \sqsubseteq r, t' \sqsubseteq t, v' \sqsubseteq v$ also evaluate to *true*.

Taking advantage of those definitions, we can introduce the notions of metadata completeness and soundness. We define that a set of N statements $\{(r, t_1, v_1), \ldots, (r, t_N, v_N)\}$ pertaining to a resource r is *complete* when $v_i \neq \perp \; \forall \; v_i$. A set of statements is *sound* if all the statements evaluate to true. We generally assume that our process starts with sets of statements that are sound but incomplete. Our recontextualization process then tries to complete the statements while minimizing the number of unsound statements generated.

9.4 Recontextualizing Metadata

This section presents our general approach for generating additional metadata to recontextualize shared metadata in large scale settings. A specific implementation of this approach is described in Section 9.5 in the context of an image sharing scenario.

The heterogeneity, autonomy and large number of peers we consider precludes the use of centralized techniques. Traditional integration techniques (e.g., the mediator architecture introduced in Chapter 2) are impractical in our context as no global schema can be enforced in heterogeneous and decentralized communities. Traditional metadata management techniques are not applicable either, due to the lack of shared information (resources, vocabulary terms) and the sheer size of the problem which precludes the use of algorithms scaling exponentially or even linearly with the size of the data.

Instead, we propose local, probabilistic heuristics aiming at recontextualizing metadata extracted from a specific source to a decentralized collaborative context. Following a long tradition of providing scalable application-level services on top of an existing physical network, we push the "intelligence" of the approach towards the edge of the network, i.e., perform all complex operations locally at the peers, while only considering simple in-network operations on a shared hash-table. In the following, we suppose that all resources and peers are identified by globally unique identifiers. Our heuristics are based on decentralized data indexing, data imputation and data integration techniques. We start by defining the notion of entropic metadata that will be used throughout the rest of this chapter to guide the process of metadata self-organization.

9.4.1 Entropic Metadata

As we recontextualize statements in the shared infrastructure, we always keep track of the incomplete metadata (metadata scarcity) attached to the resources. We introduce the notion of metadata entropy to capture the degree of uncertainty related to the incomplete statements. Keeping track of this uncertainty is important to detect the resources requiring further recontextualization (too many incomplete metadata), and to propagate metadata in a meaningful way by associating uncertainty to the metadata that are inferred automatically.

We extend our model to write statement as quadruples $(r, t, \boldsymbol{v}, \boldsymbol{p})$ where $\boldsymbol{v}$ is a list of possible values $v_k \in \boldsymbol{v}$ for the statement and $p_k \in \boldsymbol{p}$ stands for the probability of the statement (r, t, v_k) evaluating to *true*. The *entropy* $H(r, t, \boldsymbol{v}, \boldsymbol{p})$ of a statement measures the degree of uncertainty related to its set of possible values $\boldsymbol{v}$, and evaluates to:

$$H(r, t, \boldsymbol{v}, \boldsymbol{p}) = -\sum_{k=1}^{K} p_k log_K(p_k)$$

where K is the number of possible values in $\boldsymbol{v}$. The entropy of all complete statements exported by the peers is zero, as they consider a single possible value with a probability of 1 of evaluating to *true* (i.e., we consider that all statements stored locally at the peers are correct; if some of these statements are created semi-automatically, we can alternatively start with a smaller value $0 < p < 1$). Incomplete statements $(r, t, \perp)$ start with an entropy of one initially, representing an unknown (and potentially infinite) set of identically distributed values. Their entropy will decrease over the course of our recontextualization process as plausible values get discovered trough metadata imputation and propagation.

We define the entropy of a resource as the normalized sum of the entropy of its N associated metadata statements:

$$H(r) = \sum_{n=1}^{N} H(r, t_n, \boldsymbol{v_n}, \boldsymbol{p_n}) N^{-1}.$$

A resource with half of its metadata left incomplete will thus start with an entropy of 0.5.

9.4.2 Sharing Metadata through Data Indexing

Our recontextualization process starts with the indexing of the shared metadata and resources. We index the location p_0 of each resource r_0 a peer wants to share in a shared hash-table. We then index all metadata statements $(r, t, v, 1)$ pertaining to the resource that has just been indexed. The indexing process continues recursively by indexing all resources r' appearing as values v in the already indexed metadata, and

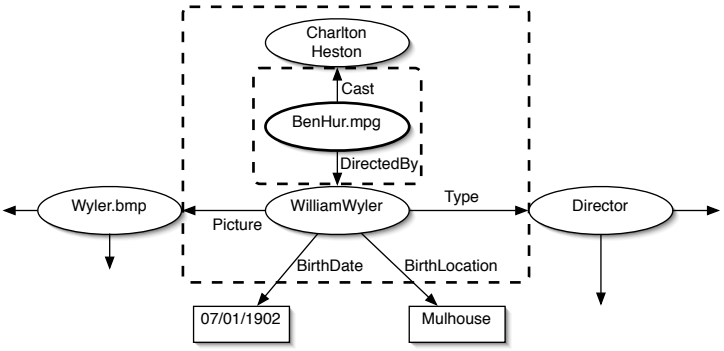

Figure 9.3: Indexing resources and statements from an RDF/S graph; the inner and outer boxes correspond to the a recursion depth limited to respectively zero and one.

their respective statements $(r', t', v', 1)$. Figure 9.3 shows an example of the indexing process on a simple RDF graph with a recursion depths limited to zero and one respectively. All statements are exported using a common representation (e.g., XML serialization of RDF triples) and are indexed in such a way that they can be retrieved based on their resource r, term t or value v (as promulgated by the GridVine system – see the preceding chapter). Higher recursion values lead to sharing more information, which can then be used in the rest of the recontextualization process to relate semantically similar resources. On the other hand, higher recursion values also impose a higher network traffic and load on the shared hash-table.

9.4.3 Dealing with Metadata Scarcity through Intra-Community Metadata Imputation

We take advantage of data imputation techniques [FKP04] to replace missing values in incomplete statements with plausible values derived from similar statements shared in a given community of interest. In our case, we are confronted to values *missing completely at random*, i.e., metadata can be missing irrespective of the resource they are attached to or their actual value. We base our imputation process on a K-Nearest Neighbor search, which has been shown as being very effective in many contexts [BM03] and has two distinctive advantages in our context: i) it does not require building a predictive model for each predicate for which a value is missing and ii) it can be based on a simple index lookup in the shared hash-table.

We proceed as follows: when indexing a resource, we analyze it and generate *feature values* representing its content and/or metadata statements. Feature values can for example be based on a low-level analysis of the resource (e.g., image analysis) or a lexicographical analysis of its meta-

data (see next section for some concrete examples). Features should be extracted in such a way that similar resources get closely related feature values, which might or might not be verified in practice and which naturally impacts on the effectiveness of our approach (see Section 9.5.2). Feature extractors might be different for different types of resources (e.g., pictures, text files, etc.). We index each resource r based on its feature value $FV(r)$ in the shared hash-table to be able retrieve resources with similar feature values. Also, we consider a distance $D(r, r') = |FV(r') - FV(r)|$ based on those values.

For each instance r for which we have to insert at least one incomplete statement $(r, t, \perp)$, we search for K similar resources r' in the hash-table, such that r and r' are annotated using the same schema, $D(r, r')$ is minimal – and below a similarity threshold τ – and $H(r')$ as low as possible. That is, we search for resources coming from the same community of interest that are most similar according to our feature value metric and whose statements are as sound and complete as possible. Probabilistic values $\boldsymbol{v''}$, $\boldsymbol{p''}$ are created for the incomplete statement by combining all related statements $(r'_k, t, \boldsymbol{v'_k}, \boldsymbol{p'_k})$ from the K candidates. Probabilities p''_{kl} are computed by taking into account the distance $D(r, r'_k)$ between the resource in question and the candidate resources whose values are being considered, and the respective probabilities p_{kl} of the set of L values $v_{kl} \in \boldsymbol{v'_k}$ appearing in the statements of the candidate resources:

$$p''_{kl} = \frac{D(r, r'_k)^{-1}}{\sum_{k=1}^{K} D(r, r'_k)^{-1}} \frac{p'_{kl}}{\sum_{l=1}^{L} p'_{kl}}.$$

By doing so, sound statements or statements coming from very similar instances are preferred. We combine the probabilities p''_1 and p''_2 attached to the same value $v'_1 \equiv v'_2$ but coming from two different statements (r'_1, t, v'_1) and (r'_2, t, v'_2) by summing up the two probabilities p''_1 and p''_2. When less than K similar resources exist in the radius of the similarity threshold τ, abstract resources with incomplete statements $(r_\perp, t, \perp)$ with $D(r, r_\perp) = \tau$ are considered. Figure 9.4 shows an example of the imputation process for an incompletely annotated movie file r, combining the statements of its two nearest neighbors r' and r''. Note that a similar imputation process could again take place later on, once the statements have already been recontextualized but new resources have been indexed, for example periodically every T period of time for resources with a high entropy. In a dynamic context and for high values of K, one should additionally avoid storing too many unlikely values by eliminating all values with low probabilities ($p''_{kl} < p_{min}$).

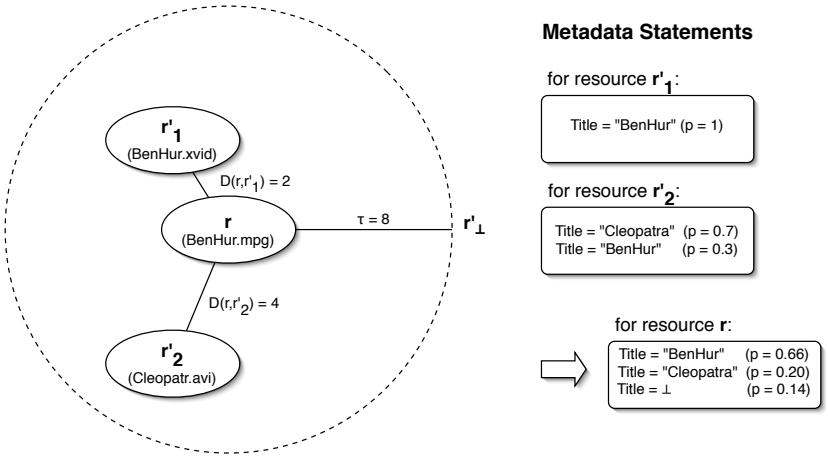

Figure 9.4: An example of data imputation: statements coming from two nearby candidate resources r'_1 and r'_2 and an abstract instance $r_\perp$ are combined to complete the statements attached to resource r whose *Title* was missing.

9.4.4 Dealing with Semantic Heterogeneity with Pairwise Schema Mappings

We address semantic heterogeneity issues by creating pairwise mappings linking semantically related vocabulary terms. We extend the application of self-organization principles to the creation of the mappings based on the information available in the shared hash-table. Those mappings can be used to identify equivalent terms in the data propagation process (see below Section 9.4.5), and to reformulate queries iteratively as in a peer data management system – see Chapter 3.

A peer p indexes into the shared hash-table all locally known equivalence relations $(t', \equiv, t'', 1)$, with $t', t'' \in S_p$ pertaining to the terms appearing in its statements. We then try to discover cross-schemas relationships in a decentralized fashion. Automatic schema matching is an active area of research [Euz04b] and is not the focus of this work. For our purpose, we use a simple instance-based schema matching approach by piggybacking on the imputation process: we create a new mapping $(t', \equiv, t'', p_\equiv)$ whenever two statements (r', t', v', p') and (r'', t'', v'', p'') on two similar resources r' and r'' with $D(r', r'') < \tau$ with equivalent values $v' \equiv v''$ are discovered. The probability $p_\equiv$ that this relation holds is derived by retrieving analogous statements (r_j, t', v_j, p_j) (r_k, t'', v_k, p_k) from the shared hash-table:

$$p_\equiv \quad = \quad \frac{\sum p_j \ \forall (r_j, t', v_j, p_j), (r_k, t'', v_k, p_k) | D(r_j, r_k) < \tau \wedge v_j \equiv v_k}{\sum p_j \forall (r_j, t', v_j, p_j), (r_k, t'', v_k, p_k) | D(r_j, r_k) < \tau}.$$

The probability is thus computed by counting the number of equivalent values appearing in instances considered as being similar for the two terms. Incomplete assertions (i.e., assertions with $v = \perp$) are not considered in these computations. Unsound assertions (i.e., assertions with $p < 1$) can be taken into account in this process by weighting their importance with their likeliness (i.e., less likely values v_i with probabilities p_i close to zero will have less impact than more probable values). Note that subsumption mappings can be exported and discovered in an identical manner by taking into account subsumption relations $\sqsubseteq$ in place of the equivalence relations above.

9.4.5 Dealing with Metadata Scarcity through Inter-Community Metadata Propagation

Schema mappings can be used to propagate metadata across different communities of interest. The process is similar to the imputation process confined to a single community of interest (see Section 9.4.3), but takes into account schema mappings $(t, \equiv, t', p_{\equiv})$ to impute values based on equivalent vocabulary terms t and t'. Probabilities attached to values retrieved through a schema mapping are always multiplied by the probability attached to that mapping $p_{\equiv}$ to account for the fact that the mapping is in itself uncertain. Note that a value can be propagated iteratively across series of communities of interest in that manner, and that propagated values can in turn bootstrap the creation of new schema mappings.

9.4.6 Possible Answers and User Feedback

User queries $r? : (r_1, t_1, v_1), \ldots, (r_n, t_n, v_n)$ can be resolved by iterative lookup on the shared hash-table (see the preceding chapter): for each triple pattern in the query, candidate triples are retrieved by looking-up one of the constant terms of the triple pattern in the shared hash-table. Answers to the query are then obtained by combining the candidate triples. In addition to the certain answers obtained in that way, *possible* [DS04] answers are generated by reformulating queries following (probabilistic) schema mappings to query distant communities of interest:

$$r'? : (r_1, t'_1, v_1), \ldots, (r_n, t'_n, v_n) \mid (\exists S_j \mid t'_1, \ldots, t'_n \in S_j) \wedge t'_1 \equiv t_1, \ldots, t'_n \equiv t_n$$

and by taking into account the probabilities attached to the values of the entropic statements generated by the metadata imputation and propagation processes. The resulting answers can be ranked with respect to their likelihood to present the most likely results first to the user. Additionally, resources with a high entropy (i.e., resources with many incomplete or unsound statements) can be at this stage proposed to the user in order to

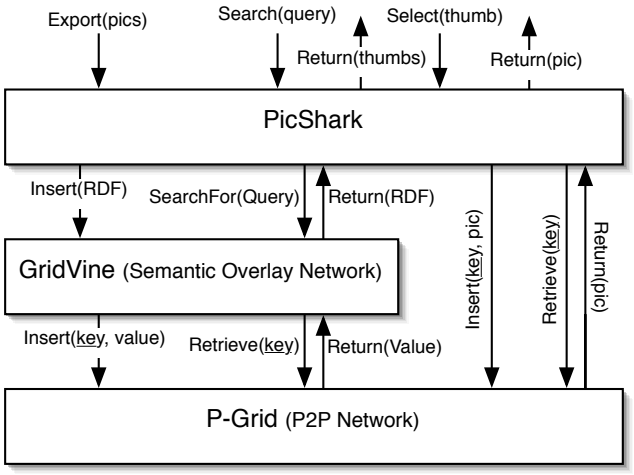

Figure 9.5: The PicShark architecture: PicShark uses P-Grid to store shared resources and GridVine to share semi-structured metadata.

take advantage of his feedback to classify those highly uncertain resources and to bootstrap a new data imputation round.

9.5 PicShark: Sharing Annotated Pictures in the Large

To demonstrate the viability of our metadata recontextualization strategies, we are developing a system called *PicShark*. PicShark is an application built on top of a semantic overlay network allowing global searches on shared digital pictures with incomplete, local and semi-structured annotations.

Our approach follows the principle of data independence by separating the logical layer, a semantic overlay managing structured metadata and schemas, from the physical layer consisting of a structured peer-to-peer overlay network for efficient routing of messages (see Figure 9.5). The physical layer is used to implement various functions at the logical layer, including query resolution, information imputation and integration.

We use P-Grid [ACMD+03] as a substrate for storing all shared information in a Distributed Hash-Table (DHT). Indexing of statements is handled by GridVine, described in the preceding chapter. GridVine implements heuristics for storing RDF/S triples in a decentralized way, and facilitate efficient resolution of conjunctive queries in $\mathcal{O}(log(n))$ messages, where n is the number of peers in the system.

On top of this architecture, PicShark takes care of fostering global semantic interoperability by recontextualizing local statements exported

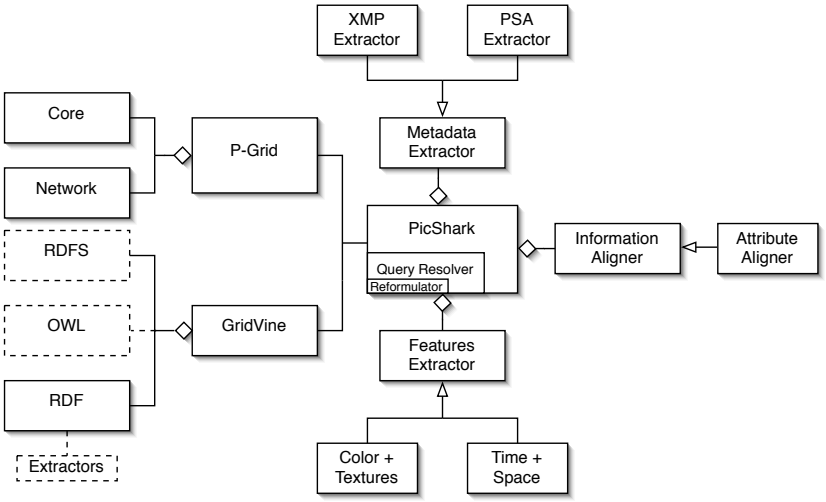

Figure 9.6: The PicShark components: PicShark uses *metadata extractors* to align metadata on a syntactic level, *aligners* to align schemas on a semantic level, and *feature extractors* to relate semantically similar images.

to the P2P network. Users can export sets of local pictures through the *export(pics)* functionality and search for pictures by specifying simple conjunctive queries against their local schemas.

Figure 9.6 gives an overview of the various components used in PicShark. Data indexing takes advantages of *metadata extractors* (see below) to syntactically align the statements before sharing them through GridVine. Creation of mappings is handled by *aligners*, while *feature extractors* are used to extract the feature vectors used to relate similar images.

9.5.1 Information Extraction in PicShark

Metadata extractors are used in PicShark to extract local metadata from the images and to syntactically align them to a common representation. PicShark uses RDF/S as semantic interlingua, and converts all supported metadata formats to this representation. The application currently supports two very different semi-structured metadata formats: PSA, which is a proprietary format used by Photoshop Album[2] and based on structured hierarchies of tags, and XMP, which is a standard based on RDF/S. Both standards are extensible and let users define new vocabulary terms to annotate their pictures. The *PSA Extractor* extracts semi-structured statements and vocabulary terms from the local relational database used by Photoshop Album to store all metadata, while its XMP counterpart

[2]http://www.adobe.com/products/photoshopalbum/

extracts statements and vocabulary terms from the payload of the pictures. The extractors generate all missing GUIDs (for local vocabulary terms and pictures), index statements using GridVine and images using P-Grid directly.

Features can be extracted from the images by a low-level analysis based on sixty texture and color moments, or by the extraction of spatial and temporal metadata from the images; with time-stamps directly embedded into the images and with the proliferation of GPS devices and localization services (such as ZoneTag[3]), we believe that the combination of both temporal and spatial information represents a new and computationally inexpensive way of identifying similar images (see also below for a discussion on that topic).

9.5.2 Performance Evaluation

Evaluating the performance of a system like PicShark is intrinsically difficult for several reasons: first, PicShark is (to the best of our knowledge) the first application taking advantage of semi-structured, local, heterogeneous and incomplete metadata statements. As statements from other popular image classification applications such as Photoshop Album or Extensis Portfolio[4] are not searchable in the large, constituting a realistic and sufficiently large data set is currently difficult. Second, recontextualization is a highly recursive, distributed and parallel process, such that getting a clear idea of the ins and outs of the process is difficult for large data sets or numerous peers. In the following, we detail series of controlled experiments pertaining to a set of three hundred photos[5], which were manually annotated using Adobe Photoshop Album Starter Edition 3. The set of photos is divided into three subsets, each taken by a different person during a common trip to Japan. The first two subsets were annotated using the same schema, while the third subset was annotated using a different – but semantically related – schema. Temporal information was directly taken from the time-stamp embedded by the cameras, while spatial information was added manually to each picture.

Intra-Community Recontextualization

We start by exporting the first two subset of 100 images each, along with their metadata. We drop each statement – except spatial and temporal information, which are always preserved – with a probability $pMissing$ to simulate metadata scarcity. We then recontextualize the 100 images from the first subset one by one using images and statements from the second subset to simulate intra-community recontextualization (remember that

[3] http://research.yahoo.com/zonetag/

[4] *http://www.extensis.com/*

[5] both photos and semi-structured metadata are available at *http://lsirpeople.epfl.ch/pcudre/PicShark*

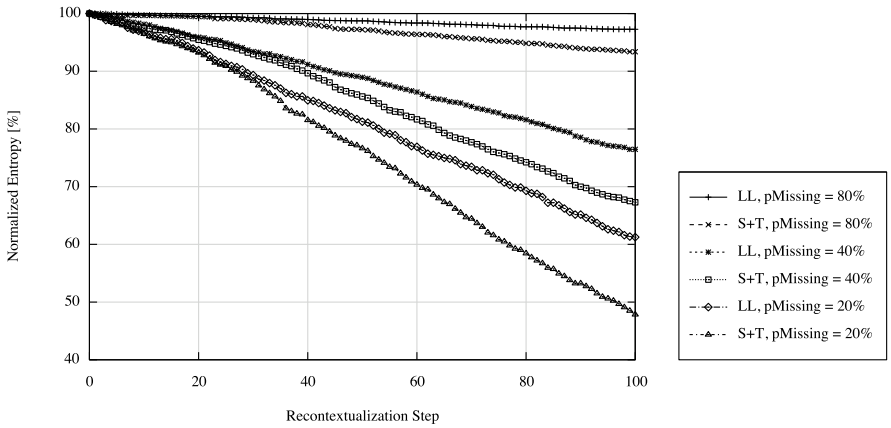

Figure 9.7: Normalized total entropy pertaining to the first subset of images, for metadata missing with various probabilities $pMissing$; at each step, we recontextualize one of the 100 images from the first subset with its two nearest neighbors from the second subset, using either low-level features (LL) or spatial and temporal information ($S+T$).

both subsets use the same schema). We alternatively base the imputation process on either low-level features or spatial and temporal metadata. This experiment is thus interesting to analyze results pertaining to metadata imputation in a given community of interest. As our image set is pretty homogeneous, we set $\tau = \infty$ and $K = 2$, i.e., we always take the two nearest neighbors to recontextualize a given picture.

Figure 9.7 shows the evolution of the total entropy pertaining to the first subset of photos during the recontextualization process, for various values of $pMissing$ ranging from 20% to 80%. The figure gives a normalized value of the total entropy (the absolute entropies start at 82, 166, and 329 for $pMissing = 20\%$, 40%, and 80% respectively). The curves represent the average value obtained over 10 consecutive runs. Note that the results are pretty stable: the standard deviation never exceeds 10% of the absolute value. The entropy – and thus, the uncertainty on the set of images – decreases as more and more pictures get recontextualized. The imputation process based on spatial and temporal values ($S + T$) is slightly better than the process based on low-level features (LL) at finding images with very related statements. For high $pMissing$ values, many values are missing and fewer metadata statement get propagated.

The impact of the nearest-neighbor search is best illustrated in Figure 9.8 and Figure 9.9, which respectively depict the aggregated probability for the sound and unsound metadata generated by the system. We call aggregated probability the sum of all probabilities attached to propagated metadata (propagated metadata with $\perp$ values are not taken into account). Note that propagating metadata usually decreases the total

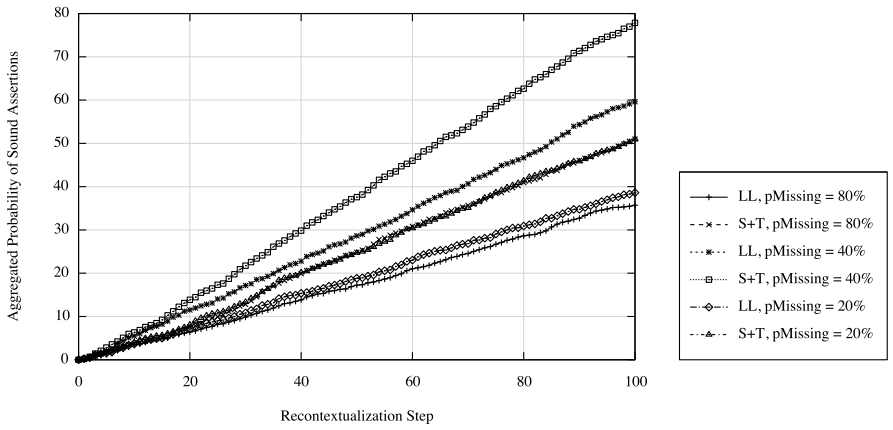

Figure 9.8: Aggregated probability of the sound statements generated by the system, for metadata missing with various probabilities $pMissing$; at each step, we recontextualize one of the 100 images from the first subset with its two nearest neighbors from the second subset, using either low-level features (LL) or spatial and temporal information ($S + T$).

entropy of the system, except when highly uncertain metadata are generated (e.g., when a $\perp$ value is replaced by two generated values with 50% probability each). $S + T$ is systematically better than LL at finding good neighbors, as it always generates more sound and less unsound statements than LL. This is not surprising, as finding similar photos based on color and textures moments only is known to be a difficult problem in general. $S + T$ generates high-quality metadata that are sound more than 80% of the time. On the other hand, $S + T$ is often wrong when propagating metadata about people appearing on the pictures: here, spatial and temporal information is typically not sufficient and a combination of both $S + T$ and LL would probably be more efficient.

In absolute terms, more statements are propagated for $pMissing = 40\%$. For $pMissing = 20\%$, few metadata are propagated (few values are missing), while for $pMissing = 80\%$, few values are available for propagation initially.

Inter-Communitiy Recontextualization

In the the second part of the experiment, we continue the recontextualization process started above and further recontextualize the 100 photos coming from the first subset with 100 photos coming from the third subset annotated with a different schema. In that way, we simulate the creation of mappings and the propagation of metadata coming from different communities of interest.

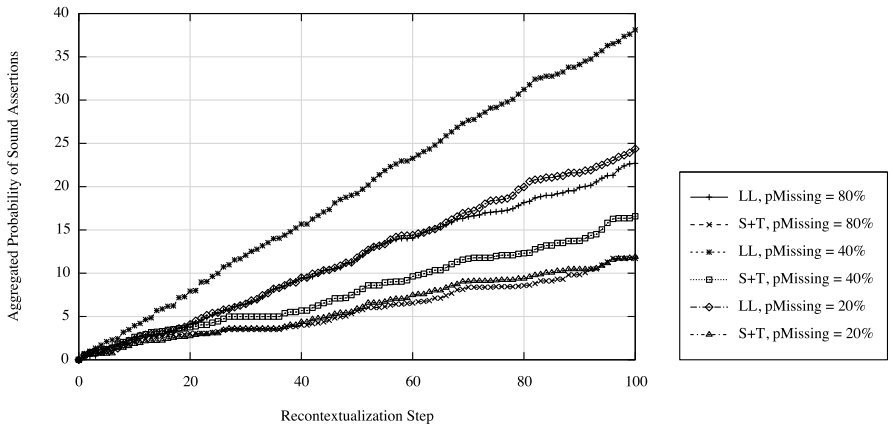

Figure 9.9: Aggregated probability of the unsound statements generated by the system, for metadata missing with various probabilities $pMissing$; at each step, we recontextualize one of the 100 images from the first subset with its two nearest neighbors from the second subset, using either low-level features (LL) or spatial and temporal information ($S + T$).

First, the third set of images and their related metadata are exported. We do not drop metadata in the third set, thus simulating a large set of metadata encoded according to different schemas. Schema mappings are created between the two schemas using the method described above. Once the mappings are created, we further recontextualize each of the 100 images of the first set with their two closets-neighbors from the third set. We only use $S + T$ this time, as LL systematically yields inferior results as for the intra-community recontextualization step described above. Figure 9.10 gives the evolution of the normalized entropy for the first set of images. More uncertain metadata are propagated than for the previous case due to mappings, which were generated totally automatically based on the values of the statements and are uncertain in this case. Images with a high entropy (e.g., for $pMissing = 80\%$), however, benefit a lot from this second recontextualization round, since their statements were still largely incomplete after the first recontextualization round and since all statements from the third set are complete.

Figure 9.11 shows the aggregated probability of the sound statements generated during this second round of recontextualization. Unsound statements follow a similar trend, but never represent more than 20% of the generated statements. At the end of our recontextualization process and depending on the value of $pMissing$, 60 to 75% of the initial entropy of the system induced by incomplete metadata has been alleviated. Most statements contain now entropic metadata that are sound in their majority (less than 20% of the propagated statement are unsound on average

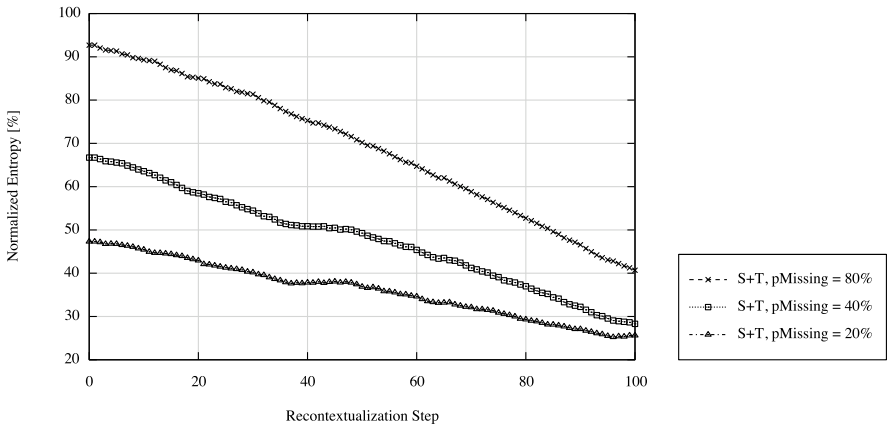

Figure 9.10: Normalized total entropy pertaining to the first subset of images, for metadata missing with various probabilities $pMissing$; at each step, we recontextualize an image from the first subset of images with its two nearest neighbors from the third subset, based on spatial and temporal information ($S + T$).

with $S + T$). Also, schema mappings relating the two communities of interest have been created automatically. Thus, we are now able to query the system and retrieve relevant images from both communities, while this was totally impossible before the recontextualization process because of the lack and of the heterogeneity of the metadata.

9.6 Related Work

To the best of our knowledge, our approach is the first approach aiming at recontextualizing semi-structured, heterogeneous and scarce metadata in large scale decentralized environments. Our work is at the confines of decentralized data integration, personal information management and data imputation techniques.

The way we propagate queries is typical of a new type of large scale semantic infrastructures named peer data management (see Chapter 3). Stuckenschmidt et al. [SVHB04] recently addressed the problem of integrating distributed RDF repositories with a central mediator. They focus on the optimization of query resolutions but does not directly address semantic heterogeneity. REMINDING [TSW04] is an emergent semantics approach where the network memorizes which queries are successfully answered by which peers in order to optimize the routing of future and related queries. More recently, Aurnhammer et al. [AHS06] proposed an emergent semantics approach to retrieve images based on collaborative tagging. Their approach is however limited to unstructured annotations

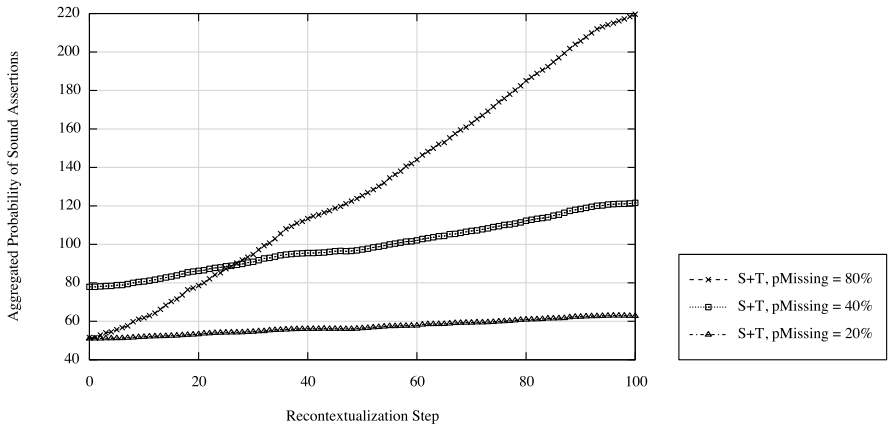

Figure 9.11: Aggregated probability of the sound statements generated by the system, for metadata missing with various probabilities $pMissing$; at each step, we recontextualize an image from the first subset of images with its two nearest neighbors from the third subset, based on spatial and temporal information ($S + T$).

and does not consider metadata scarcity or heterogeneity.

Haystack [KBH+05] is an information management system, which uses extractors and lets non-technical users teach the application how to extract semantic Web content to generate RDF triples from various sources. In Haystack, search is a user-centric process handled by orienteering, i.e., the iterative reformulation is handled entirely by the users and is not automated. Gnowsis [FHG+06] is a collaborative semantic desktop where semantic information is collected from different applications on the desktop and integrated with information coming from external tagging portals. Semantic annotations are either extracted or derived from user's interactions. The Semex System [DH05] is a platform for personal information management. Semex reconciles heterogeneous references to the same real-world object using context information and similarity values computed from related entities. The system leverages on previous mappings provided by the users and on object and association databases to foster interoperability. Reconciliation of data was also recently revisited in the context of the ORCHESTRA [TI06] project; in ORCHESTRA, participants publish their data on an ad hoc basis and simultaneously reconcile updates with those published by others. Individual updates are associated with provenance information, and each participant only accepts updates with a sufficient authority ranking, meaning that each participant may have different (though conceptually overlapping) data instances.

Data imputation denotes techniques aiming at replacing missing values in a data set by some plausible values (see Farhangfar *et al.* [FKP04] for a recent survey of the field).

9.7 Conclusions

With the rapid emergence of socially-driven applications on the Web, self-organization principles have once again proven their practicability and scalability: through Technorati Ranking[6], Flickr Interestingness[7] or del.icio.us recommendations[8], an ever-increasing portion of the Web self-organizes around end-users semantic input. In this chapter, we advocated a decentralized, community-based and imperfect (in terms of soundness and completeness) way of integrating semi-structured metadata through self-organizing assertions. Our PicShark system aims at automatic metadata creation by using intra and inter-domain propagation of entropic statements and schema alignment through decentralized instance-based schema matching.

PicShark represents a first proof-of-concept of the applicability of self-organization principles to the organization of semi-structured, heterogeneous and partially annotated content in large-scale settings. We showed in our experiments how an incomplete set of metadata can be enhanced collaboratively using our approach. To the best of our knowledge, PicShark is currently the only system capable of using incomplete and heterogeneous data sets such as the one we used to foster global, structured search capabilities automatically. This first implementation effort opens the door to many technical refinements. As future work, we plan to improve our imputation process to include personalized and fuzzy classification rules to relate semantically similar content. Also, we intend to analyze the system churn – in terms of total entropy, user feedback, and recently indexed instances, metadata or mappings – in order to determine the optimal scheduling of recontextualization rounds. Finally, we want to improve the deployability of our application in order to test our approach *in situ* on large and heterogeneous communities of real users, and are currently launching an initiative jointly with an art center in that context.

[6] *http://www.technorati.com/*
[7] *http://www.flickr.com/*
[8] *http://del.icio.us/*

Chapter 10

Conclusions

The last decade saw the rise of various mechanisms for organizing minimally-structured, human-processable data in the large, from ranking of HTML pages at the scale of the Web to classification of keyword-annotated digital images. Today, we believe that a new revolution targeting declarative, semi-structured machine-processable information is on its way. End users, who used to be restricted to passively consuming manually curated digital information, are today evolving into industrious supervisors of semi-automatic processes creating digital artifacts on a continuous basis. *Peer production* [Ben05], where decentralized communities of individuals collaborate to create complex digital artifacts, or *human computation* [vA06], where, interestingly, computational processes perform their function via *outsourcing* certain steps to human agents, are just two facets characterizing this evolving trend towards a data industrial revolution. Networks of computers, yesterday considered as a convenient medium to store-and-transmit human-targeted information, are today evolving into autonomous spaces consuming, transforming, but also producing their own information. As structure is still inherently implied by all machine-processable data, we believe that this revolution represents a formidable challenge towards creating next generation information management algorithms, relying on increasingly complex – but also uncertain – digital information to support higher-level data processing.

Throughout this thesis, we advocated a human-inspired but machine-targeted, bottom-up view on the problem of semi-structured data integration in large scale settings. We introduced a holistic view on semantics by focusing on implicit agreements through transitive and large-scale analyses of schema mappings simultaneously relating sets of heterogeneous representations of structured information. Also, we presented system architectures and experimental evidences supporting the validity and applicability of our concepts. We tackled four specific issues related to the current ecology of the Web (see Chapter 3): scalability, through P2P architectures and decentralized communications, uncertainty, by explic-

itly modeling the sets of *possible* information, dynamicity, through self-organizing and continuous processes, and finally limited expressivity, by concentrating on fundamental constructs of various data representation languages. As more and more digital information is today generated in automated and decentralized manners, it is getting more and more important to support incremental interoperability mechanisms to meaningfully process data in the large [HFM06]. Emergent semantics, as described in our work, is currently receiving increasing attention from the research community in that context – as indicated by the recent invited papers [ACMO04, ACCM+04] or special issue [ACM06] on the subject, or by the Emergent Semantics EU challenge[1]. We see the emergent, decentralized phenomena we fostered and analyzed throughout this thesis as not only complementing the traditional approaches organizing semi-structured information through top-down consensus creation, but also as the only resort for organizing data in the distributed, autonomous and complex data spaces currently emerging.

Directions for Future Research

The novel results introduced by our research open the door for a multitude of improvements and further developments. A first important effort is the development of various tools to test our algorithms on large populations of users. Towards that direction, we are currently implementing some of our algorithms in the context of the Nepomuk Social Semantic Desktop[2], with the explicit ambition of producing a fully functional, distributed desktop managing semi-structured information in emergent semantics ways in a couple of years.

From a more theoretical side, we are currently dissatisfied with the ways used to characterize semantic correspondences in standard data representation formats. We believe it would be beneficial to characterize correspondence of classes at a finer granularity, for example using probabilistic description logics approaches, and to formalize possible-worlds schema mappings and their implications. Along the same lines, we believe it is important to extend emergent semantics processes to a wider palette of language constructs, including relational constraints and description logics-based ontological properties.

While focusing on decentralized and collaborative mechanisms, we did not take into account security and trust-related issues in our work. Security and trust are essential aspects of P2P architectures and have received increased attention from the research community recently [DA06]. As for all other endeavors based on distributed processes or computations, we believe it will be important to tackle those issues in order to maximize the

[1] *http://complexsystems.lri.fr/Portal/tiki-index.php?page=Emergent%20semantics*
[2] *http://nepomuk.semanticdesktop.org/*

utility of our systems, e.g., for handling freeloading behaviors or detecting malicious nodes. Also to be tackled, the tractability of all transformation processes, in order to guarantee accurate prior information on the various pieces of information scattered throughout the system.

Finally, note that the view we took on mappings, schemas and data throughout this thesis was almost always systematically biased: mostly, we focused on uncertain schema mappings and considered schemas and data as being readily available and processable (with one notable exception in Chapter 9). In emergent environments where both data and schemas can be missing, uncertain, or generated on the fly, a better way to look at the problem would be to consider all pieces of information on uncertain and reciprocal bases. We already proposed the creation of data based on uncertain mappings in the context of PicShark. Under different circumstances, it might for example be beneficial to curate schemas based on available mappings. In the end, we believe we should regard data, schemas and schema mappings as complementary elements shaping a triadic structure, used by collections of autocatalytic processes fostering the reinforcement of uncertain information throughout the system.

List of Frequently Used Symbols and Abbreviations

Greek Symbols

δ	Probability that two (or more) unsound mappings compensate each other and create a sound reformulation along a mapping path
$\mu_{S_i \to S_j}$	A schema mapping allowing to reformulate a query posed against schema S_i into a new query posed against S_j
π	The projection operator used in structured queries and schema mappings
ρ	The renaming operator used in structured queries and schema mappings
σ	The selection operator used in structured queries and schema mappings
τ	A semantic threshold on the soundness of a mapping

Latin Symbols

A_i	An attribute part of a schema S
f	Feedback information gathered from the network of mappings
$f_{\circlearrowright}$	Feedback information gathered by analyzing cycles in the network of mappings
$f_{\rightrightarrows}$	Feedback information gathered by analyzing parallel paths in the network of mappings

$f_{\rightleftarrows}$	Feedback information gathered by analyzing results received from other peers
m_i	An attribute mapping, which maps one or several attributes from a target schema onto an attribute A_i from a source schema
$M_{f()\rightarrow x}(x)$	A message between a factor node $f()$ and a variable node x used in sum-product message passing computations
p_i	A peer representing an autonomous information source connected to the network
p_{jk}	Probability of a random schema of having in-degree j and out-degree k in a network of mappings
q	A structured query
S_i	A structured schema constituted of a set of attributes $\boldsymbol{A} = \{A_1, \ldots, A_j\}$

Abbreviations

DB	Database
DHT	Distributed Hash Table
GAV	Global As View
GUID	Globally Unique Identifier
ID	Identifier
LAV	Local As View
P2P	Peer-To-Peer
PDMS	Peer Data Management System
RDF	Resource Description Framework
SON	Semantic Overlay Network
SRS	The Sequence Retrieval System
TTL	Time-To-Live
XML	Extensible Markup Language

Bibliography

[AB02] R. Albert and A.-L. Barabási. Statistical mechanics of complex networks. *Reviews of Modern Physics*, 74(47), 2002.

[Abe01] K. Aberer. P-Grid: a self-organizing access structure for P2P information systems. In *International Conference on Cooperative Information Systems (CoopIS)*, 2001.

[Abe02] K. Aberer. Efficient Search in Unbalanced, Randomized Peer-To-Peer Search Trees. Technical Report IC/2002/79, Swiss Federal Institute of Technology, Lausanne (EPFL), 2002. http://www.p-grid.org/Papers/TR-IC-2002-79.pdf.

[ACCM+04] K. Aberer, T. Catarci, P. Cudré-Mauroux, T. Dillon, S. Grimm, M. Hacid, A. Illarramendi, M. Jarrar, V. Kashyap, M. Mecella, E. Mena, E. J. Neuhold, A. M. Ouksel, T. Risse, M. Scannapieco, F. Saltor, L. de Santis, S. Spaccapietra, S. Staab, R. Studer, and O. De Troyer. Emergent Semantics Systems. In *International Conference on Semantics of a Networked World (ICSNW)*, 2004.

[ACG+06] P. Adjiman, P. Chatalic, F. Goasdoué, M.-C. Rousset, and L. Simon. Distributed Reasoning in a Peer-to-Peer Setting: Application to the Semantic Web. *Journal of Artificial Intelligence Research*, 25, 2006.

[ACM05] K. Aberer and P. Cudré-Mauroux. Semantic Overlay Networks. In *International Conference on Very Large Data Bases (VLDB)*, 2005.

[ACM06] K. Aberer and P. Cudré-Mauroux (Eds.). *Journal on Data Semantics VI: Special Issue on Emergent Semantics*. Springer Verlay, 2006.

[ACMD+03] K. Aberer, P. Cudré-Mauroux, A. Datta, Z. Despotovic, M. Hauswirth, M. Punceva, and R. Schmidt. P-grid: A self-organizing structured p2p system. *ACM SIGMOD Record*, 32(3), 2003.

[ACMH02] K. Aberer, P. Cudré-Mauroux, and M. Hauswirth. A Framework for Semantic Gossiping. *SIGOMD RECORD*, 31(4), December 2002.

[ACMH03a] K. Aberer, P. Cudré-Mauroux, and M. Hauswirth. Start making sense: The Chatty Web approach for global semantic agreements. *Journal of Web Semantics*, 1(1), 2003.

[ACMH03b] K. Aberer, P. Cudré-Mauroux, and M. Hauswirth. The Chatty Web: Emergent Semantics Through Gossiping. In *International World Wide Web Conference (WWW)*, 2003.

[ACMHvP04] K. Aberer, P. Cudre-Mauroux, M. Hauswirth, and T. van Pelt. GridVine: Building Internet-Scale Semantic Overlay Networks. In *International Semantic Web Conference (ISWC)*, 2004.

[ACMO04] K. Aberer, P. Cudré-Mauroux, and A. M. Ouksel (Eds.). Emergent Semantics Principles and Issues. In *International Conference on Database Systems for Advanced Applications (DASFAA)*, 2004.

[ACPS96] S. Adali, K.S. Candan, Y. Papakonstantinou, and V.S. Subrahmanian. Query caching and optimization in distributed mediator systems. In *SIGMOD International Conference on Management of Data*, 1996.

[AD98] S. Abiteboul and O. Duschka. Complexity of answering queries using materialized view. In *Symposium on Principles of Database Systems (PODS)*, 1998.

[AHS06] M. Aurnhammer, P. Hanappe, and L. Steels. Integrating collaborative tagging and emergent semantics for image retrieval. In *Collaborative Web Tagging Workshop*, 2006.

[AKK+03] M. Arenas, V. Kantere, A. Kementsietsidis, I. Kiringa, R.J. Miller, and J. Mylopoulos. The Hyperion Project: From Data Integration to Data Coordination. *SIGMOD Record, Special Issue on Peer-to-Peer Data Management*, 32(3), 2003.

[AP03] K. Aberer and M. Punceva. Efficient Search in Structured Peer-to-Peer Systems: Binary v.s. k-ary Unbalanced Tree Structures. In *International Workshop On Databases, Information Systems and Peer-to-Peer Computing*, 2003.

[BCF+06] S. Boag, D. Chamberlin, M.F. Fernández, D. Florescu, J. Robie, and J. Siméon (Ed.). XQuery 1.0: An XML

Query Language. W3C Candidate Recommendation, June 2006. http://www.w3.org/TR/xquery/.

[Ben05] Y. Benkler. *Common Wisdom: Peer Production of Educational Materials*. COSL Press, 2005.

[BG04] D. Brickley and R.V. Guha (Ed.). RDF Vocabulary Description Language 1.0: RDF Schema. W3C Recommendation, February 2004. http://www.w3.org/TR/rdf-schema/.

[BGK+02] P. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data Management for Peer-to-Peer Computing : A Vision. In *International Workshop on the Web and Databases (WebDB)*, 2002.

[BHP92] M.W. Bright, A.R. Hurson, and S.H. Pakzad. A taxonomy and current issues in multidatabase systems. *IEEE Computer*, 25(3), 1992.

[BKM+] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. http://www.almaden.ibm.com/cs/k53/www9.final.

[BM03] G. E. A. P. A. Batista and M. C. Monard. An analysis of four missing data treatment methods for supervised learning. *Applied Artificial Intelligence*, 17(5-6), 2003.

[BM05] G. Bianconi and M. Marsili. Loops of any size and hamilton cycles in random scale-free networks. In *cond-mat/0502552 v2*, 2005.

[BOT03] Y. Shu B.C. Ooi and K.-L. Tan. Relational data sharing in peer-based data management systems. *SIGMOD Record*, 32(3), 2003.

[Bou04] P. Bouquet et al. Specification of a common framework for characterizing alignment. KnowledgeWeb Deliverable 2.2.1, 2004. http://knowledgeweb.semanticweb.org.

[BPSM+04] T. Bray, J. Paoli, C.M. Sperberg-McQueen, E. Maler, and F. Yergeau (Ed.). Extensible Markup Language (XML) 1.0. W3C Recommendation, February 2004. http://www.w3.org/TR/REC-xml/.

[BY97] Y. Bar-Yam. *Dynamics of Complex Systems*. Perseus Books Group, 1997.

[CCRM02] G. Caldarelli, A. Capocci, P. De Los Rios, and M.A. Muoz. Scale-free networks from varying vertex intrinsic fitness. *Phys. Rev. Lett.*, 89, 258702, 2002.

[CF04a] M. Cai and M. Frank. RDFPeers: A Scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network. In *International World Wide Web Conference (WWW)*, 2004.

[CF04b] M. Cai and M. Frank. RDFPeers: A Scalable Distributed RDF Repository based on A Structured Peer-to-Peer Network. In *International World Wide Web Conference (WWW)*, 2004.

[CFMR04] S. Castano, A. Ferrara, S. Montanelli, and G. Racca. Semantic Information Interoperability in Open Networked Systems. In *International Conference on Semantics of a Networked World (ICSNW)*, 2004.

[CGLR04] D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Logical Foundations of Peer-To-Peer Data Integration. In *Symposium on Principles of Database Systems (PODS)*, 2004.

[Cli01] Clip2. The Gnutella Protocol Specification v0.4 (Document Revision 1.2), June 2001. www9.limewire.com/developer//gnutella_protocol_0.4.pdf.

[CMA04] P. Cudré-Mauroux and K. Aberer. A Necessary Condition For Semantic Interoperability in the Large. In *Ontologies, DataBases, and Applications of Semantics for Large Scale Information Systems (ODBASE)*, 2004.

[CMAF06] P. Cudré-Mauroux, K. Aberer, and A. Feher. Probabilistic Message Passing in Peer Data Management Systems. In *International Conference on Data Engineering (ICDE)*, 2006.

[CNSW00] Duncan S. Callaway, M.E.J. Newmann, Steven H. Strogatz, and Duncan J. Watts. Network robustness and fragility: Percolation on random graphs. *Phys. Rev. Lett.*, 85, 54685471, 2000.

[CSWH00] I. Clarke, O. Sandberg, B. Wiley, and T.W. Hong. Freenet: A Distributed Anonymous Information Storage and Retrieval System. In *International Workshop on Design Issues in Anonymity and Unobservability*, 2000.

[DA06] Z. Despotovic and K. Aberer. P2p reputation management: Probabilistic estimation vs. social networks. *Computer Networks*, 50(4), 2006.

[DDH01] A. Doan, P. Domingos, and A.Y. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *International Conference on Management of Data (SIGMOD)*, 2001.

[DG97] O.M. Duschka and M.R. Genesereth. Query planning in infomaster. In *ACM Symposium on Applied Computing*, 1997.

[DH05] X. Dong and A. Y. Halevy. A platform for personal information management and integration. In *CIDR*, 2005.

[DHA03] A. Datta, M. Hauswirth, and K. Aberer. Updates in Highly Unreliable, Replicated Peer-to-Peer Systems. In *International Conference on Distributed Computing Systems (ICDCS)*, 2003.

[DHS+05] A. Datta, M. Hauswirth, R. Schmidt, R. John, and K. Aberer. Range queries in trie-structured overlays. In *International Conference on Peer-to-Peer Computing*, 2005.

[DLR77] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society*, 39(Series B), 1977.

[DR02] H.H. Do and E. Rahm. COMA - a system for flexible combination of schema matching approaches. In *International Conference on Very Large Data Bases (VLDB)*, 2002.

[DS04] N. N. Dalvi and D. Suciu. Efficient Query Evaluation on Probabilistic Databases. In *International Conference on Very Large Data Bases (VLDB)*, 2004.

[EHS+03] M. Ehrig, P. Haase, R. Siebes, S. Staab, H. Stuckenschmidt, R. Studer, and C. Tempich. The SWAP Data and Metadata Model for Semantics-Based Peer-to-Peer Systems. In *Multiagent System Technologies (MATES)*, 2003.

[ES04] M. Ehrig and S. Staab. Qom quick ontology mapping. In *International Semantic Web Conference (ISWC)*, 2004.

[Euz04a] J. Euzenat. An api for ontology alignment. In *International Semantic Web Conference (ISWC)*, 2004.

[Euz04b] J. Euzenat et al. State of the art on current alignment techniques. KnowledgeWeb Deliverable 2.2.3, 2004. http://knowledgeweb.semanticweb.org.

[FHG+06] C. Fluit, B. Horak, G. A. Grimnes, A. Dengel, D. Nadeem, L. Sauermann, D. Heim, and M. Kiesel. Semantic desktop 2.0: The gnowsis experience. In *International Semantic Web Conference (ISWC)*, 2006.

[FKMP05] R. Fagin, P.G. Kolaitis, R.J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1), 2005.

[FKP04] A. Farhangfar, L. A. Kurgan, and W. Pedrycz. Experimental analysis of methods for imputation of missing values in databases. *Intelligent Computing: Theory and Applications II*, 5421(1), 2004.

[FLM99] M. Friedman, A. Levy, and T. Millstein. Navigational plans for data integration. In *National Conference on Artificial Intelligence (AAAI)*, 1999.

[FTS00] M. Fernandez, W.-C. Tan, and D. Suciu. Silkroute: Trading between relations and xml. In *International World Wide Web Conference (WWW)*, 2000.

[FW04] D.C. Fallside and P. Walmsley (Ed.). XML Schema Part 0: Primer. W3C Recommendation, October 2004. http://www.w3.org/TR/xmlschema-0/.

[Gal06] A. Gal. Managing Uncertainty in Schema Matching with Top-K Schema Mappings. *Journal on Data Semantics*, VI, 2006.

[GKD97] M. Genesereth, A.M. Keller, and O.M. Duschka. Infomaster: an information integration system. In *ACM SIGMOD International Conference on Management of Data*, 1997.

[GMJ04] A. Gal, G.A. Modica, and H.M. Jamil. OntoBuilder: Fully Automatic Extraction and Consolidation of Ontologies from Web Sources. In *International Conference on Data Engineering (ICDE)*, 2004.

[GMPQ+97] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, and J. Widom. The tsimmis project: Integration of heterogeneous inormation sources. *Journal of Intelligent Inf. Systems*, 8(2), 1997.

[Gru93] T.R. Gruber. A Translation Approach to Portable Ontology Specifications. *Knowledge Acquisition*, 6(2):199–221, 1993.

[Gua98] N. Guarino. Formal ontologies and information systems. In Nicola Guarino, editor, *Proceedings of FOIS '98*, pages 3 – 15. IOS Press, 1998.

[HA87] M.R. Horton and R. Adams. Standard for interchange of USENET messages. Network Information Center RFC 1036, December 1987.

[Hal01] A. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4), 2001.

[HCH+05] R. Huebsch, B. Chun, J. M. Hellerstein, B. T. Loo, P. Maniatis, T. Roscoe, S. Shenker, I. Stoica, and A. R. Yumerefendi. The Architecture of PIER: an Internet-Scale Query Processor. In *In Conference on Innovative Data Systems Research (CIDR)*, 2005.

[Hel03] J. M. Hellerstein. Toward network data indepence. *ACM SIGMOD Record*, 32(3), 2003.

[Hel04] J. Hellerstein. Architectures and Algorithms for Internet-Scale (P2P) Data Management. In *International Conference on Very Large Data Bases (VLDB)*, 2004.

[HFM06] A.Y. Halevy, M.J. Franklin, and D. Maier. Principles of dataspace systems. In *Principles of Database Systems (PODS)*, 2006.

[HHL+03] R. Huebsch, J. M. Hellerstein, N. Lanham, B. T. Loo, S. Shenker, and I. Stoica. Querying the Internet with PIER. In *Conference On Very Large Data Bases (VLDB)*, 2003.

[HIMT03] A.Y. Halevy, Z.G. Ives, P. Mork, and I. Tatarinov. Piazza: Data Management Infrastructure for Semantic Web Applications. In *International World Wide Web Conference (WWW)*, 2003.

[HIST03] A.Y. Halevy, Z.G. Ives, D. Suciu, and I. Tatarinov. Schema Mediation in Peer Data Management Systems. In *International Conference on Data Engineering (ICDE)*, 2003.

[HIST05] A.Y. Halevy, Z.G. Ives, D. Suciu, and I. Tatarinov. Schema mediation for large-scale semantic data sharing. *VLDB Journal*, 14(1), 2005.

[HMN+99] L.M. Haas, R.J. Miller, B. Niswonger, M.T. Roth, P.M. Schwarz, and E.L. Wimmers. Transforming heterogeneous data with database middleware: Beyond integration. *IEEE Data Eng. Bull.*, 22(1), 1999.

[JCH06] G. Jiang, G. Cybenko, and J.A. Hendler. Semantic Interoperability and Information Fluidity. *Int. J. Cooperative Inf. Syst.*, 15(1), 2006.

[KA04] A. Kementsietsidis and M. Arenas. Data Sharing through Query Translation in Autonomous Systems. In *International Conference on Very Large Data Bases (VLDB)*, 2004.

[KBH+05] D. R. Karger, K. Bakshi, D. Huynh, D. Quan, and V. Sinha. Haystack: A general-purpose information management tool for end users based on semistructured data. In *Biennial Conference on Innovative Data Systems Research (CIDR)*, 2005.

[KC04] G. Kokkinidis and V. Christophides. Semantic Query Routing and Processing in P2P Database Systems: The ICS-FORTH SQPeer Middleware. In *EDBT Workshops*, 2004.

[KFL01] F.R. Kschischang, B.J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47(2), 2001.

[LAZ+89] W. Litwin, A. Abdellatif, A. Zeroual, B. Nicolas, and Ph. Vigier. MSQL: a multidatabase language. *Source Information Sciences: an International Journal archive*, 49(1), 1989.

[LCC+02] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In *International Conference on Supercomputing*, 2002.

[Len02] M. Lenzerini. Data integration: A theoretical perspective. In *Principles of Database Systems (PODS)*, 2002.

[LIK06] E. Liarou, S. Idreos, and M. Koubarakis. Evaluating Conjunctive Triple Pattern Queries over Large Structured Overlay Networks. In *International Semantic Web Conference (ISWC)*, 2006.

[LKR06] J. Liang, R. Kumar, and K.W. Ross. The FastTrack overlay: A measurement study. *Computer Networks*, 50(6), 2006.

[LRO96] A.Y. Levy, A. Rajaraman, and J. O. Ordille. Query-answering algorithms for information agents. In *National Conference on Artificial Intelligence (AAAI)*, 1996.

[LST05] A. Loeser, S. Staab, and C. Tempich. Semantic Methods for P2P Query Routing. In *German Conference on Multi Agent Technologies (MATES)*, 2005.

[MH03a] J. Madhavan and A.Y. Halevy. Composing Mappings Among Data Sources. In *International Conference on Very Large Data Bases (VLDB)*, 2003.

[MH03b] J. Madhavan and A.Y. Halevy. Composing Mappings Among Data Sources. In *SIGMOD International Conference on Management of Data*, 2003.

[MHH+01] R.J. Miller, M.A. Hernàndez, L.M. Haas, L.-L. Yan, C.T.H. Ho, R. Fagin, and L. Popa. The Clio project: Managing heterogeneity. *SIGMOD Record*, 30(1), 2001.

[MKSI00] E. Mena, V. Kashyap, A. P. Sheth, and A. Illarramendi. OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies. *Distributed and Parallel Databases*, 8(2), 2000.

[MM04] F. Manola and E. Miller (Ed.). RDF Primer. W3C Recommendation, February 2004. http://www.w3.org/TR/rdf-primer/.

[MME06] A. Malhotra, J. Melton, and N. Walsh (Ed.). XQuery 1.0 and XPath 2.0 Functions and Operators. W3C Proposed Recommendation, November 2006. http://www.w3.org/TR/xpath-functions/.

[Mor38] C.W. Morris. Foundations of the theory of signs. *International Encyclopedia of Unified Science*, 1(2), 1938.

[Mv04] D.L. McGuinness and F. van Harmelen (Ed.). OWL Web Ontology Language Overview. W3C Recommendation, February 2004. http://www.w3.org/TR/owl-features/.

[MWJ99] K. M. Murphy, Y. Weiss, and M. I Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Uncertainty in Artificial Intelligence (UAI)*, 1999.

[NaCQD+02] W. Nejdl, B. Wolf anc C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmér, and T. Risch. EDUTELLA: a P2P networking infrastructure based on

RDF. In *International World Wide Web Conference (WWW)*, 2002.

[NOT02] W.S. Ng, B.C. Ooi, and K.L. Tan. Bestpeer: A self-configurable peer-to-peer system. In *International Conference on Data Engineering (ICDE)*, 2002.

[NOT03] W. S. Ng, B. C. Ooi, and K. L. Tan. BestPeer: A self-configurable peer-to-peer system. In *International Conference on Data Engineering (ICDE)*, 2003.

[NSW01] M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Random graphs with arbitrary degree distributions and their applications. *Phys. Rev.*, E 64(026118), 2001.

[NWS+03] W. Nejdl, M. Wolpers, W. Siberski, C. Schmitz, M.T. Schlosser, I. Brunkhorst, and A. Loeser. Super-peer-based routing and clustering strategies for RDF-based peer-to-peer networks. In *International World Wide Web Conference (WWW)*, 2003.

[OA99] A. M. Ouksel and I. Ahmed. Ontologies are not the panacea in data integration: A flexible coordinator for context construction. *Journal of Distributed and Parallel Databases*, 7(1), 1999.

[OLR96] J. Ordille, A. Levy, and A. Rajaraman. Querying heteregeneous information sources using source descriptions. In *International Conference on Very Large Data Bases (VLDB)*, 1996.

[OR23] C.K. Ogden and I.A. Richards. *The Meaning of Meaning: A Study of the Influence of Language upon Thought and of the Science of Symbolism*. Routledge & Kegan Paul Ltd., London, 10 edition, 1923.

[OST03] B. C. Ooi, Y. Shu, and K.-L. Tan. Relational Data Sharing in Peer-based Data Management Systems. *ACM SIGMOD Record*, 32(3), 2003.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems : Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[PG04a] M.A. Paskin and C.E. Guestrin. A robust architecture for distributed inference in sensor networks. In *Intel Research Technical Report IRB-TR-03-039*, 2004.

[PG04b] E. Prud'hommeaux and B. Grosof. "rdf query and rules: A framework and survey", 2004. http://www.w3.org/2001/11/13-RDF-Query-Rules/.

[PL00] R. Pottinger and A.Y. Levy. A scalable algorithm for answering queries using views. In *International Conference on Very Large Data Bases (VLDB)*, 2000.

[PRR97] C. G. Plaxton, R. Rajaraman, and A. W. Richa. Accessing Nearby Copies of Replicated Objects in a Distributed Environment. In *Annual ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 1997.

[PS06] E. Prud'hommeaux and A. Seaborne van Harmelen (Ed.). SPARQL Query Language for RDF. W3C Candidate Recommendation, April 2006. http://www.w3.org/TR/rdf-sparql-query/.

[Rap03] W.J. Rapaport. What Did You Mean by That? Misunderstanding, Negotiation, ans Syntactic Semantics. *Minds and Machines*, 13(3), 2003.

[RB01] E. Rahm and P. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4), 2001.

[RD01] A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems. In *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 2001.

[RFH+01] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A Scalable Content-Addressable Network. In *ACM SIGCOMM*, 2001.

[Rij79] C.J. Van Rijsbergen. *Information Retrieval*. Butterworths, 1979.

[RM04] J. Risson and T. Moors. Survey of Research towards Robust Peer-to-Peer Networks: Search Methods. Technical Report UNSW-EE-P2P-1-1, University of New South Wales, Sydney, Australia, Sep. 2004.

[Sea04] A. Seaborne. RDQL - A Query Language for RDF". W3C Member Submission, 2004. http://www.w3.org/Submission/RDQL/.

[SH06] L. Steels and P. Hanappe. Interoperability through Emergent Semantics. A Semiotic Dynamics Approach. *Journal on Data Semantics*, VI, 2006.

[SL90] A. P. Sheth and J. A. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3), 1990.

[SMK+01] I. Stoica, R. Morris, D. Karger, M.F. Kaashoek, and H. Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *ACM SIGCOMM*, 2001.

[Ste96] L. Steels. Self-organising vocabularies. In *Artificial Life V*, 1996.

[SVHB04] H. Stuckenschmidt, R. Vdovjak, G. J. Houben, and J. Broekstra. Index structures and algorithms for querying distributed rdf repositories. In *International World Wide Web Conference (WWW)*, 2004.

[TI06] N. E. Taylor and Z. G. Ives. Reconciling while tolerating disagreement in collaborative data sharing. In *SIGMOD Conference*, 2006.

[TIaAH+03] I. Tatarinov, Z. Ives, J. Madhavan amd A. Halevy, D. Suciu, N. Dalvi, X. Dong, Y. Kadiyaska, G. Miklau, and P. Mork. The Piazza Peer Data Management Project. *ACM SIGMOD Record*, 32(3), 2003.

[TSW04] C. Tempich, S. Staab, and A. Wranik. Remindin': Semantic query routing in peer-to-peer networks based on social metaphors. In *International World Wide Web Conference (WWW)*, 2004.

[vA06] L. von Ahn. Games with a purpose. *IEEE Computer Magazine*, 39(6), 2006.

[Wie92] G. Wiederhold. Mediators in the Architecture of Future Information Systems. *IEEE Computer*, 25(3), 1992.

[Wil94] H. S. Wilf. *Generatingfunctionology.* 2nd Edition, Academic Press, London, 1994.

[WS98] D. J. Watts and S. H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, 393:440–442, 1998.

[YFW00] J.S. Yedidia, W.T. Freeman, and Y Weiss. Generalized belief propagation. *Advances in Neural Information Processing Systems (NIPS)*, 13, 2000.

[YGM03] B. Yang and H. Garcia-Molina. Designing a Super-Peer Network. In *International Conference on Data Engineering (ICDE)*, 2003.

[ZLF+05] H. Zhuge, J. Liu, L. Feng, X. Sun, and C. He. Query routing in a peer-to-peer semantic link network. *Computational Intelligence*, 21(2), 2005.

[ZLFH04] H. Zhuge, J. Liu, L. Feng, and C. He. Semantic-Based Query Routing and Heterogeneous Data Integration in Peer-to-Peer Semantic Link Network. In *International Conference on Semantics of a Networked World (ICSNW)*, 2004.

CURRICULUM VITAE

Name:	**Philippe CUDRÉ-MAUROUX**
Date of birth:	September 13, 1976
Place of birth:	Fribourg, Switzerland
Languages:	French, English and German

EDUCATION

10.01 – 10.06 **Ph.D.** degree in Distributed Information Systems
Advisor: prof. Karl Aberer
EPFL, Lausanne – Switzerland.

07.05 – 10.05 Visiting Scholar (Database Group)
University of California at Berkeley – USA.

02.00 – 09.01 **M.Sc.** degree in Multimedia Communications (rank: 1^{st})
Eurecom Institute & EPFL, Sophia Antipolis – France.

in parallel with

M.Sc. degree in Network and Distributed Systems
INRIA SOP & University of Sophia Antipolis – France.

11.96 – 02.00 **B.Sc.** degree in Communication Systems
EPFL, Lausanne – Switzerland

09.92 – 07.96 Scientific Baccalauréat (rank: 1^{st})
Collège du Sud, Bulle – Switzerland.

09.89 – 07.92 Secondary School in Classics (Latin & Ancient Greek)
École Secondaire de la Gruyère, Bulle – Switzerland.

WORK EXPERIENCE

03.05 – 10.06 Lecturer
School of Computer and Communication Sciences
EPFL, Lausanne – Switzerland.

07.04 – 10.04 Visiting Researcher
Web Search & Mining Group
Microsoft Research Asia, Beijing – P.R. China

12.03 – 03.04 External Lecturer
United Nations IT Group
United Nations Headquarters, Geneva – Switzerland

10.01 – 03.05	Research & Teaching Assistant School of Computer and Communication Sciences EPFL, Lausanne – Switzerland.
03.01 – 09.01	Research Intern Media Delivery Architectures Group IBM T.J. Watson Research Center, Hawthorne, NY – USA.
03.99 – 02.00	Software Intern Hewlett-Packard Research Group EPFL, Lausanne – Switzerland.
07.96 – 10.96	Telecommunications Specialist Swiss Army Kloten – Switzerland.

PUBLICATIONS

Book & Book Chapters

- Karl Aberer and Philippe Cudré-Mauroux (Eds): Journal on Data Semantics VI, Springer Verlag 2006.
- Philippe Cudré-Mauroux and Karl Aberer: Belief Propagation on Uncertain Schema Mappings in Peer Data Management Systems. *Global Data Management*, IOS Press, 2006.
- Karl Aberer, Philippe Cudré-Mauroux and Manfred Hauswirth: Semantic Gossiping: Fostering Semantic Interoperability in Peer Data Management Systems. *Semantic Web and Peer-to-Peer*, Springer Verlag, 2006.

Journal Papers

- Philippe Cudré-Mauroux, Karl Aberer, Alia I. Abdelmoty, Tiziana Catarci, Ernesto Damiani, Arantxa Illaramendi, Mustafa Jarrar, Robert Meersman, Erich J. Neuhold, Christine Parent, Kai-Uwe Sattler, Monica Scannapieco, Stefano Spaccapietra, Peter Spyns, and Guy De Tré: Viewpoints on Emergent Semantics. *Journal on Data Semantics VI*, 2006.
- Adriana Budura, Philippe Cudré-Mauroux and Karl Aberer: From Bioinformatic Web Portals to Semantically Integrated Data Grid Networks. *Future Generation Computer Systems*, 22, 2006.

- Karl Aberer, Philippe Cudré-Mauroux and Manfred Hauswirth: Start making sense: The Chatty Web approach for global semantic agreements. *Journal of Web Semantics*, 1 (1), December 2003.
- Karl Aberer, Philippe Cudré-Mauroux, Anwitaman Datta, Zoran Despotovic, Manfred Hauswirth, Magdalena Punceva and Roman Schmidt: P-Grid: A Self-organizing Structured P2P System. *SIGMOD Record*, 32(2), September 2003.
- Karl Aberer, Philippe Cudre-Mauroux, Anwitaman Datta, Zoran Despotovic, Manfred Hauswirth, Magdalena Punceva, Roman Schmidt and Jie Wu: Advanced Peer-to-Peer Networking: The P-Grid System and its Applications. *PIK (Praxis der Informationsverarbeitung und Kommunikation)*, 26(3), 2003.
- Karl Aberer, Philippe Cudré-Mauroux and Manfred Hauswirth: A Framework for Semantic Gossiping. *SIGMOD Record*, 31(4), 2002.

Conference Papers

- Philippe Cudré-Mauroux, Menasheh Fogel, Ken Goldberg and Michael J. Franklin: "Sentry Pallets" for Automated Monitoring of Spatial Constraints. *International Conference on Robotics and Automation* (ICRA), 2006.
- Philippe Cudré-Mauroux, Karl Aberer and Andras Feher: Probabilistic Message Passing in Peer Data Management Systems. *International Conference on Data Engineering* (ICDE), 2006.
- Karl Aberer, Philippe Cudré-Mauroux and Zoran Despotovic: On the Convergence of Structured Search, Information Retrieval and Trust Management in Distributed Systems. *German Conference on Multiagent Systems* (MATES), 2005.
- Philippe Cudré-Mauroux and Karl Aberer: A Necessary Condition For Semantic Interoperability In The Large. *International Conference on Ontologies, DataBases, and Applications of Semantics* (ODBASE), 2004.
- Karl Aberer, Philippe Cudré-Mauroux, Manfred Hauswirth and Tim van Pelt: GridVine: Building Internet-Scale Semantic Overlay Networks. *International Semantic Web Conference* (ISWC), 2004.
- Karl Aberer, Tiziana Catarci, Philippe Cudré-Mauroux, Tharam Dillon, Stephan Grimm, Mohand-Said Hacid, Arantza Illarramendi, Mustafa Jarrar, Vipul Kashyap, Massimo Mecella, Eduardo Mena,

Erich J. Neuhold, Aris M. Ouksel, Thomas Risse, Monica Scannapieco, Fèlix Saltor, Luca de Santis, Stefano Spaccapietra, Steffen Staab, Rudi Studer and Olga De Troyer: Emergent Semantics Systems. *International Conference on Semantics of a Networked World* (ICSNW), 2004.

- Karl Aberer, Philippe Cudré-Mauroux, Aris M. Ouksel, Tiziana Catarci Mohand-Said Hacid, Arantza Illarramendi, Vipul Kashyap, Massimo Mecella, Eduardo Mena, Erich J. Neuhold, Olga De Troyer, Thomas Risse, Monica Scannapieco, Fèlix Saltor, Luca de Santis, Stefano Spaccapietra, Steffen Staab and Rudi Studer: Emergent Semantics Principles and Issues. *International Conference on Database Systems for Advanced Applications* (DASFAA), 2004.

- Karl Aberer, Philippe Cudré-Mauroux and Manfred Hauswirth: The Chatty Web: Emergent Semantics Through Gossiping. *International World Wide Web Conference* (WWW), 2003.

- Philippe Cudré-Mauroux and Karl Aberer: A Decentralized Architecture for Adaptive Media Dissemination. International Conference on Multimedia and Expo (ICME), 2002.

Workshop Papers

- Philippe Cudré-Mauroux, Julien Gaugaz, Adriana Budura and Karl Aberer: Analyzing Semantic Interoperability in Bioinformatic Database Networks. *Semantic Network Analysis Workshop* (SNA), 2005

- Karl Aberer, Philippe Cudré-Mauroux, Anwitaman Datta and Manfred Hauswirth: PIX-Grid: A Platform for P2P Photo Exchange. *Workshop on Ubiquitous Mobile Information and Collaboration Systems* (UMICS), 2003.

Tutorial

- Karl Aberer and Philippe Cudré-Mauroux: Semantic Overlay Netwoks. *International Conference on Very Large Data Bases* (VLDB), 2005.

TEACHING

Lecturer for the Introduction to Information Systems course, EPFL, B.Sc. program in Communication Systems, Spring 2005.

Lecturer for the Advanced XML & Web Services course, U.N. Headquarters in Geneva, Switzerland, 2003 - 2004.

Teaching Assistant for the Distributed Information Systems course, EPFL, M.Sc. program in Communication Systems, Fall 2003 & 2004.

Teaching Assistant for the Conception of Information Systems course, EPFL, M.Sc. program in Communication Systems, Spring 2002 & 2003.

Teaching Assistant for the Bases de Données course, EPFL, B.Sc. program in Computer Science, Fall 2001 & 2002.

PROFESSIONAL ACTIVITIES

Member of the IFIP Working Group 2.6. on Databases, of the IEEE and of the ACM.

Co-Guest Editor for the Journal on Data Semantics VI, special issue on Emergent Semantics, Springer Verlag 2006.

Program Committee Member for ISWC 2006, DISWeb 2006, MCISME 2006, P2P Data and Knowledge Sharing 2006, Semantics 2005, EEE 2005 and OTM 2004.

Reviewer for VLDB (2002, 2004, 2005, 2006), ICDE (2004, 2006), ISWC (2003, 2004, 2005), WWW (2004, 2005, 2006) and SIGMOD (2003, 2004).

Reviewer for SIGMOD Record, the Journal of Web Semantics, the Journal of Cooperative Information Systems and the Journal of Data Semantics.

AWARDS

- Outstanding EMEA Student, P&G, Geneva. Invited to the International Program in IT Business Management, Roma – Italy (2003).
- Best Student, M.Sc. program in Multimedia Communications, Eurecom Institute & EPFL, France & Switzerland (2001).
- Best Scientific Student, Collège Du Sud, Bulle – Switzerland (1996).