Rule-Based Social Networking for Expert Finding

by

Jie Li

Bachelor of Science, Bachelor of Arts,
Taiyuan University of Technology, China, 2004

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF

Master of Computer Science

In the Graduate Academic Unit of Computer Science

Supervisor(s):	Harold Boley, Ph.D., Computer Science
	Virendrakumar C. Bhavsar, Ph.D., Computer Science
Examining Board:	Przemyslaw Rafal Pochec, Ph.D., Computer Science, Chair
	Weichang Du, Ph.D., Computer Science
External Examiner:	Donglei Du, Ph.D., Business Administration

This thesis is accepted

Dean of Graduate Studies

THE UNIVERSITY OF NEW BRUNSWICK

September, 2006

Abstract

Web-based social networking has emerged as a major application area for the Internet. Some of the recently launched social networking portals use Semantic Web metadata vocabularies (classes and properties) to describe and connect people, groups, as well as organisations. However, existing portals are fact-based; there is a lack of rule-based reasoning in social networking. Rules added to social networking can make implicit properties explicit; rules can also constitute properties conditional on other agents, the time, the location and so on. Therefore, we put forward rule-based social networking for enhancing the current social networking. In this thesis, we have implemented a system that applies rule-based social networking to expert finding. The assumed business-service model is bartering-like exchange of expertise between an expert and a co-expert, where the latter initiates the search using our system. When searching for an expert, in any domain, we often need to rely on referrals by other experts using their social network. Our system thus provides both direct searches and referrals, which are both accomplished by applying rules to users' expert queries. We also propose a benchmark suite for expert finding, more generally, testing expert-finding systems against expert profiles. This is exemplified with our implemented system, testing it against the expertise and co-expertise domains of Computer Science and music, respectively.

Acknowledgements

First of all, I would like to greatly thank my supervisors, Dr. Harold Boley and Dr. Virendrakumar C. Bhavsar, for all the guidance they gave and the help they offered me. Thanks for their inspiration and encouragement during the completion of my MCS thesis.

I would also like to take this time to express my appreciation to Dr. Przemyslaw Rafal Pochec, Dr. Weichang Du and Dr. Donglei Du, for their time and patience they spend on reading my MCS thesis and their valuable suggestions for improving it.

Finally, let me express my heartfelt gratitude to my dear parents for their deep love and unconditional support. Special thanks also go to my friends, for the friendship and help they offered.

Table of Contents

List of Tables

List of Figures

Glossary

ACM	-	Association for Computing Machinery
RuleML	-	Rule Markup Language
OO RuleML	-	Object Oriented Rule Markup Language
FindXpRT	-	Find an eXpert via Rules and Taxonomies
FOAF	-	the Friend Of A Friend project
RDF	-	Resource Description Framework
RDFS	-	Resource Description Framework Schemas
POSL	-	Positional-Slotted Language
XSLT	-	The Extensible Stylesheet Language Transformations
XML	-	Extensible Markup Language
jDREW	-	java Deductive Reasoning Engine for the Web
OO jDREW	-	Object Oriented java Deductive Reasoning Engine for the Web
RNF	-	Rule-oriented Normal Form
FNF	-	Fact-oriented Normal Form
WWW	-	World Wide Web
URI	-	Universal Resource Identifier
HTTP	-	Hypertext Transfer Protocol
HTML	-	HyperText Markup Language
W3C	-	World Wide Web Consortium
SVG	-	Scalable Vector Graphics
DC	-	Dublin Core

SWRL	-	Semantic Web Rule Language
WRL	-	Web Rule Language
KR	-	Knowledge Representation
DAML	-	DARPA Agent Markup Language
TD	-	Top Down
BU	-	Bottom Up
Prolog	-	PROgramming in LOGic
OWL	-	Web Ontology Language
SeSDL	-	Scottish electronic Staff Development Library

Chapter 1

Introduction

RuleML FOAF, a web rule language for social networking, is discussed in this thesis and applied for social networking. In this chapter, we provide a high-level introduction of FOAF (Friend Of A Friend) and the need for RuleML FOAF.

In Section 1.1, we give an overview of the development of web-based social networking. Next we give the thesis objective, followed by the thesis organization in Section 1.2.

1.1 Overview

Web-based social networking is emerging as a major application area for Semantic Web metadata. "Social networking is built on the idea that there is a determinable structure to how people know each other, whether directly or indirectly" [34]. The well-known notion of "six degrees of separation" [34] means that a person can reach any other person with at most six intermediate personal relationships (because there is a connecting path containing seven nodes with five persons acting as mediators). This supports the idea that people, even directly not knowing each other, can be connected and thus develop a virtual community. This connectivity enables people to both keep in touch with friends and meet new friends. Weblogging, introducing the idea of social networking to today's Internet, is gaining more and more popularity. Webloggers can cite each other and even post testimonials about other

persons. In this way, people can get to know new friends through forums or groups that consist of people with special interests. Moreover, many communities have proliferated on the Internet, from companies through professional organizations to social groupings, so it is very important to find the same interest business partner.

Recently, a number of portals have become popular in this area, including FOAF [31], Friendster [3] and Stumbleupon [5]. In particular, the RDF-based Friend-Of-A-Friend (FOAF) project (http://foaf-project.org), originated by Dan Brickley and Libby Miller [33], has been set up to meet this increasing demand. FOAF allows expression of personal information and relationships, and permits machine-readable homepages for grouping, categorization and linking of persons. In this sense, FOAF supports online communities and more importantly, it has the potential to become an important tool in managing communities. In addition to providing simple directory services, one can use information from FOAF in many ways. For example [37]:

1. Augment e-mail filtering by prioritizing mail from trusted colleagues

2. Provide assistance to new entrants in a community

3. Locate people with interests similar to yours.

The FOAF vocabulary does not, however, capture rule knowledge, and Web Rule Languages such as the Rule Markup Language (RuleML) [27] have only recently been applied to social networking. For example, rules can be employed for dynamically deriving certain FOAF facts on demand. Using the Objected Oriented java Deductive Reasoning Engine for the Web (OO jDREW), one can find people with similar interests more quickly, and RuleML tools enable the XML-based elicitation (VDR-Device [24]), interchange (XSLT [61]), and execution (OO jDREW [22]) of such rules.

1.2 Thesis Objective and Methodology

The current RDF-based Friend-Of-A-Friend (FOAF) project (www.foaf-project.org), which is attracting increasing attention of researchers as well as practitioners, only provides person-centric facts.

One possible approach proposed in this thesis is combining RuleML (www.ruleml.org)and FOAF, which enriches FOAF facts with RuleML rules and hence leads to the enhancement of current FOAF project.

In this thesis, we focus on the expressiveness of FOAF on expert finding, consulting and collaboration in the computer science domain. In this sense, we extend the RDF FOAF vocabulary to the RuleML FOAF vocabulary specification, such as expert(?P, ?E) expressing an expert ?P with an expertise ?E.

Our research objective is to extend FOAF facts to RuleML rules for dynamically deriving certain FOAF facts.

In particular, our research methodology is as follows:

- Develop the current RDF FOAF vocabulary for both elementary and rule-derivable facts
- Enrich such descriptive facts by OO RuleML rules deriving new descriptions
- Develop a general RuleML FOAF vocabulary for rules
- Build rulebases for the FOAF community to exchange and share information
- Implement both FNF and RNF to benefit both the current FOAF community and the development of FOAF projects by extending them with more new features
- Apply RuleML FOAF to expert finding, leading to a system called FindXpRT (Find an eXpert via Rules and Taxonomies), which is benchmarked by music and Computer Science collaboration

1.3 Thesis Organization

The organization of the thesis is as follows. Social networking and other related basic concepts in this thesis are introduced in Chapter 2. In Chapter 3, we describe Rules and Rule engines.Then in Chapter 4, we propose an approach aiming at achieving the goal of this thesis: the design of a FOAF rule vocabulary, which is the principal component in the realization. Two normal forms, the FNF and RNF, are provided for FindXpRT users with different preferences. Finally, an XSLT translation from an FNF subset of RuleML to RDF is defined, producing RDF syntax as used by the FOAF community. Chapter 5 details the use case focused in this thesis: applying RuleML FOAF to expert finding in the domain of computer science and music preparing a possible client-expert collaboration. Sample facts and rules are provided in the human-oriented POSL of RuleML. Experiment results and evaluation are given in Chapter 6. Finally, in Chapter 7, we conclude the thesis with an overview of contributions and future work.

Chapter 2

Social Networking

The following sections in this chapter describe the concepts that are relevant to social networking. In Section 2.1, we give a brief introduction to the Semantic Web, followed by the social web in Section 2.2. We then explain what expert finding is in Section 2.3. The Friend Of A Friend (FOAF) project is described in Section 2.4 including its current development, its vocabulary and its limitations as well. Finally, in Section 2.5 few concluding remarks are given.

2.1 The Semantic Web

The Semantic Web is put forward by Tim Berners-Lee, who also invented the WWW, URIs, HTTP as well as HTML [59].

Simply put, the notion of the Semantic Web, according to Tim Berners-Lee himself, is defined as (www.w3.org/2001/sw/EO/points):

> "The Semantic Web is an extension of the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation."

Similarly, W3C's definition to the Semantic Web is (www.w3.org/2001/sw):

> "The Semantic Web provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries. It is a collaborative effort led by W3C with participation from a large number of researchers and industrial partners. It is based on the Resource Description Framework (RDF), which integrates a variety of applications using XML for syntax and URIs for naming."

The Semantic Web focuses not only on display purposes, but also the expressions of machine readable metadata.

Why the Semantic Web? In observation that the current web encounters the difficulty in managing, searching and processing the information, the Semantic Web emerges to address these problems by means of building a "manageable" web via adding explicit meaning to the information distributed within the web [51]. For these reasons, the Semantic Web is striving for information integration from heterogeneous sources, interchange and reuse of distributed information in a shared and uniform format, and the automation in managing, searching and processing the information on the web.

To be more intuitive, the architecture of the Semantic Web is shown in Figure 2.1 [25], where in this thesis we will discuss from the second layer from the bottom up to the fifth:

Currently, the World Wide Web consortium (W3C) is making efforts to realize the vision of the Semantic Web by developing the Semantic Web techniques and adapting them to the Semantic Web applications.

2.1.1 Metadata and Resource Description Framework (RDF)

In this subsection, we introduce two important notions in the Semantic Web: metadata and Resource Description Framework (RDF).

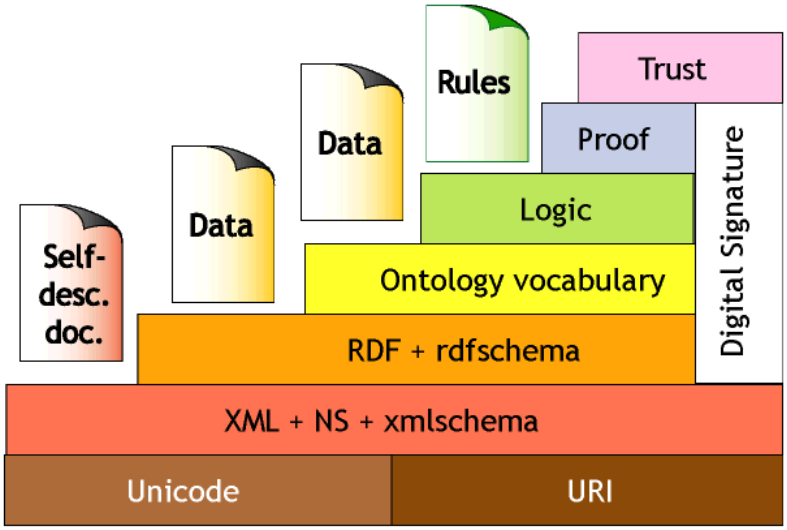

Figure 2.1: Semantic Web Architecture [25]

2.1.1.1 Metadata

Simply put, metadata is data about data, or information about information. Metadata is defines as "structured information that describes, explains, locates, or otherwise makes it easier to retrieve, use, or manage an information resource." in [58]. The World Wide Web Consortium (W3C), more concisely, explains metadata as machine readable information on the web.

Why do we need metadata? Metadata can benefit, not only resource discovery, but also the organization of electronic resources, interoperability, digital identification and archiving and preservation (http://en.wikipedia.org/wiki/Metadata). Among these, interoperability and exchange of metadata, to some extent, drawing upon the effort by RDF introduced in the next subsection, are where our focus lies in in this thesis.

2.1.1.2 Resource Description Framework (RDF)

The Resource Description Framework (RDF), recommended by W3C, is a framework for modeling information at a meta data level of the Web [62]. "RDF is designed for widespread, decentralised use." [32]. Aiming at the understandability by computers, RDF enables using

and interchange of information.

RDF is written in XML syntax and is highly involved in the W3C's Semantic Web Activity. RDF uses a triple expression, consisting subject, predicate and object. The subject represents the resource; the predicate expresses the relationship between the subject and the object, while the object is the object of this relationship.

RDF Schema (RDFS) "RDF Schema is a language for describing vocabularies in RDF." (http://en.wikipedia.org/wiki/RDF_Schema). Being a semantic extension of RDF, RDFS has mechanisms that can describe the RDF properties, such as attributes of resources and relationships between them.

To sum up, RDF provides a mechanism with the purpose of integrating multiple metadata schemas extracting from distributed information [58].

2.2 The Social Web

Unlike the World Wide Web, which shares distributed data by linking documents together, the social web achieves this by linking agents, e.g., people and organizations.

To support social networking it is helpful to represent various properties of, and relationships between, persons expressing a wide range of self-description and social connectivity. Persons who participate in such a networked (sub)society may be friends, relatives, work collaborators, employees, and so on. A diagram that illustrates a schematic example of a social network with four kinds of links is depicted in Figure 2.2.

2.3 Expertise Finding

In the era of rapidly developing technology and economics, there emerge many multi-regional corporations. However, these large organizations often suffer from the fact that people in the same organization do not know each other well or what other members' expertise is. Effective collaboration among employees is thus hampered.

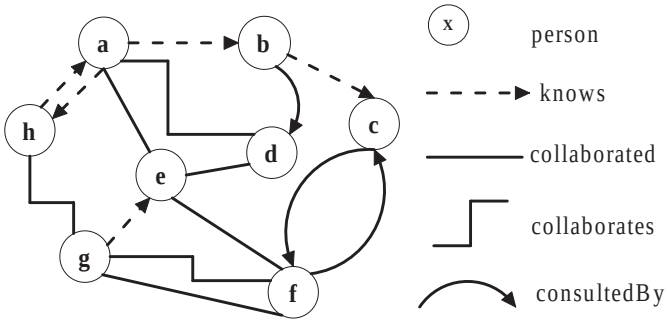

Figure 2.2: A social network with 'knows', 'collaborates', 'collaborated' and 'seek advice' links

Portals such as ExpertWitness [11], Expertise Search [10] and Teclantic [17], providing services for finding an expert or an expertise-developing project, have become popular recently, as they provide an appropriate way to help solving the above problem. These portals enable users to query for a specific expertise and provide them with a list of experts, who have the relevant expertise, through match-making.

However, it is quite common that novice end users have difficulty in characterizing their request of specific expertise, and current systems are not user-friendly enough to help users to find an expert and collaborate with him or her if desired.

Therefore, a major application of our model is expert finding, where a taxonomy of expertise is needed. The technology taxonomy from Teclantic and its classification data are the main sources for the expertise taxonomy utilized in this thesis. The expertise taxonomy can be implemented in RDFS [62], providing the order-sorted type system for OO jDREW [22]. With the help of the expertise taxonomy and rule specification, an expert finder can help people inside or outside a company to find an appropriate expert with whom they can collaborate and even provide 'proxy' suggestions, thus supporting the collaboration between people.

2.4 Friend Of A Friend (FOAF) Project

In this section, we first provide an introduction to RDFWeb, and then an overview of FOAF. Next, we describe the FOAF vocabulary, followed by the limitation of RDF-based

FOAF.

2.4.1 Introduction to RDFWeb

Nowadays, information tends to be distributed on the internet, while these information can be linked together through various connections. The Semantic Web considers the Web as a distributed database holding arbitrary metadata. The idea of RDFWeb emerges so as to fulfill this goal. RDFWeb focuses on developing techniques to show people the connections between things that they are interested in [32].

RDFWeb has become a popular means for people to describe themselves in ways which machines can understand. RDFWeb deals with a web of inter-related homepages of people using W3C's RDF technology to integrate information from different homepages and connect them together where there is some relationships among them [32].

Therefore, RDFWeb strives to fulfill the following goals [32]:

- Keep track of the distributed metadata among the Web in an effective way
- Find useful metadata on the Web by identifying their properties and inter-relationships
- Among common infrastructure, useful metadata are enabled to be shared
- A system served as database for Web searching
- Metadata are distributed, decentralised, and content-neutral

With semantics presented by RDFWeb, people are able to get desired result through query. Moreover, more effective searching is also enabled by RDFWeb tools which "watch the Web for new files, and combine your data into a database along with many other similar files" [32].

2.4.2 Overview of FOAF

FOAF, standing for Friend Of A Friend project, is originated by Dan Brickley and Libby Miller. FOAF emerged in 1998, as an RDF description of Dan Brickley in his homepage.

"FOAF is an open community-lead initiative which is tackling head-on the wider Semantic Web goal of creating a machine processable web of data. Achieving this goal quickly requires a network-effect that will rapidly yield a mass of data." Developed as its original idea, FOAF enables the Semantic Web idea to be applied into personal homepage, linking together FOAF profiles of different persons that present data with well-defined semantics (www.xml.com/pub/a/2004/02/04/foaf.html).

During recent years, FOAF, has been not only attracting more and more industry attention, but research interests. Many efforts have been made, and many technologies have been applied within this domain. For example, foaf-a-matic (www.ldodds.com/foaf/foaf-a-matic), is a handy tool invented by Leigh Dodd which enables an easy way for people to create their own FOAF description; FOAFNaut, online available at http://foafnaut.org, is an SVG-based navigator which provides people with a visualization of his/her social network. FOAF Explorer, online available at http://xml.mfd-consult.dk/foaf/explorer, is a server-based navigator using an XSLT bookmarklet that facilitates people exploring a FOAF neighborhood by presenting their profiles in a human-readable syntax.

The RDF-based FOAF is all about agent-centric (person and organization) web that describes agents' information. FOAF permits detailed description of profiles of persons and the relationships between them using a machine-readable syntax, i.e., a person's name, gender and other persons that he/she knows. Each FOAF user owns his/her own profile and what FOAF gives him/her are some terminology and structure so that his/her profile can be easily shared.

2.4.3 FOAF Vocabulary

FOAF, an application to RDFWeb, provides the basic vocabulary for RDFWeb by defining useful RDF properties. FOAF provides the basic RDF vocabulary we invented for itself, plus other useful existing vocabulary such as the Dublin Core metadata elements. (Dublin Core, abbreviated to DC, provides the metadata for "common semantic building block", while RDF is an infrastructure for supporting the integration of DC [67].)

FOAF is originally realized as a Semantic Web vocabulary serialized in RDF/XML, from which browsable HTML can be generated. Indeed, FOAF is mostly an RDF vocabulary, as is expressed as its significant contribution. Being in an early stage, the FOAF vocabulary specification is not yet a standard in the sense of ISO Standardization, or that associated with W3C Process [33].

It is impossible for FOAF vocabulary to capture everything that would satisfy people with different purposes. Instead of developing FOAF vocabulary into a full dictionary, a better solution can be reached by means of RDF, with which we can take advantage of other useful vocabulary specification. Therefore, presented in the view of Ontology, FOAF permits simultaneous extension for its author with particular intentions [33].

FOAF vocabulary is specified via the namespace URI 'http://xmlns.com/foaf/0.1/', and consists of two components: vocabulary about classes and vocabulary about properties. Vocabulary about classes is designed to express the type of an object (e.g., foaf:Person), while vocabulary about properties is used to express the type of a relationship or an attribute (e.g., foaf:knows and foaf:name respectively).

2.4.4 Limitation of RDF-Based FOAF

Admittedly, in spite of the impressive FOAF vocabulary specification, RDF-based FOAF also has its limitations.

The current FOAF project contains only fact vocabulary since RDF cannot express rules. Therefore rule-based deduction is not usually done in FOAF. However, rules have significant use, which can be brought to bear to FOAF rules. For example, rules can be used to derive FOAF properties and relationships from existing ones; profile 'facts' can be made conditional on the situation, context, time and/or location, like the time preference of and distance from an access.

Therefore, building on top of the current FOAF project, in this thesis, we propose a solution, namely RuleML FOAF, combining RuleML (Rule Markup Language) [27] with FOAF by studying the properties and use of RuleML rules for FOAF homepages and thus

enhancing the RDF-based FOAF by applying rules where needed.

Chapter 3

Rule Languages and Tools

The following sections in this chapter describe the main concepts, namely rule languages and tools, that are relevant to this thesis. In Section 3.1, we give a brief introduction to what a (Horn) rule is. Rule languages are introduced in Section 3.2, followed by rule engines in Section 3.3. Finally, in Section 3.4 we conclude this chapter.

3.1 (Horn) Rules

In logic, a rule is syntactically expressed by a relation consists of a set of formulas, the premises and an assertion (the conclusion) (http://en.wikipedia.org/wiki/Inference_rule). In this thesis, we focus only on rules in the domain of Horn logic.

A Horn clause (http://mathworld.wolfram.com/HornClause.html) is defined as a clause that contains zero or one positive literal. A Horn rule, therefore, is a rule with at most one conclusion and zero or more premises.

3.2 Rule Languages

Rule languages have evolved quickly over the last few years as there is an increasing demand in the Semantic Web, e.g., Semantic Web Rule Language (SWRL), Web Rule Language (WRL) and Rule Markup Language (RuleML).

In this section, we only give an overview of the declarative programming languages relevant to this thesis, including rule languages such as RuleML and POSL [28].

3.2.1 Rule Markup Language (RuleML)

Over years, rules, especially transformation rules and inference rules, have shown their impact on E-Commerce as well as the Semantic Web; moreover, rule interchange is playing a significant role in Knowledge Representation (KR) [26].

Rule Markup Language (RuleML), designed by the RuleML Markup Initiative, is intended to meet these needs. It is built on other standards work, i.e., Mathematical Markup Language (MathML), DARPA Agent Markup Language (DAML), Predictive Model Markup Language (PMML), Attribute Grammars in XML (AG-markup), and Extensible Stylesheet Language Transformations (XSLT). RuleML is an XML-based markup language that allows one to publish and share rulebases on the World Wide Web. The goal of RuleML, quoted as follows, is put forward in its technology report[1].

> "Our main goal is to provide a basis for an integrated rule-markup approach that will be beneficial to all involved and to the rule community at large ... This RuleML kernel language can serve as a specification for immediate rule interchange and can be gradually extended ... "

RuleML is an evolving logic language with a current version RuleML 0.9. It permits both forward (bottom-up) and backward (top-down) rules for deduction and even other usages such as rewriting, and further inferential-transformational tasks. RuleML sustains a family of sublanguages, such as FOL RuleML, SCLP RuleML and Fuzzy RuleML (http://en.wikipedia.org/wiki/RuleML).

In our thesis, however, we focus only on Object-Oriented RuleML (OO RuleML), online available at www.ruleml.org/indoo/indoo.html, an extension of RuleML that has developed from a "slotted" sublanguage of RuleML.

3.2.2 Positional-Slotted Language (POSL)

Prolog is the acronym of PROgramming in LOGic. It is based on the mathematical notions of relations and logical inference and therefore a logic programming language using Horn clauses (http://cs.wwc.edu/KU/PR/Prolog.html).

Prolog is a declarative language that holds the fact as well as rules, and rather than executing the program, it allows the user to issue a query and the system searches through the facts in order to provide the user with an answer.

POSL (Positional-Slotted Language), derived from Prolog, is a human-readable format for Semantic Web Knowledge that combines Prolog's positional and F-logic's (standing for Frame Logic, which is the logic basis of frame-based and object-oriented languages for data and knowledge representation [50]) slotted syntaxes for representing knowledge (facts and rules) in the Semantic Web [28]. Harold Boley in [28] describes that POSL not only accommodates many kinds of assertional-logical and object-centered modeling styles, but also achieves conciseness and orthogonality at large.

With the compactness that POSL has gained, it is easier for humans to write and read in POSL than in XML syntax. Moreover, since it is interconvertible with RuleML, the advantage of XML, being more machine-readable, is thus preserved.

3.3 Tools for Rules

In this section, we describe rule engines such as OO jDREW, as well as translating environments, such as XSLT transformation, both of which serve as tools for RuleML.

3.3.1 Rule Engines

Mandarax [4], SweetRules [6] and jDREW [64] have been developed as rule engines supporting various subset of RuleML.

OO jDREW, standing for Object Oriented java Deductive Reasoning Engine for the Web, is an object oriented extension of jDREW (java Deductive Reasoning Engine for the

Web). OO jDREW is a Java reasoning engine for executing RuleML rule markup and the Object Oriented extension for RuleML as well.

There are two kinds of deduction in OO jDREW: OO jDREW BU (Bottom Up) and OO jDREW TD (Top Down). Bottom up reasoning, in the sense of forward reasoning, resolves a fact against a premise of a rule. The visualization of bottom up reasoning shows the input facts at the bottom of the tree while the derived ones are seen at the top. In contrast, top down reasoning, namely backward reasoning, is used to resolve a problem from the more general to the more specific and finally reaches the exact query of the reasoning.

OO jDREW BU: In OO jDREW Bottom Up, rules are used to derive new facts from given facts until a fix point is reached. Three features are designed in OO jDREW BU: Type Definition, realized in RDFS, which can load the type hierarchy for the type system the user uses; Knowledge Base, permitting both RuleML and POSL syntax, stores the facts and rules useful for Bottom up deduction; Running the Forward Reasoner, outputting either RuleML or POSL syntax, provides the users with all facts and rules after executing the forward reasoning.

OO jDREW TD: In OO jDREW Top Down, rules are used to answer queries by reducing them to subqueries until facts are reached. Three features are designed in OO jDREW BU: Type Definition, which is same as in OO jDREW BU; Knowledge Base, instead, stores the fact and rules for query on demand. Querying the Knowledge Base, permitting only POSL syntax, is the Top Down application, also known as backward reasoning.

3.3.2 XSLT Transformation

XSL stands for EXtensible Stylesheet Language. Being an XML-based Stylesheet Language, XSL is regarded as the XML Style Sheets that describes how the XML should be displayed (www.w3schools.com/xsl/xsl_languages.asp).

XSL consists of three parts: XSLT (a language for transforming XML documents),

XPath (a language for navigating in XML documents) and XSL-FO (a language for formatting XML documents).

XSLT, namely XSL Transformations, is the crucial part of XSL and is served as a W3C Recommendation. It, drawing upon XPath, can help transform one XML file to another XML file, or even another type of XML file which can be recognized by a browser (www.w3schools.com/xsl/xsl_intro.asp). XSLT facilitates people to make use of the input document and obtain the output file with desired style he/she wants. For example, one can add/remove, or even rearrange and sort the elements, perform tests and choose whether to display or hide those elements.

XSLT has been developed with a current version XSLT 2.0, released in November, 2005, serving as the revised and enhanced version of the previous version XSLT 1.0.

In this thesis, XSLT is used, edited and executed via XML Spy, not only to map RuleML facts back to RDF facts, but also transform an XML document to RDFS, which can be later used as type definition loaded into OO jDREW.

Chapter 4

RuleML FOAF: Fact and Rule Profiles for Agents

RuleML FOAF (www.ruleml.org/usecases/foaf), combining RDF-based FOAF with RuleML, can extend the factual FOAF vocabulary by RuleML rules. Rules enriched for deriving new facts can infer relations, for example, reflexive, symmetric, and transitive. With RuleML FOAF, users can derive useful information by employing (person-centric) rules, either before RDF FOAF publication or, on demand, from published RuleML FOAF pages.

The following sections are organized as follows: in Section 4.1 we give an overview of what RuleML FOAF is and what it can do for social networking; the characteristics of FOAF rules are introduced in Section 4.2; in Section 4.3 we describe how rules can extend FOAF profiles for social networking; taxonomies in RDFS are depicted in Section 4.4, followed by the classification of facts in Section 4.5; we then explain how rules can enhance FOAF in details in Section 4.6; RuleML FOAF vocabulary specification is focused in Section 4.7; two normal forms are given later in Section 4.8 and in Section 4.9 we draw concluding remarks of this chapter.

4.1 Overview

Most social networking sites are based on a centralized architecture: all user descriptions are stored in one database. There is, however, growing user and business interest in portability between such sites, and for 'single sign-on' mechanisms that reduce the need for data re-entry, while allowing users to publish different aspects of themselves in different contexts. The Friend-Of-A-Friend (FOAF) vocabulary allows such sites to address the user demand for control of 'their' data [47].

Lorne H. Bouchard in [29] introduces that the web can be described from different phases: syntax, semantics and pragmatics. Recall the Semantic Web architecture shown in Figure 1.1 in Chapter 1. Syntax, describing the structure, corresponds to XML/S, which is the bottom layer. Semantics, namely meaning of an object, correspond to the second and third layers, where RDF/S and OWL lay. Finally, pragmatics, representing the upper three layers, focuses on the "intended effect of the utterance" [29], and is where SWRL, explanations, proofs are involved in.

The RDF-based FOAF concentrates solely on syntax and semantics. Pragmatics (rules), have not yet been applied to FOAF. Figure 4.1 illustrates Tim Bernners-Lee's vision on the Semantic Web Bus [25]. As shown in the diagram, rules applying to facts (ontologies), when executed in heuristic engines (e.g. rule engine) can enrich the database by inferring new facts. Developing rules can hence also enable more advanced query (search).

Moreover, we need to give each person a unique identification so that FOAF can represent them unambiguously. There are examples using email address, homepages and so on. Note that the means to represent people need not be unique, that is, it can be people's email addresses, homepages or cellphone numbers, RDFWeb software can integrate and normalize this data. In our case, we use each person's FOAF page which contains the profiles describing their FOAF owners.

Therefore, a Web Rule Language is now seen as the next research target for the Semantic Web. While metadata were traditionally handcrafted and stored statically, rules can be

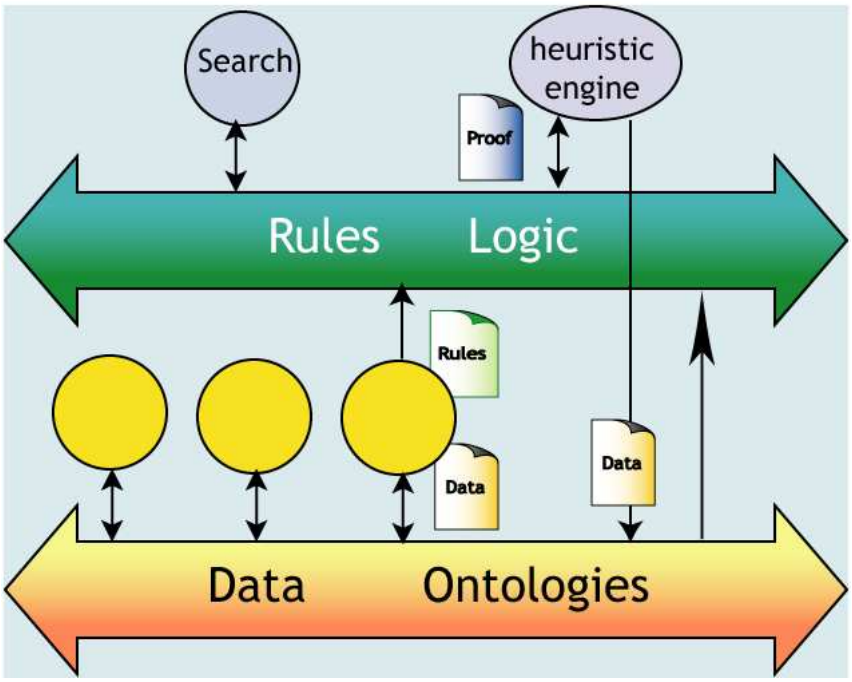

Figure 4.1: Semantic Web Bus [25]

employed for dynamically deriving required metadata on demand. The Rule Markup Language (RuleML) enables the XML-based elicitation, interchange and execution of rules. The FOAF vocabulary does not currently capture rule knowledge, and the RuleML specification has not before been applied to social networking. The RuleML FOAF research combines both strands by studying the mathematical properties and use of RuleML rules for FOAF homepages [47].

4.2 Characteristics of FOAF Rules

In this section we describe the characteristics of FOAF rules, including *Symmetric* relations, *'Harvesting'* property of rules, as well as *Individualized* and *Non-Individualized*, *Local* and *Global* properties of rules.

4.2.1 Symmetric Relations

In binary relations, a relation between the two elements can either be *Symmetric* or *Asymmetric*. A relation R over a set S is *Symmetric*, as expressed in Equation 4.1, if for any x, y in S, we have xRy if and only if yRx:

$$\forall x, y \in S, xRy \Rightarrow xRy \tag{4.1}$$

In particular, a FOAF rule is *Symmetric* ***iff*** it satisfies the condition above. For example (foaf:knows is defined to be Symmetric):

```
IF Peer1 'knows' Person2
THEN Peer2 'knows' Person1
```

In our programming language POSL, the above relation can be expressed in Example 4.1 (although it will lead to infinitive loops in sequential Prolog implementations):

Example 4.1

```
knows(?Peer2, ?Peer1) :-
                          knows(?Peer1, ?Peer2).
```

Likewise, a relation R is *Asymmetric*, as shown in Equation 4.2, if for any x, y in S, where there is xRy, then yRx is not satisfied:

$$\forall x, y \in S, xRy \Rightarrow \neg(xRy) \tag{4.2}$$

Therefore, a FOAF rule, in particular a binary relation, is *Asymmetric* only if it is not *Symmetric*. An example of an *Asymmetric* FOAF rule is given as follows:

```
IF Peer1 'seeksExpertise' X,
   Peer2 'offersExpertise' X
THEN Person1 'expertiseMatch' of Person2
```

In POSL, the above relation can be expressed as as shown in Example 4.2:

Example 4.2

```
expertiseMatch(?Peer1,?Peer2) :-
                                  seeksExpertise(?Peer1,?X),
                                  offersExpertise(?Peer2,?X).
```

It is obvious that, from *expertiseMatch(?Peer1,?Peer2)*, we cannot infer *expertiseMatch(?Peer2,?*
since no evidence shows that *Person1* has the expertise X which *Person2* seeks.

Note that in FOAF rules, we can extend a *Symmetric*/*Asymmetric* rule to ternary, or even n-ary, while only the first two premises are of the specific relation, and other premises could be a third party person or some certain conditions. For example, the *Symmetric* relation 'collaborates' is described as follows:

```
IF Peer1 'collaborates' with Peer2 on Project
THEN Peer2 'collaborates' with Peer1 Project
```

We describe the *Symmetric* rule above in POSL shown in Example 4.3:

Example 4.3

```
collaborates(?Peer2,?Peer1,?Project) :-
                                    collaborates(?Peer1,?Peer2,?Project).
```

4.2.2 Transitive Relation

A binary relation can also be defined either as *Transitive* relation or *Non-Transitive* relation. A relation R over a set S is *Transitive*, as expressed in Equation 4.3, for any x, y, z in S, if there exists xRy and yRz then we have xRz. This *Transitive* characteristic can be expressed as follows:

$$\forall x, y, z \in S, xRy \wedge yRz \Rightarrow xRc \tag{4.3}$$

In particular, a FOAF rule is *Transitive* ***iff*** it satisfies the condition above. For example:

```
IF Peer1 'knows' Peer2,
   Peer2 'knows' Peer3
THEN Peer1 'knows' Peer3
```

The POSL version of the rule given above is as in Example 4.4:

Example 4.4

```
knows(?Peer1, ?Peer3) :-
                          knows(?Peer1, ?Peer2),
                          knows(?Peer2, ?Peer3).
```

Likewise, a *Non-Transitive* relation is described in Equation 4.4. That is, if for any x, y, z in S, given xRy and yRz, we cannot infer xRz.

$$\forall x, y, z \in S, xRy \wedge yRz \Rightarrow \neg(xRz) \tag{4.4}$$

Therefore, a FOAF rule, particular a binary relation, is *Non-Transitive* only if it is a relation that is not *Transitive*. An example of a *Non-Transitive* FOAF rule is given as follows:

```
IF Peer1 'knows' Peer2,
   Peer1 'collaborates' Peer2 on Project
THEN Peer1 'knowsWell' Peer2
```

Example 4.5 depicts the POSL expression of this rule.

Example 4.5

```
knowsWell(?Peer1,?Peer2) :-
                              knows(?Peer1,Person2),
                              collaborates(?Peer1,?Peer2,?Project).
```

Given knowsWell(Person1, Person2) and knowsWell(Person2, Person3), we cannot draw a conclusion that knowsWell(Person1, Person3) since no evidence shows that *Person 1* and *Person 3* have collaborated in a same project.

Similarly, as discussed in subsection 4.2.1, we can also extend this *Transitive* relation to ternary, or even n-ary. An example is given below:

```
IF Peer1 'collaborates' Peer2,
   Peer2 'collaborates' Peer3
THEN Peer1 'collaborates' Peer3
```

In POSL, this relation can be expressed as in Example 4.6:

Example 4.6

```
collaborates(?Peer1,?Peer3,?Project):-
                                        collaborates(?Peer1,?Peer2,?Project),
                                        collaborates(?Peer2,?Peer3,?Project).
```

Table 4.1: *Individualized* Vs. *Non-Individualized* Rules

Classification	**Expressions**
Individualized	Rel(?Person, ?Argument$_2$..., ?Argument$_N$)
Non-Individualized	Rel(Person$_1$, ?Argument$_2$..., ?Argument$_N$)

4.2.3 Individualized and Non-Individualized Rules

We distinguish *Individualized* and *Non-Individualized* rules in RuleML FOAF, both of which are within the scope of agent-centric rules.

Agent, as introduced in FOAF, namely FOAF:Agent, is used to allow other kinds of entities, besides person, to describe themselves by means of FOAF page. Currently, foaf:Agent is constituted by three subclasses, foaf:Person, foaf:Organization and foaf:Group.

Since FOAF has been making an effort to describe persons, along with their relationships and their activities, we find that there is a necessity in RuleML FOAF to distinct *Individualized* and *Non-Individualized* rules.

Individualized rules are n-ary rules whose first agent argument is individual constant. Likewise, *Non-Individualized* rules are n-ary rules whose first agent argument is individual variable. A comparison between them is given in Table 4.1.

Examples of *Individualized* rule and *Non-Individualized* rule are shown below sequentially:

As shown in Example 4.7.1, the first argument of rule *atWork* is an individual constance expressing a person whose name is Peter_Pan.

Example 4.7.1

```
atWork(Peter_Pan, ?Time) :-
                              inInterval(?Time, 9, 17).
```

As shown in Example 4.7.2, the first argument of rule *fanOf* is an individual variable expressing an arbitrary person, introduced by a variable symbol '?'.

Example 4.7.2

```
fanOf(?Person,?Band) :-
                        go2Concert(?Person, ?Band, ?frequency),
                        greaterThan(?frequency, 2:Integer).
```

4.2.4 Local and Global Rules

In RuleML FOAF, rules can also be categorized as *Local* rules and *Global* rules. Similarly to *Individualized* and *Non-Individualized* rules, *Local* and *Global* rules also deal with agent-centric rules. More specifically, *Local* and *Global* rules are related to a person's FOAF page, that is, if the inference of a rule need only the metadata from only one FOAF page, it is regarded as a *Local* rule. Otherwise, if the inference of a rule needs metadata from other FOAF page(s), it is then regarded as a *Global* rule. Therefore, agents in this domain are those who have FOAF pages.

The categorization of rules in RuleML FOAF, therefore, can be represented as *Global*, with its subcategories *Individualized/Local* and *Non-Individualized/Local*, and *Local* rules, with its subcategories *Individualized/Global* and *Non-Individualized/Global*.

4.2.4.1 Local Rules

An n-ary rule, having exactly one argument that is a person while zero or more arguments are non-persons, is defined as a *Local* rule. A *Local* rule need only information from a single FOAF page, since this agent-centric rule contains exactly one agent as its argument. However, this agent need not be specified, that is, s/he can be an arbitrary agent. The reason for it is that the rule carrying a variable agent argument can be applied to any single FOAF page of this agent. To be more explicit, an example in RuleML FOAF is provided as follows:

Given a certain date, as shown in Example 4.8, a person's availability is set to a real number 0.9 when he/she is at work then.

Example 4.8

```
availability(?Peer, ?Date, 0.9:Real) :-
                                    atWork(?Peer, ?Date).
```

Local rules can derive predicates of one agent at a time. Two kinds of rules are involved in this category.

- *Individualized/Local*: *Individualized/Local* rules are those *Individualized* rules using individual constants for the person. Since the rules in this scope are agent-centric, with a single constant agent argument, *Individualized* and *Local* rules are of a specified agent's FOAF page. An example of this *Individualized/Local* rule has been given in Example 4.7.1. Note that *Individualized/Local* rules are attached to a certain agent's FOAF page.

- *Non-Individualized/Local*: *Non-Individualized/Local* rules are those *Individualized* rules using individual variables instead. Different from *Individualized/Local* rules, a *Non-Individualized /Local*, with a single variable agent argument, can be applied to an arbitrary FOAF page. Example 4.8 provides as an instance of a *Non-Individualized/Local* rule, which is applicable to any individual person. Note that a *Non-Individualized/Local* rule can be stored either in an agent's FOAF page or in a shared database, depending on the requirement of the user.

4.2.4.2 Global Rules

Likewise, a *Global* rule is defined where at least two arguments are agents. *Global* rules, are not restricted of the metadata of a single FOAF page. Instead, the execution of a *Global* rule need the metadata from at least two FOAF pages because the arguments of a *Global* involves two or more agents. Similarly, the agents in this rule need not be specific. These rules can be applicable to either all agents' FOAF pages or some certain agent(s)' FOAF pages. An example of a *Global* rule is depicted in the following:

As depicted in Example 4.9, the communication between two persons can be determined as 'synchronization' if the geographic distance of their addresses is defined as 'reachable'.

Table 4.2: *Individualized/Global* Vs.*Non-Individualized/Global*

	Individualized	**Non-Individualized**
Local	R(Person, ?Non-Person$_1$, ...?Non-Person$_N$)	R(?Person, ?Non-Person$_1$, ...?Non-Person$_N$)
Global	R(Person$_1$, ?Person$_2$, ?Argument$_3$, ...?Argument$_N$)	R(?Person$_1$, ?Person$_2$, ?Argument$_3$, ...?Argument$_N$)

Example 4.9

```
communication(?Peer1,?Peer2,synchron) :-
                                        address(?Peer1,?Address1),
                                        address(?Peer2,?Address2),
                                        geoDis(reachable,?Add1,?Add2).
```

Global rules can derive relations between two or more agents at a time. Two kinds of rules are involved in this category, where Table 4.2 provides a comparison among the four categories of rules.

- *Individualized/Global*: *Individualized/Global* rules are those *Global* rules using individual constants for the first person. A rule that is both *Individualized* and *Global* is attached to a certain agent's FOAF page for it is specified by this agent as the first individual constant argument of the rule. However, unlike *Individualized/Local* rules, a *Individualized/Global* rule describes the relationships with other agents, which could be pre-defined agent(s) or arbitrary ones.

- *Non-Individualized/Global*: *Non-Individualized/Global* rules are those *Global* rules using individual variables for the first person. A *Non-Individualized /Global* rule, like *Non-Individualized/Local*, can be either attached to an agent's FOAF page or stored in a shared database, depending on the requirement of the user.

4.2.5 Ground and Non-Ground Rules

A rule can be either *Ground* or *Non-Ground*. Ground rules, in a general sense, are rules that are modified and amended to meet specified circumstances [60].

In RuleML FOAF, we distinguish three kinds of rules, two of which are *Non-Ground*, namely *All arguments are variables* and *Some arguments are variables*, while the remaining one is *Ground*, namely *All Arguments are Constant*.

4.2.5.1 Ground Rules

In RuleML FOAF, ground rules are those rules of which all the arguments are constant. They are in certain person's profile in special circumstances.

An example of ground rules has been shown in Example 4.10, a set of time-dependant rules showing the phone preferences of Peter_Pan.

Example 4.10

```
phonePreference(Peter_Pan,office) :-
                                    time(9-12) OR time(13-17).
phonePreference(Peter_Pan,cell) :-
                                    time(12-13) OR time(17-18).
phonePreference(Peter_Pan,home) :-
                                    time(18-21).
phonePreference(Peter_Pan,voicemail) :-
                                    time(21-9).
```

4.2.5.2 Non-Ground Rules

Non-ground rules includes two subcategories: all arguments are variables and some arguments are variables, as listed as follows:

- All Arguments are Variables: None of the arguments in this kind of rule is constant. This kind of rule is a Non-Individualized rule, applicable to anybody, in any circumstance. Early in this chapter, we have shown many rules of this category, such as Example 1, Example 3 and Example 5.

- Some Arguments are Variables: There are cases, in RuleML FOAF, that some arguments are variables while others are constant. Example 4.7.1, Example 4.8 and Example 4.9 are all of this category.

4.2.6 Conditional Rules and Exceptional Rules

RuleML FOAF also distinguishes conditional rules and exceptional rules.

Conditional Rules Most of the RuleML FOAF rules are conditional rules, expressing if the premisse meets this condition, then the conclusion is drawn. These conditional rules are simply introduced by ':-' in POSL and '¡Implies¿ in RuleML.

Exceptional Rules Exceptional rules are needed to express 'if not'. Negation as failure is used in RuleML FOAF to address this requirement, meaning the negation of a condition is true ***iff*** the condition cannot be proved true. Negation as failure is expressed as 'naf(...)' in POSL and '<Naf>' in RuleML.

4.3 Rules Extending FOAF Profiles for Social Networking

Currently, FOAF has been applied to many areas in social networking. Applications of this category include: Flink (http://prauw.cs.vu.nl:8080/flink) which enables the FOAF-based website to display the social network of a group of researchers; the RDFweb (http://rdfweb.org/2002/01/photo/) where the FOAF developer website conducts the first image annotation experiment using FOAF; Plink (http://beta.plink.org), a FOAF-based search engine providing social networking services not only links between people, but also people with closer relationships.

Therefore, RuleML FOAF aims at applying rules to extend the current FOAF fact profiles for social networking. Rules added to FOAF can extend a person's profile facts to make implicit properties and relationships with other persons explicit. For example, the symmetry property of the 'knows' relation shown in Example 4.1 and the transitive 'knows' closure, 'knowsTrans', expressed below, can help group together persons for networking, consultation, collaboration on the same interests and so on.

```
IF A knows B THEN A knowsTrans B
IF A knows B AND B knowsTrans C
THEN A knowsTrans C
```

RuleML markup can also constitute properties conditional on other agents, the time, the location [49] and so on. Time-dependent rules such as preferred phones to call a FOAF page owner for different time intervals can alert both humans and machines about the preferences of homepage owners depending on their agent-centric metadata.

4.4 Taxonomies in RDFS

As FOAF has an ontology vocabulary, we give a brief introduction on the definition of ontology.

People define ontology as "an explicit specification of a conceptualization" [42] or "a shared understanding of some domain of interest" [66].

In RuleML FOAF, ontologies are used for [53]:

A common vocabulary of terms: Our adaptation of original FOAF vocabulary

Some specification of the meaning of the terms: Our RuleML FOAF specification

A shared understanding for people and machines: Taxonomies in RDFS

In RuleML FOAF, we use RDFS to describe taxonomies, since its subPropertyOf takes great advantage in describing classification. Moreover, OO jDREW also provides a type definition for RDFS.

We used ACM classification for peoples' expertise in our use case of expert finding. Benefitting from our previous Semantic Web Techniques Final Report which provides the XML file of the ACM computing classification, we designed an XSLT to transform this XML file to RDF.

The generated RDFS can be run in OO jDREW type definition. After parsing these types, the taxonomy provided by RDFS can be used for RuleML FOAF execution. A

hierarchy of expertise is provided in our use case such that people can express their specified expertise and query their required expertise, benefitting from this refined structure.

The stylesheet *ACM2RDF.xsl* is designed for transforming the ACM Computing Classification from XML into RDFS.

ACM2RDF.xsl

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
    <xsl:template match="/">
        <rdf:RDF xml:lang="en"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#">
            <rdf:Description
            xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
             rdf:ID="{name(Computing)}">
                <rdf:type
                rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
            </rdf:Description>
            <xsl:apply-templates/>
        </rdf:RDF>
    </xsl:template>
    <xsl:template match="*">
        <xsl:for-each select="*">
            <rdf:Description
            xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
            xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
            rdf:ID="{name()}">
                <rdf:type
                rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/>
                <xsl:for-each select="../.">
                    <rdfs:subClassOf rdf:resource="#{name()}"/>
                </xsl:for-each>
            </rdf:Description>
        </xsl:for-each>
        <xsl:for-each select="*">
            <xsl:apply-templates select="."/>
        </xsl:for-each>
    </xsl:template>
</xsl:stylesheet>
```

The following shows the first two levels of ACM Computing Classification, with "computing" as its root node.

Document 1: ACM in XML

```
<Computing>
    <Hardware/>
    <Computer_systems_organization />
    <Software />
    <Data />
    <Theory_of_computation />
    <Mathematics_of_computing />
    <Information_systems />
    <Computing_methodologies />
    <Computer_applications />
    <Computing_milieux />
</Computing>
```

The RDFS of ACM Computing Classification can be derived by applying the stylesheet *ACM2RDF.xsl* to the *ACM in XML*. The following shows the output file in RDFS of *ACM2RDF.xsl* corresponding to *Document 1: ACM in XML*. For the whole ACM classification in RDFS, see http://www.ruleml.org/usecases/foaf/findxprt/acmclassification.

ACM Computing Classification in RDFS

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs="http://www.w3.org/TR/1999/PR-rdf-schema-19990303#"
xml:lang="en">
    <rdf:Description rdf:ID="Computing">
        <rdf:type
        rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
    </rdf:Description>
    <rdf:Description rdf:ID="Hardware">
        <rdf:type
        rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
        <rdfs:subClassOf rdf:resource="#Computing" />
    </rdf:Description>
    <rdf:Description rdf:ID="Computer_systems_organization">
        <rdf:type
        rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
        <rdfs:subClassOf rdf:resource="#Computing" />
    </rdf:Description>
    <rdf:Description rdf:ID="Software">
        <rdf:type
        rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
        <rdfs:subClassOf rdf:resource="#Computing" />
    </rdf:Description>
    <rdf:Description rdf:ID="Data">
        <rdf:type
        rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
        <rdfs:subClassOf rdf:resource="#Computing" />
```

```
    </rdf:Description>
    <rdf:Description rdf:ID="Theory_of_computation">
        <rdf:type
        rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
        <rdfs:subClassOf rdf:resource="#Computing" />
    </rdf:Description>
    <rdf:Description rdf:ID="Mathematics_of_computing">
        <rdf:type
        rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
        <rdfs:subClassOf rdf:resource="#Computing" />
    </rdf:Description>
    <rdf:Description rdf:ID="Information_systems">
        <rdf:type
        rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
        <rdfs:subClassOf rdf:resource="#Computing" />
    </rdf:Description>
    <rdf:Description rdf:ID="Computing_methodologies">
        <rdf:type
        rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
        <rdfs:subClassOf rdf:resource="#Computing" />
    </rdf:Description>
    <rdf:Description rdf:ID="Computer_applications">
        <rdf:type
        rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
        <rdfs:subClassOf rdf:resource="#Computing" />
    </rdf:Description>
    <rdf:Description rdf:ID="Computing_milieux">
        <rdf:type
        rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
        <rdfs:subClassOf rdf:resource="#Computing" />
    </rdf:Description>
</rdf:RDF>
```

However, there are cases that, in RDFS, we need to deal with multiple inheritance. A subclass may have been defined for multiple times, while only one unique "rdf:ID" is required. An example has been shown below:

The class *Hardware*, as shown in Figure 4.2, is an example of multiple inheritance. *Hardware* is not only the subclass of the root element *Computing*, but also the subclass of class emphPersonal_computing as well as class *History_of_computing*. We describe the multiple inheritance issue as follows, providing class "Hardware" a unique ID:

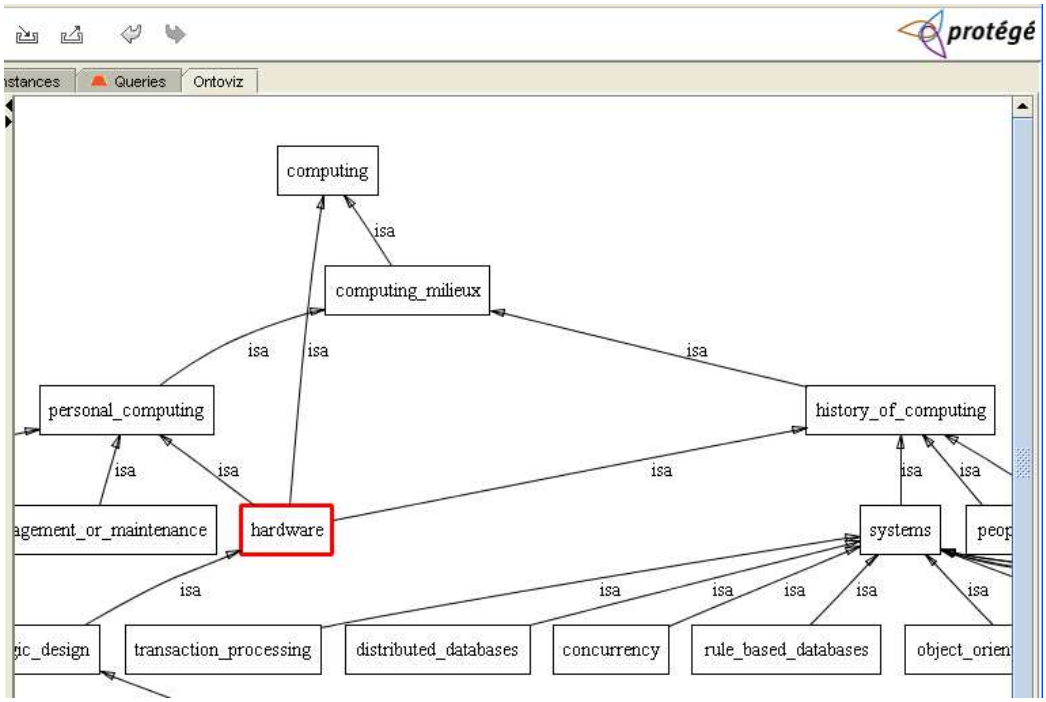

Figure 4.2: Visualization of a part of ACM Taxonomy

```
<rdf:Description rdf:ID="Hardware">
    <rdf:type
    rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class" />
    <rdfs:subClassOf rdf:resource="#Computing" />
    <rdfs:subClassOf rdf:resource="#Personal_computing" />
    <rdfs:subClassOf rdf:resource="#History_of_computing" />
</rdf:Description>
```

4.5 Classification of Facts

In RuleML FOAF, facts can be classified into two categories: given facts (properties) and rule-derivable facts (properties), while rule-derivable facts include taxonomic derivations and general derivations. Note that since RuleML FOAF, like RDF-based FOAF, focuses mostly on describing agents (mainly people) and their relationships, both the definition of given facts and rule-derivable facts are in the scope of agent-centric metadata.

The classification of facts is essential when implementing the Rule-oriented Normal Form (RNF) which will be introduced in Section 4.8. Here given facts and rule-derivable facts are

introduced in subsection 4.5.1 and 4.5.2 respectively.

4.5.1 Given Facts (Properties)

Given facts are facts that are not derivable by rules. In other words, these facts do not have enough premisses to be inferred by rules. Therefore given facts must be stored in database, not being removed in any case. Otherwise the information will get lost and thus cause the loss of possible premisses for deriving new conclusions.

Expertise of a person, in Example 4.11 (rule-1), is described as being rated in that area with a score greater than 4 and having had working experience of more than two years. With (rule-2), a person's expertise is described as if he/she has more than three publications. And (rule-3) expresses that a person has that expertise if he/she has more than six recorded CDs of that area.

In Example 4.11, fact-0 is a given fact because, being the only metadata describing Bill, none of rule-1, rule-2 and rule-3 have enough premisses that can infer that Bill has the expertise of AI. Indeed fact-2, fact-3, fact-4 and fact-5 are given facts in RuleML FOAF, for they are all agent-centric and non-derivable facts.

Example 4.11

```
(rule-1)
expertise(?Peer,?Area) :-
                          rating(?Peer,?Area,?Score),
                          greaterThan(?Score,4),
                          workDuration(?Peer,?Area,?Year),
                          greaterThanOrEqual(?Year,2).
(rule-2)
expertise(?Peer,?Area) :-
                          publication(?Peer,?Area,?Amount),
                          greaterThan(?Amount,3:Integer).
(rule-3)
expertise(?Peer,?Area) :-
                          RecordedCDs(?Peer,?Area,?Amount),
                          greaterThan(?Amount,6).
```

(fact-0)	expertise(Bill,AI).
(fact-1)	expertise(Peter_Pan,AI).
(fact-2)	rating(Peter_Pan,AI,5).
(fact-3)	workDuration(Peter_Pan,2).
(fact-4)	publication(Peter_Pan,AI,4).
(fact-5)	RecordedCDs(Lucy_Alm,Pop,6).

4.5.2 Rule-derivable Facts (Properties)

Rule-derivable facts are facts that are derivable by relevant rules, together with the related primitive facts. Note that rule-derivable facts can also be stored in database before deduction. However, it is not necessary to store them, and in one of our normal form, namely Rule-oriented Normal Form (RNF), discussed later in Section 4.8, all rule-derivable facts are removed when published for compactness.

As a rule-derivable fact can always be derived by correspondent rules, it can be also conceived as the conclusion of a rule, i.e., the head of the rules. As such, the specification of rule-conclusion vocabulary involves the design of rule-derivable fact vocabulary.

As shown in Example 4.11, fact-1 is a rule-derivable fact which can be derived by both rule-1 with the satisfying premisses fact-2 and fact-3, and rule-2 with the satisfying premisses fact-4. In this sense, fact-1 is both stored and derivable. Moreover, after executing the rules in Example 4.11, there will be a newly derived fact in the database, expertise(Lucy_Alm,AI), by rule-3 with fact-5 which satisfies it.

The following describes two types of rule-derivable facts: from taxonomic derivations and from general derivations.

4.5.2.1 Taxonomic Derivations

This category includes properties that can be generated by taxonomic derivations (e.g., using RDF's subPropertyOf or subClassOf).

Example 4.12.1 shows generating properties by taxonomic derivations using RDF's subPropertyOf. If Person1 *knowsWell* Person2 then they must *knows* Person2. Likewise, if Person1 and Person2 have a *partner* relationship, they must also *knowsWell* each other.

Example 4.12.1

```
knows(?Peer1, ?Peer2) :-
                                knowsWell(?Peer1, ?Peer2).
knowsWell(?Peer1, ?Peer2) :-
                                partner(?Peer1, ?Peer2).
```

Example 4.12.2 expresses if a person is a *rocketScientist* then he/she must be an *expert*, as is using RDF's subClassOf.

Example 4.12.2

```
expert(?Peer) :-
                        rocketScientist(?Peer).
```

4.5.2.2 General Derivations

This category includes properties that are generated by general derivations. Most of the RuleML FOAF properties are of this category.

Example 4.12.3 depicts that we can infer that Person1 *knowsWell* Person2 if they have collaborated in the same project and they have the same hobby.

Example 4.12.3

```
knowsWellEachOther(?Peer1, ?Peer2) :-
                                        collaborateIn(?Peer1, ?Peer2, ?Project),
                                        like(?Peer1, ?Hobby),
                                        like(?Peer2, ?Hobby).
```

4.6 Rules for Enhancement of FOAF

Dan Brickley in [30] provides us with a brief explanation of FOAF:

> "FOAF is all about creating and using machine-readable homepages that describe people, the links between them and the things they create and do."

FOAF expresses online relationships between people. People have kinds of relationships between each other. Suppose that the degree of closeness decreases from the innermost circle to the outer circle. Peoples's social networks are then represented as interlocking social circles, while there are circles that are also localized.

FOAF is built on an open infrastructure and enables each FOAF person to own their own profiles, having access and modifying the data in it [36]. One of the advantages of FOAF is supported by *foaf:knows* which makes use of *rdfs:seeAlso* so that in his/her own homepage, a FOAF person can point to the homepages of whom he/she knows.

Portals of e-Commerce, such as Amazon, have already been developing personalization for each of their potential customer. They suggest commodities to their customers based on their declared preference and more importantly, on their purchase history.

However, the attractiveness of FOAF is that information can be gathered from multiple homepages and can be of multiple uses by many sites. Moreover, using FOAF, such portals can also make suggestions to their customers by inferencing from the purchase history of those people who share the similar interests with them.

Besides the application in social network, FOAF has also been applied to other areas. For example, using FOAF in Email spam, people can avoid junk mail from people that they do not know. However, a more impressive application would be leveraging a network of business contacts [2].

FOAF, as an Semantic Web application, strives for the best understandability of computer. Moreover, FOAF harvesters have hence been developing with a Web-crawling aspect. Applying rules in FOAF can not only help the understandability of computers, but also enrich the FOAF harvester both in crawling across the Web and updating FOAF profiles, which can be achieved by reapplying rules in the rule engine.

The main purpose of implementing rules is to perform certain actions, such as querying a certain expert with several conditions; finding a specialized expert via consulting a more general expert; making decisions on whether to collaborate according to experts own criteria; designing different collaboration modes according to the convenience of people, the location, the time zone and so on, as can be developed into rule-based intelligent assistants ('proxies').

Rules are also able to 'harvest' metadata from other homepages, accumulating and filtering the results. An important special case is computing those subsets of the transitive closure of the foaf:knows property (relation) that contain one or more given persons. Like

in the HTML Web, distributed RuleML FOAF homepages can permit everyone to copy and edit from other persons' published rulebases, and agents will be able to apply the rules. This transitive relationship can be applied to an expert's social networks so that he is able to recommend a specialized expert when a client consults with him.

Therefore, in this thesis we specify a rule vocabulary augmenting the current fact-only FOAF vocabulary. Once rules are realized, various mathematical methods, such as graph-theoretic, algebraic, and logic, can be applied to RuleML FOAF. Moreover, we can provide foundations for making FOAF tools rule-aware by coupling them with rule engines such as OO jDREW for specifying and executing FOAF rules.

4.7 RuleML FOAF Vocabulary Specification

The FOAF community has published FOAF vocabulary as its schema and specification, online available at *http://xmlns.com/foaf/0.1*, where the FOAF vocabulary are designed to be categorized as classes and properties.

However, since FOAF is an RDF application, its vocabulary is inherently extensible to a comprehensive application area. Many extensions to FOAF vocabulary have already been made for variant applications.

Impressively, extensions of FOAF vocabulary need not be realized through a centralized model. Independent vocabulary design is the main stream in FOAF vocabulary extension, that is, no universal agreement is needed. This can benefit the inter-operation within the RDF framework and hence enhance the collaborative theme of FOAF [19].

RuleML FOAF vocabulary specification is essential for the implementation of this thesis. Reuse of, and extension to, FOAF vocabulary also varies among different applications.

4.7.1 Reuse of the Current FOAF Vocabulary

The current FOAF vocabulary has been developed to describe agents, mainly people. It groups its vocabulary into five categories: FOAF basis (including foaf:Agent, foaf:Person,

foaf:name and so on), Personal Info (including the famous foaf:knows, foaf:interest and so on), Online Accounts / IM (e.g. foaf:OnlineAccount, foaf:accountName and foaf:jabberID), Projects and Groups (e.g. foaf:Group, foaf:Organization, foaf:member and foaf:Project), and Documents and Images (e.g. foaf:Document, foaf:Image and foaf:topic).

FOAF has its priority in describing relationships among people. The foaf:knows property, expressing the relationship that one foaf:Person knows the other foaf:Person, has become well-known along with the growth of the FOAF project. By making use of rdfs:seeAlso property, a foaf person's homepage always points to other foaf persons' homepages in which machines can process the metadata within them. Moreover, rdfs:seeAlso corresponds to the anchor element in HTML for referencing display purposes (An anchor is for marking the beginning and/or the end of a hypertext link [1]). This enables FOAF harvester to gather metadata through homepages and build a database about them [36].

However, only foaf:knows cannot express the different levels of closeness among people. There have been many efforts on complementing FOAF vocabulary since FOAF is designed as an open interchange format that can be extended for variant applications. There are attempts to define additional vocabularies such as *knowsWell*, *livesWith*, *husband* as sub-properties of foaf:knows. Masahiro Hamasaki et al. in [43] also show some early work on defining an ontology on human relationships, which extends the foaf:knows with friendOf, acquaintanceOf, parentOf, and hasMet, expressing different degrees of acquaintance.

In our earlier development of RuleML FOAF, we have extended FOAF vocabulary for finding common interests among people within the domain of music. Different degrees of similarity towards music preference imply the different degrees of closeness between two persons. We distinguished different degrees of familiarity among people by identifying different relationships, namely 'knows', 'knowsWell'.

In this thesis, FOAF vocabulary has been extended for our expert finding application. Table 4.3 shows a set of vocabularies adapted either from the official release of FOAF vocabulary or from [43] vocabulary extensions. However, these specifications have been modified

[1] http://www.utoronto.ca/webdocs/HTMLdocs/NewHTML/anchors.html

Table 4.3: Reuse of the FOAF Vocabulary

Vocabulary	Category	Source	Description
knows	Property	FOAF Vocabulary	Knows person
worksWith	Property	Reference [43] Extension	Work for the same employer
participantIn	Property	[43] Extension	Participates in a project
participant	Property	[43] Extension	Participant of a project
mentorOf	Property	[43] Extension	Serves as a trusted expert
employedBy	Property	[43] Extension	Engages the services
employerOf	Property	[43] Extension	Engaged in the services
collaboratesWith	Property	[43] Extension	Work towards a common goal
mbox	Property	FOAF Vocabulary	A personal mailbox
page	Property	FOAF Vocabulary	A FOAF page
name	Property	FOAF Vocabulary	A name for something
title	Property	FOAF Vocabulary	Title (Mr, Mrs, Ms, Dr. etc)
interest	Property	FOAF Vocabulary	Interested research area
topic_interest	Property	FOAF Vocabulary	Interested area for knowledge transfer
publications	Property	FOAF Vocabulary	Publications of this FOAF person
currentProject	Property	FOAF Vocabulary	A ongoing project of this FOAF person
pastProject	Property	FOAF Vocabulary	A finished project of this FOAF person
fundedBy	Property	FOAF Vocabulary	Funded by an organization
membershipClass	Property	FOAF Vocabulary	A member of a Group
Person	Class	FOAF Vocabulary	A FOAF person
Group	Class	FOAF Vocabulary	A class of Agents
Project	Class	FOAF Vocabulary	A (collective) project
Organization	Class	FOAF Vocabulary	An organization
PersonalProfileDocument	Class	FOAF Vocabulary	Personal profile RDF/RuleML document

based on two criteria:

- We have Non-Individualized the expression of RuleML FOAF vocabulary: for vocabulary of classes, the first letter should be upper case; for vocabulary of properties, the first letter should be lower case.
- Descriptions of each vocabulary have been adapted with respect to the expert finding application of RuleML FOAF.

4.7.2 RuleML FOAF Extensions

As mentioned above, FOAF is a open source project so it tends to be extended for variant applications. RuleML FOAF has been endeavoring to extend current FOAF vocabulary.

Vocabulary extension for RuleML FOAF can be categorized as two parts: extensions to fact vocabulary and rule-conclusion vocabulary specification.

Note that RuleML FOAF extension of the current FOAF vocabulary mainly involves vocabulary of properties in, but not vocabulary of classes. It is because the vocabulary of class expresses the categories or types of an object, while the vocabulary of properties introduces relationships between its two parameters. Both FOAF and RuleML FOAF are agent (person) centric so that the types of objects are almost similar. However, the vocabulary of properties varies from application to application since different applications focus on describing different relationships between objects. Therefore this thesis mainly develops the vocabulary of properties.

In this section, we mainly focus on the vocabulary specification of our use cases, especially the expert finding use case. Later in this section, discussions are also provided on FOAF vocabulary extensions for other applications, in a broad view.

Extensions to FOAF vocabulary for RuleML FOAF can be categorized as extending fact vocabulary and rule-conclusion vocabulary, introduced as follows:

4.7.2.1 Classification of RuleML FOAF Specification

Specification of RuleML FOAF vocabulary involves two parts: specifying the fact vocabulary and the rule-conclusion vocabulary as well. Since the reuse of the current FOAF vocabulary for RuleML FOAF has already been discussed in the previous subsection, we concentrate solely on extensions to both fact and rule-conclusion vocabulary in this subsection.

1. **Extensions to Fact Vocabulary:**

   In order to improve the expressivity of RuleML FOAF, we attempt to refine RuleML FOAF vocabulary for FOAF persons fact profiles. As has already been discussed in Section 4.5, the classification of facts in RuleML FOAF can be represented as given facts and rule-derivable facts. Here, extensions to fact vocabulary are within

the domain of given facts.

RuleML FOAF fact vocabulary describes a person's fact profile, either in RuleML or in POSL syntax. Fact vocabulary involves both the slot names and the filler names. However, in RuleML FOAF vocabulary specification we only concentrate on slot names and those filler names where certain taxonomies are involved in.

We first give descriptions to vocabularies about the slot names, which draw the principal segment of the fact vocabulary specification, followed by a discussion about the specification about filler names.

Fact vocabulary includes all vocabulary of classes, while the Person class remains the most essential part in it. Therefore, our extension to fact vocabulary is designed with Person vocabulary as the core. In RuleML FOAF, fact vocabulary also often constitutes ontologies, more precisely, taxonomies. For instance, the expert finding application draws upon ACM computing classification as well as the SeSDL taxonomies.

In application of expert finding, we focus on describing information about researchers, their research interests, professional history and other personal information. All slot names with RuleML FOAF extension are marked by an 'ex' namespace, while the original FOAF vocabulary specification can be identified by 'foaf' namespace. Filler names also belong to fact vocabulary specification, where no namespace is introduced for them.

In expert finding application, we also take advantage of the 1998 ACM Computing Classification [18] for specifying the vocabulary about research areas of researchers, particularly computer scientists. Human skills of these researchers on their research area are also specified as filler names based on the SeSDL (Scottish electronic Staff Development Library) Taxonomy Evaluation Technology [35]. Both the ACM classification and SeSDL taxonomy are developed as filler names in RuleML FOAF. RuleML FOAF vocabulary specifies them with the purpose of precisely distinguishing peoples' expertise and human skills.

Table 4.4: RuleML FOAF Rule-Conclusion Vocabulary Extension

Vocabulary	Category	Source	Description
Category	Class	General Extension	Specifies the subcategories
rating	Property	General Extension	Rating of a person in a certain area
...	...	General Extension	...
go2Concert	Property	Music Application	The amount of concerts of a certain band or singer that a person goes to
watchTVLive	Property	Music Application	A person watches TV live of a certain band or singer
hasCD	Property	Music Application	The amount of the CD of a certain band or singer that a person has
talkedIn	Property	Music Application	A person is talked in by a third party to go to some activities (e.g. concert)
...	...	Music Application	...
participatesIn	Property	Expert Finding Application	A person participate in a project
workDuration	Property	Expert Finding Application	Years of working of a person in a specific area
hasPublications	Property	Expert Finding Application	The amount of publications that a person has
...	...	Expert Finding Application	...

Table 4.4 provides us with an overview of the extended RuleML FOAF fact vocabulary from our existing applications.

2. **Rule-Conclusion Vocabulary: Derivable Properties**

Rule-conclusion vocabulary is featured as possessing derivable properties. It plays a role as the head of the rule, as can be derived after applying the correspondent rule where all its premisses are satisfied. In this sense, rule-conclusion vocabulary also corresponds to the rule-derivable facts discussed in the previous subsection. Extensions to Rule-Conclusion vocabulary are within the scope of extending the vocabulary of rule-derivable facts.

Since rule-conclusion vocabulary closely relates to rules, which are essential in developing RuleML FOAF, emphasis has been put on extending rule-conclusion vocabulary.

In accordance with RDF FOAF, RuleML FOAF rule-conclusion vocabulary, representation of the conclusions of rules, is also person centric. It dedicates to describe different phases of a person, e.g., his/her networking (originally from FOAF), hobbies (music application) and professional work (expert finding) as well.

Table 4.5: RuleML FOAF Rule-Conclusion Vocabulary Extension

Vocabulary	Category	Source	Description
knowsEachOther	Property	General Extension	Two persons knows each other
partner	Property	General Extension	Two persons are friends with each other
phonePreference	Property	General Extension	Specifies the phone preference during different time interval
...	...	General Extension	...
fanOf	Property	Music Application	A person is a fan of a band or a singer
...	...	Music Application	...
coWorkers	Property	Expert Finding Application	Two persons work for the same organization
colleagues	Property	Expert Finding Application	Two persons participate in the same activity (e.g. project)
consultedBy	Property	Expert Finding Application	The expert is consulted by the client
offersExpertise	Property	Expert Finding Application	An expert offers certain expertise
seeksExpertise	Property	Expert Finding Application	An client seeks certain expertise
collaborates	Property	Expert Finding Application	Two persons are currently collaborating with each other
collaborated	Property	Expert Finding Application	Two persons formerly collaborated with each other
busyWith	Property	Expert Finding Application	A persons is currently busy with some project or other affairs
collaborationMode	Property	Expert Finding Application	Means of collaboration between people
collaborationTopic	Property	Expert Finding Application	Topic of collaboration between people
knowsTrans	Property	Expert Finding Application	An expert transfers his knowledge to
geoDistance	Property	Expert Finding Application	Geographic distance between two persons
...	...	Expert Finding Application	...

Generally speaking, rule-conclusion vocabulary is the basis for two uses, a person's rule profile and global rulebase. Rule-conclusion vocabulary constitutes persons' rule profiles where it represents specified rules by different individuals (personal tailored rules), while it composes global rulebases when it represents rules of general use that can be applied to all individuals.

Unlike fact vocabulary which specifies its source as its namespace, since all the rule-conclusion vocabulary is within the domain of RuleML FOAF extension, we assume its namespace is 'ex.' by default and hence omit it universally.

Table 4.5 provides us with the current rule-conclusion vocabulary specification.

4.7.2.2 Current RuleML FOAF Vocabulary Specification

The current development of RuleML FOAF vocabulary has been depicted in Table 4.3, Table 4.4, and Table 4.5. It first takes advantage of original RDF FOAF vocabulary, and

then develops both fact and rule-conclusion vocabulary for different applications, the music application and find expert application.

Note that, unlike RDF FOAF, neither fact nor rule-conclusion vocabulary are restricted to can have arbitrary arguments, that is, both of them can have arbitrary ones.

4.7.2.3 Extensions for Other Applications

FOAF is designed for extensions for different application domains. Currently, RuleML FOAF has extended RDF FOAF with expert finding application and some early work on music application as well.

Applications to other domains are also of interest to RuleML FOAF. One application of potential interest is the "semantic email", circulating email among those individuals that would be of interest.

RuleML FOAF vocabulary tends to be extended in order to adapt different applications. We follow the same principle when extending RuleML FOAF vocabulary: first take advantage of the existing RuleML FOAF vocabulary and only then, extend it with the specific application.

4.8 Two Normal Forms

Since RuleML FOAF is an extension to the current RDF-based FOAF, we provide different normal forms for RuleML FOAF users with different interests: people from RDF community may mostly be interested in facts, especially newly derived ones; people of rule community, on the other hand may find rules more interesting than purely facts.

Therefore, two normal forms for RuleML FOAF rulebases are proposed in this thesis: a Rule-oriented Normal Form (RNF) and a Fact-oriented Normal Form (FNF).

Both RNF and FNF are represented as published FOAF pages, being the final outcome to RuleML FOAF users. Different from traditional homepages, FOAF page has the feature of being combined with multiple FOAF pages so that all the facts and rules can be stored

in a unified database.

The rule engine OO jDREW *http://www.jdrew.org/oojdrew* has been employed to run RuleML FOAF: It permits bottom-up as well as top-down execution, supporting both Normal Forms.

We first give an introduction to RNF, followed by FNF, and then provide motivating examples for both of them.

4.8.1 Rule-Oriented Normal Form (RNF)

The rule-oriented normal form (RNF) is designed as one of the normal forms for RuleML FOAF. It is especially useful for people who are interested in rule formalization, interchange and execution.

The RNF first reserves all the rules from different sources: individual tailored rules from user's FOAF profiles and universal rules from global rulebases. Then the RNF checks derivability of all the facts through the database; the RNF retains those facts that are non-derivable and discards derivable ones.

Therefore, the RNF includes rules as well as the (elementary) facts that are needed by the premises of the rules. Those facts that are derivable from the rules by a bottom-up engine such as OO jDREW BU are removed from the rulebase.

4.8.1.1 The Need for/Applications of RNF

As already been discussed, RuleML FOAF users who are in favor of developing rulebases are very likely to feel a need for the RNF. The reason for this is that the RNF owns the advantages listed as follows:

- The RNF achieves its compactness: Any facts that can be obtained by applying rules through a forward-chaining engine (e.g. OO jDREW BU) are removed from the rule-oriented normal form. Simply put, the RNF strives for retaining as few amount of facts as possible for rule execution and thus is regarded as a compact normal form.

- The RNF is of great interest for people in rule community: All the rules are either from each individual's profile or global rulebase are retained in RNF. These rules are published so that they can be reused or adapted by people to build their own RuleML FOAF page, or even the global rulebase for people in other networks. Moreover, these published rules can still help people from rule community for developing their own rulebases. Also, the feature that the RNF can also dynamically generating new facts via rule execution inherited the great advantage of RuleML FOAF.

4.8.1.2 Implementation

Currently in RuleML FOAF, the generation of the RNF is achieved interactively. We use both bottom-up and top-down rule engines, such as OO jDREW BU and OO jDREW TD, to help the derivation of the RNF.

The steps that we are following to generate the RNF are listed below:

OO jDREW BU Execution: We post all the facts and rules that we have into a bottom-up engine such as OO jDREW BU in order to get all the derivable facts as well as the original ones.

OO jDREW TD Execution: Complementarily, corresponding queries could be posed to a top-down engine such as OO jDREW TD. After OO jDREW BU execution, we post all the facts, both newly derived and original, as well as all the rules, into OO jDREW TD. We only query for those facts that are originally in our database. Two steps for determining the fact derivability are involved:

1. Select those facts from the original database, one at a time.
2. Delete the one that can is derivable by querying

The reason that we first run OO jDREW BU is because some newly derived facts may be the premisses of some derivable facts that were non-derivable before bottom-up execution.

4.8.2 Fact-Oriented Normal Form (FNF)

In complement with RNF, a fact-oriented normal form (FNF) is put forward in RuleML FOAF. For RuleML FOAF is originated from RDF-based FOAF, one of its goal is to benefit the current FOAF community. Therefore, the FNF is proposed in order to accomplish this goal.

The FNF includes elementary facts as well as derivable facts. All the rules are removed from the published rulebase after all possible facts are derived by running a bottom-up engine such as OO jDREW BU. In short, the FNF combines given facts and newly derived ones provided by OO jDREW BU but omits rules from the published rulebase.

Note that in the FNF, rules are only omitted in the published rulebase. This is because when new given facts are asserted, the rules need to be re-applied to them. These newly inserted facts may be the premisses of certain rules and thus can further infer new facts by executing the rulebase in OO jDREW BU.

4.8.2.1 The Need for/Applications of FNF

While the RNF is more compact, the FNF directly corresponds to RDF FOAF facts. In other words, RDF compatibility is achieved via the fact-oriented normal form.

The FNF provides its users with a normal form that contains only facts, omitting rules in its published form. The original RDF FOAF can only expresses facts, but not rules. Therefore, the FNF of RuleML FOAF can be easily mapped back to the fact-only RDF FOAF.

We list here the advantages of having FNF in RuleML FOAF:

- The FNF provides a bridge to RDF FOAF for RuleML FOAF so that users with the preference of RDF FOAF can also directly make use of FNF.
- The FNF provides us with a way to compare one outcome of RuleML FOAF with RDF FOAF. The FNF contains a list of facts, both elementary and newly derived, while

RDF FOAF holds those facts that are merely elementary. Therefore, the facts that RDF FOAF owns is just a subset of what FNF has.

- FNF also maintains one major advantage of RuleML FOAF: dynamically derived new facts via rules. Although rules are omitted from the published forms, where there are any new facts inserted, the FNF re-apply rules towards facts. Therefore, the FNF is able to show how the most recent change affects its outcome.

4.8.2.2 Implementation

In RuleML FOAF, the FNF is generated via a forward chaining rule engine, such as OO jDREW BU, since bottom-up execution provides us with the all the newly derived facts as required for the FNF.

Currently, like RNF, the generation of FNF is achieved interactively as well. The steps for FNF generation are described as follows:

1. We post all the rules as well as original facts as a knowledge base and parse it via OO jDREW BU.

2. Then by running its forward reasoner, OO jDREW BU provides us with newly derived facts, along with original facts.

3. In the published FNF, we store both newly derived and original facts from the result given by OO jDREW BU.

Note that even in published FNF, facts are written in RuleML syntax, instead of RDF. However, transformation to RDF from RuleML can be easily achieved, as explained in the next subsection.

4.8.2.3 RuleML-Based and RDF-Based FNF

Facts in the RuleML FOAF vocabulary using a subset of RuleML can be easily mapped back to RDF facts via an XSLT translator when necessary. In cases where only facts are

needed in RuleML FOAF, their RDF FOAF form can be automatically generated using the XSLT stylesheet.

We implement a stylesheet, *ruleml2rdf.xslt*, for translating RuleML-Based FNF to RDF-Based FNF (see below).

ruleml2rdf.xslt

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="2.0"
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:foaf="http://xmlns.com/foaf/0.1">
    <xsl:output method="xml" version="1.0" encoding="UTF-8" indent="yes"/>
    <xsl:template match="/">
        <xsl:for-each select="descendant::Rel">
            <xsl:element name="{//Rel}">
                <xsl:apply-templates select="self::Rel/following-sibling::*"/>
            </xsl:element>
        </xsl:for-each>
    </xsl:template>
    <xsl:template match="slot">
        <xsl:for-each select="(Ind)[position()=1]">
            <xsl:element name="{.}">
                <xsl:apply-templates select="self::Ind/following-sibling::*"/>
            </xsl:element>
        </xsl:for-each>
    </xsl:template>
    <xsl:template match="Cterm">
        <xsl:for-each select="Ctor">
            <xsl:element name="{//Ctor}">
                <xsl:apply-templates select="self::Ctor/following-sibling::*"/>
            </xsl:element>
        </xsl:for-each>
    </xsl:template>
    <xsl:template match="oid">
        <xsl:choose>
            <xsl:when test="Ind">
                <xsl:element name="rdf.ID">
                    <xsl:value-of select="Ind"/>
                </xsl:element>
            </xsl:when>
            <xsl:otherwise>
                <xsl:element name="rdf.ID">
            unknown ID
        </xsl:element>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:template>
```

```
</xsl:stylesheet>
```

A simple example of RuleML-Based FNF is shown in *exa.ruleml*, as an input file for the stylesheet *ruleml2rdf.xslt*, followed by the RDF-Based FNF, which is the output file of the transformation, described in *exa.rdf*.An intuitive comparison between RuleML-Based FNF and RDF-Based FNF can thus be drawn.

exa.ruleml

```
<RuleML xmlns="http://www.ruleml.org/0.9/xsd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://www.ruleml.org/0.9/xsd
 http://www.ruleml.org/0.9/xsd/hornlog.xsd">
    <Assert>
        <And>
            <Atom closure="universal">
                <Rel>foaf.person</Rel>
                <oid>
                    <Ind>www.expert1.com</Ind>
                </oid>
                <slot>
                    <Ind>foaf.name</Ind>
                    <Ind>Expert1</Ind>
                </slot>
                <slot>
                    <Ind>foaf.knows</Ind>
                    <Ind>Expert3</Ind>
                </slot>
            </Atom>
        </And>
    </Assert>
</RuleML>
```

exa.ruleml

```
<?xml version="1.0" encoding="UTF-8"?>
<foaf.person>
    <rdf.ID>www.expert1.com</rdf.ID>
    <foaf.name>Expert1</foaf.name>
    <foaf.knows>Expert3</foaf.knows>
</foaf.person>
```

4.8.3 Motivating Examples of the RNF and FNF

In this section, motivating examples illustrating RNF and FNF are provided. These examples are describing how we define a person as a *fanOf* a band, along with exemplary

```
    (rule-1)
fanOf(?Person, ?Band) :-
  hasCD(?Person, ?Band, ?amount),
  greaterThan(?amount, 3:Integer),
  watchTVLive(?Person, ?Band).

    (rule-2)
fanOf(?Person, ?Band) :-
  go2Concert(?Person,?Band,?frequency),
  greaterThan(?frequency, 2:Integer).
```

```
    (fact-0)
fanOf(Bill, U2).

    (fact-1)
fanOf(Peter, U2).
    (fact-2)
hasCD(Peter, U2, 4:Integer).
    (fact-3)
watchTVLive(Peter, U2).
    (fact-4)
go2Concert(Peter, U2, 3:Integer).

    (fact-5)
go2Concert(Lucy, U2, 5:Integer).
```

Figure 4.3: Original Rulebase

persons: Bill, Peter and Lucy.

Figure 4.3 shows the original rulebase, including all the rules and given facts. The corresponding RNF is depicted in Figure 4.4, followed by the corresponding FNF in Figure 4.5.

For transforming this to RNF, (fact-1) can be removed from the rulebase because it can be derived from either (rule-1) or (rule-2).

Likewise, for transforming this to FNF, all these rules are removed from the published rulebase after a new fact expertise(Lucy_Alm,Pop) has been derived by running OO jDREW BU with (rule-1) and (rule-3).

The illustrations of both RNF and FNF are shown respectively in Figure 4.6, with all the rules and given facts, and Figure 4.7, containting only facts, both given and newly derived ones.

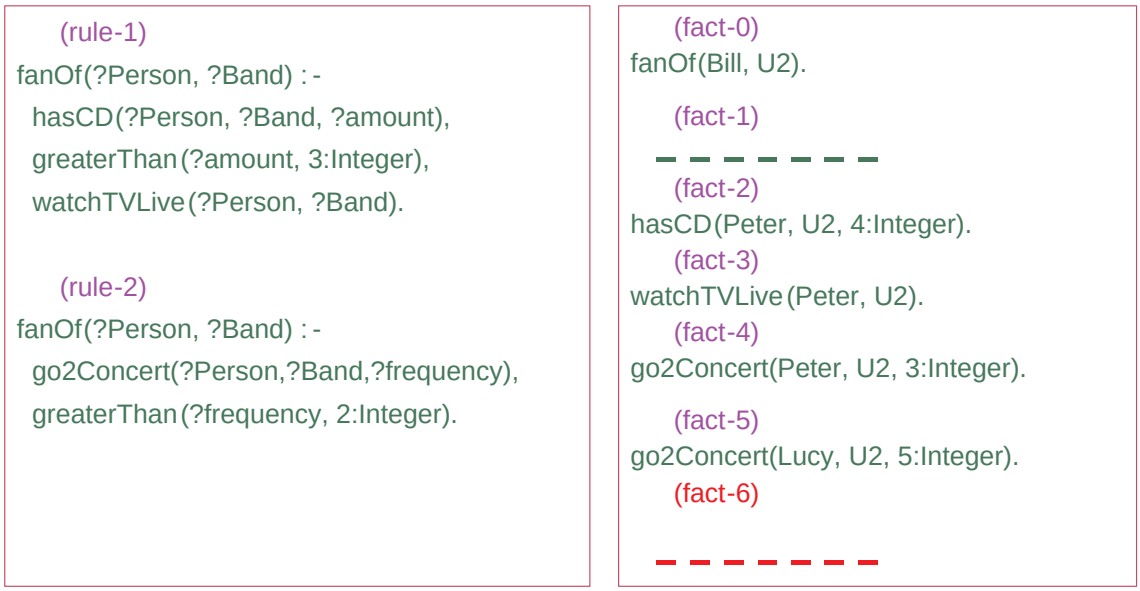

Figure 4.4: RNF Corresponding to Figure 4.3

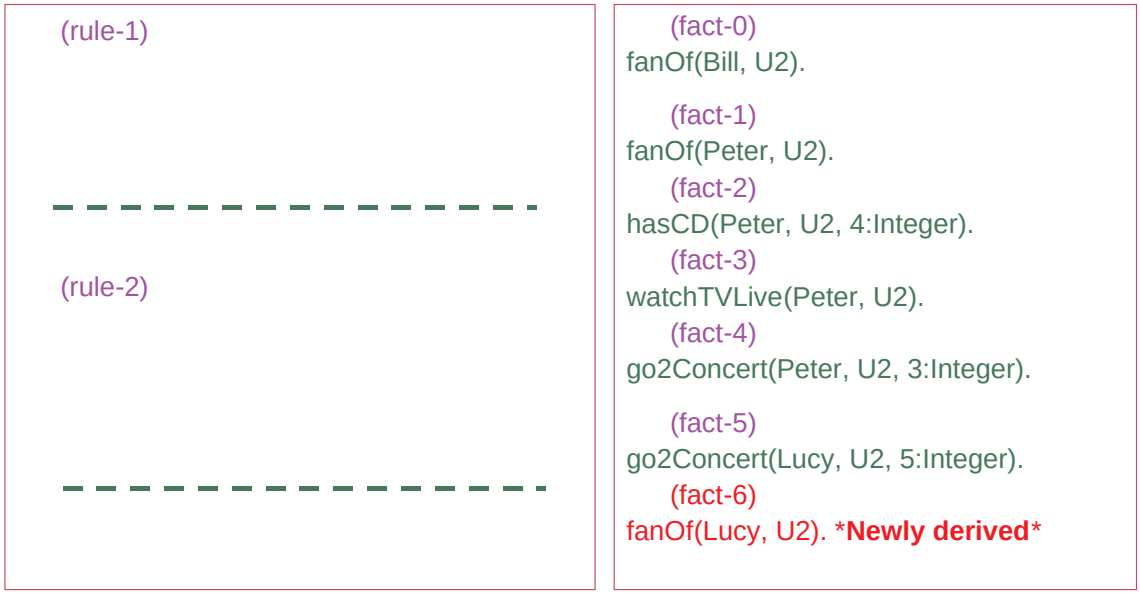

Figure 4.5: FNF Corresponding to Figure 4.3

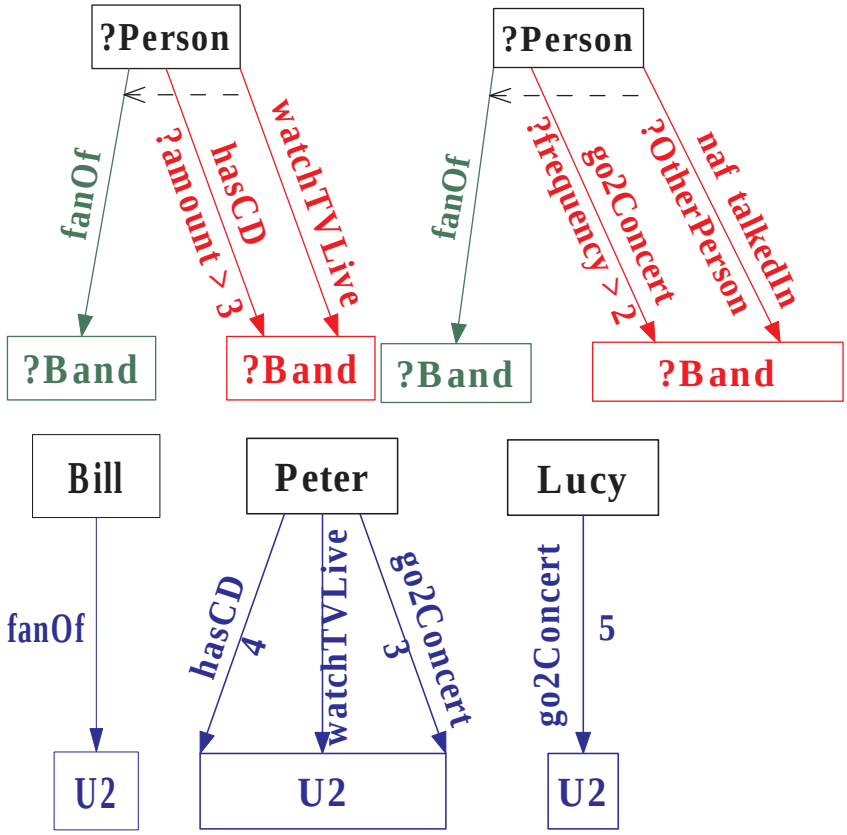

Figure 4.6: Illustration of a Published RNF

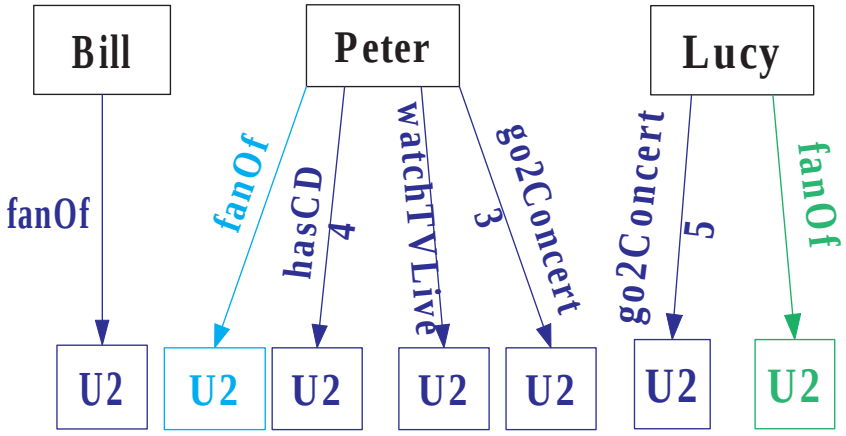

Figure 4.7: Illustration of a Published FNF

Chapter 5

FindXpRT: A Profile-Based RuleML FOAF Expert Finder

The collaboration between people through the Web infrastructure is conceived here as eCollaboration. It facilitates knowledge interchange, processing and publication. Collaboration constitutes a symmetric relationship among people where two or more people are interested in certain knowledge of each other. The crucial first step of collaboration, focused in this thesis, is expert finding, where a *co-expert (secondary expert)* is aided in finding an *expert (primary expert)* with the expertise he or she requires. While each co-expert-expert consulting is asymmetric, in a working collaboration the co-expert and expert roles are inverted for different subareas of expertise, so that an overall symmetric relationship emerges. Many projects on eCollaboration have been put forward (e.g., [7], [8], [14], and [15], [65]), one of which being our project FindXpRT (Find an eXpert via Rules and Taxonomies), conducted jointly by the National Research Council and the University of New Brunswick. An earlier version of our expert finding technology has been delivered as Teclantic (Technology transfer portal for Atlantic Canada) [17], online at teclantic.ca. Teclantic, focusing on technology transfer in Atlantic Canada, provides its co-experts with the ability to seek for other projects, or offer projects to be matched with, and found by, other users [17]. Teclantic uses a technology taxonomy for the classification of Atlantic projects [68]. Benefitting from

the refined taxonomy of technology provided by Teclantic, employers can find experts who best fit their projects, and a group of experts can collaborate through matching their similar interests.

Social networking plays a significant role in expertise finding. When searching for an expert, in any domain, we often need to rely on referrals by other experts using their social network. Social networking also helps find an expert by providing a group of people within a society and links of people outside of the society as well.

5.1 Extended FOAF Vocabulary and Translation

In this chapter we propose an approach to applying the RuleML language to metadata constituting FOAF profiles for expert finding. This section deals with the main parts of realizing our proposed approach and is divided into the following seven subsections.

We describe the structure of the facts in our knowledge base in subsection 5.1.1. In subsection 5.1.2, we propose the major component of the approach, the design of a FOAF rule vocabulary in the domain of expert finding. In subsection 5.1.3, we introduce rule execution for expert finding, i.e., computing derived FOAF properties with OO jDREW. Finally, the XSLT translation of RuleML facts to RDF, for the RDF publication of FNF facts is presented in subsection 5.1.4. Scenarios of rule-extended FOAF profiles for expert finding will be given in Section 5.2.

5.1.1 Structure of Facts in Knowledge Base

We store the information of each expert as facts in our knowledge base. All of these facts are expressed by a tree data structure. The structures of these facts need not be uniform, i.e., we do not assume a fixed schema.

As illustrated in our scenario of a fictitious expert Peter Pan (see Figure 5.2), the information about the expert normally includes his or her expertise, publications, working

location, telephone, email, and so on. When expressing the information of Peter Pan, we first take advantage of the current FOAF vocabulary, and only if the existing FOAF vocabulary cannot satisfy our needs, we propose some new vocabulary.

The detailed information on experts, down to the leaf nodes of these trees, can be extracted through queries using a top-down engine, e.g., OO jDREW TD [22].

5.1.2 Designing a FOAF Rule Vocabulary for Expert Finding

We have developed the current RDF FOAF vocabulary for both elementary and rule-derivable facts. Particularly, as we have done in the previous use cases, we extend the existing vocabulary according to the needs of the expert finder, such as seeksExpertise(Peter_Pan, ComputerScience) and publicationIn(Peter_Pan, IEEE).

We have also designed a FOAF rule specification, which does not exist in the current FOAF vocabulary. After the specification, rules can be implemented in RuleML FOAF. Principles for the FOAF rule vocabulary have been developed, e.g., relations should use the person as subject in the first argument position.

We have designed the rule vocabulary specification for music and computer science domain. In this chapter we focus designing the rule specification on expert finding. For example, ?Person1's and ?Person2's expertise match if ?Person1 seeks an expertise that ?Person2 offers (recall Example 4.2 in Chapter 4).

We distinguish different relation categories among people by identifying different relationships, namely 'knows', 'collaborates', 'collaborated' and 'consultedBy'. People's identifications, such as expert and specializedExpert, are distinguished as well. Different degrees of availability of people are also differentiated, such as atWork and onHoliday. Since the FOAF vocabulary only provides us with properties such as 'knows', we make use of our extended vocabularies, such as 'seeksAdvice' and 'atWork'.

5.1.3 Computing Derived FOAF Properties for Expert Finding

Computing derived FOAF properties involves two steps. First, we merge rules of different persons and eliminate duplicate facts and/or rules, if any, in the rulebases. Then we run the merged rulebases in OO jDREW to get the parsed rulebases with new facts.

We execute our knowledge base in OO jDREW BU and derive new facts which can be later added to the FNF. We then update the RNF by removing the derivable facts.

We query our knowledge base in OO jDREW TD to get the desired information for finding expert.

5.1.4 XSLT Translation of RuleML Facts to RDF

The knowledge base in RuleML syntax can be translated to RDF on demand via our existing XSLT translator. Obviously, only RuleML facts, not rules, obtained from the previous procedure can be mapped back to RDF syntax. Since FOAF pages are usually written in RDF syntax, it is important to enable RDF as the delivery format when there are no rules, as can be realized by applying FNF.

5.2 Scenario of Expert Finding

We give a scenario here of FindXpRT's RuleML FOAF facts and rules, which includes information about fictitious persons, and rules that identify relationships between persons and preferred actions among persons. Inspired by the Robot Composer [13], where computer programs compose music using techniques from artificial intelligence (e.g., neural networks and 'genetic algorithms' [13]), we present a scenario of establishing a collaboration between an AI expert and a Pop musician.

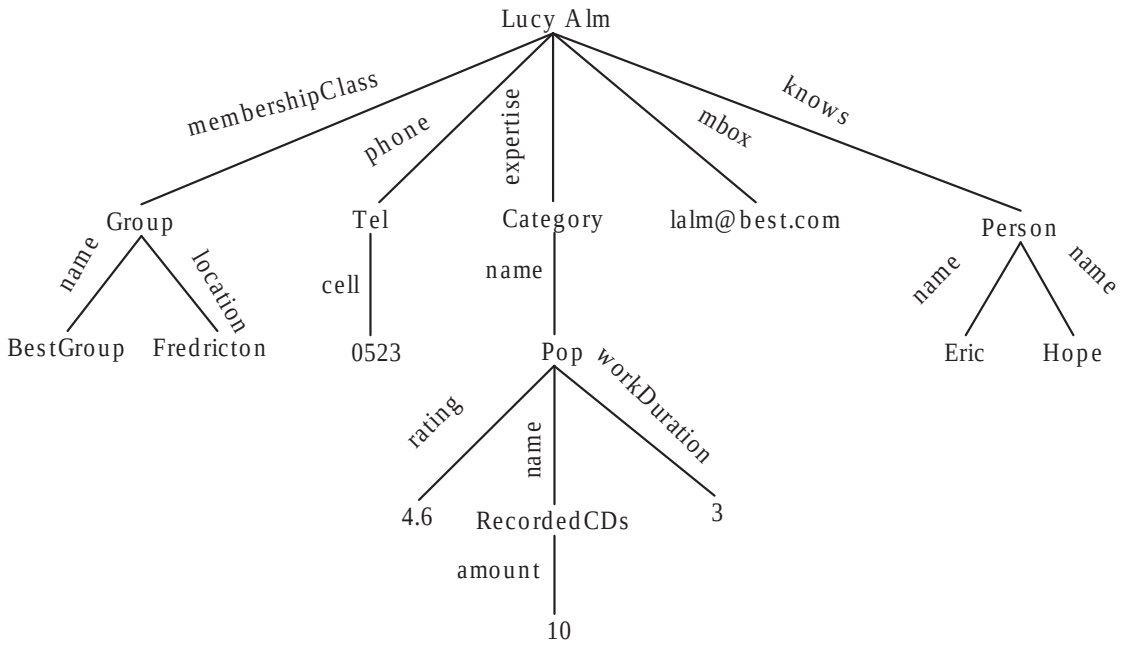

Figure 5.1: Profile of Lucy Alm (fictitious person).

5.2.1 RuleML FOAF Sample Facts

Graphical FOAF tree representations about two fictitious persons are shown in Figure 5.1 and Figure 5.2 (without namespace prefixes).

The symbolic version for Figure 5.2 in POSL can be written in the following way. Namespaces are represented as prefixes before a '.' symbol[1] for the purpose of implementation. The vocabularies with a 'foaf' namespace prefix indicates that they are borrowed from the existing FOAF vocabulary specification [33], while those with 'ex' prefixes are the vocabularies extended by, for the use in expert finding.

Profile of Lucy Alm

```
foaf.person(Lucy_Alm[
 foaf.membershipClass->Group[
  foaf.name->BestGroup;
  ex.location->Fredericton];
 ex.phone->Tel[ex.cell->0523];
 ex.expertise->Category[
  foaf.name->Pop[
   ex.rating->4.5;
```

[1]This is because the symbol ':', commonly used to express namespaces, is reserved in OO jDREW as a type infix for separating terms from their order-sorted types.

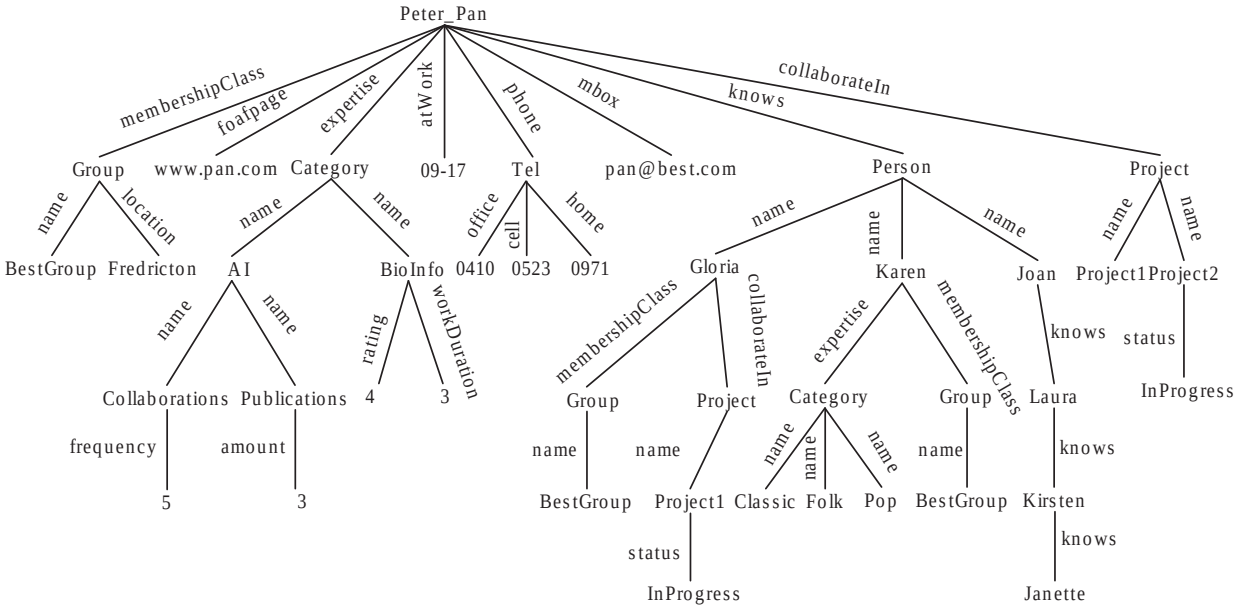

Figure 5.2: Profile of Peter Pan (fictitious person).

```
   foaf.name->RecordedCDs[
    ex.amount->10];
   ex.workDuration->3]];
 foaf.mbox->"lalm@best.com";
 foaf.knows->Person[
  foaf.name->Eric;
  foaf.name->Hope]]).
```

RuleML serialization of Lucy Alm's profile is shown as follows:

```
<Assert>
   <And>
      <Atom closure="universal">
         <Rel>foaf.person</Rel>
         <Cterm>
            <Ctor>Lucy_Alm</Ctor>
            <slot>
               <Ind>foaf.membershipClass</Ind>
               <Cterm>
                  <Ctor>Group</Ctor>
                  <slot>
                     <Ind>foaf.name</Ind>
                     <Ind>BestGroup</Ind>
                  </slot>
                  <slot>
                     <Ind>ex.location</Ind>
                     <Ind>Fredericton</Ind>
                  </slot>
               </Cterm>
            </slot>
            <slot>
```

```
   <Ind>ex.phone</Ind>
   <Cterm>
      <Ctor>Tel</Ctor>
      <slot>
         <Ind>ex.cell</Ind>
         <Ind>0523</Ind>
      </slot>
   </Cterm>
</slot>
<slot>
   <Ind>ex.expertise</Ind>
   <Cterm>
      <Ctor>Category</Ctor>
      <slot>
         <Ind>foaf.name</Ind>
         <Cterm>
            <Ctor>Pop</Ctor>
            <slot>
               <Ind>foaf.name</Ind>
               <Cterm>
                  <Ctor>RecordedCDs</Ctor>
                  <slot>
                     <Ind>ex.amount</Ind>
                     <Ind>10</Ind>
                  </slot>
               </Cterm>
            </slot>
            <slot>
               <Ind>ex.rating</Ind>
               <Ind>4.5</Ind>
            </slot>
            <slot>
               <Ind>ex.workDuration</Ind>
               <Ind>3</Ind>
            </slot>
         </Cterm>
      </slot>
   </Cterm>
</slot>
<slot>
   <Ind>foaf.mbox</Ind>
   <Ind>lalm@best.com</Ind>
</slot>
<slot>
   <Ind>foaf.knows</Ind>
   <Cterm>
      <Ctor>Person</Ctor>
      <slot>
         <Ind>foaf.name</Ind>
         <Ind>Eric</Ind>
      </slot>
      <slot>
         <Ind>foaf.name</Ind>
         <Ind>Hope</Ind>
      </slot>
```

```
              </Cterm>
            </slot>
          </Cterm>
      </Atom>
   </And>
</Assert>
```

Profile of Peter Pan

```
person(Peter_Pan[
 foaf.membershipClass->Group[
             foaf.name->BestGroup;
             ex.location->Fredericton];
 foaf.foafpage->"www.pan.com";
 ex.expertise->Category[
   foaf.name->AI[
     foaf.name->Collaboration[
      ex.frequency->5];
     foaf.name->Publication[
      ex.amount->3]];
   foaf.name->BioInfo[
    ex.rating->4;
    ex.workDuration->3]];
 ex.atWork->"09:00-17:00";
 ex.phone->Tel[
            ex.office->0410;
            ex.cell->0523;
            ex.home->0971];
 foaf.mbox->"pan@best.com";
 foaf.knows->Person[
   foaf.name->
    Gloria[
     foaf.membershipClass->Group[
      foaf.name->b]BestGroup];
     ex.collaborateIn->Project[
      foaf.name->Project1[
        ex.status->InProgress]];
   foaf.name->
    Karen[
     ex.expertise->Category[
       foaf.name->Classic;
       foaf.name->Folk;
       foaf.name->Pop];
     foaf.membershipClass->Group[
       foaf.name->BestGroup]];
   foaf.name->
    Joan[
     foaf.knows->Laura[
      foaf.knows->Kirsten]]];
 ex.collaborateIn->Project[
   foaf.name->Project1;
   foaf.name->Project2[
    ex.status->InProgress]]]).
```

5.2.2 FindXpRT's Top-Level Rule Systems

We provide rules systems to illustrate FindXpRT's method of expert finding for eCollaboration. We first represent rules for finding potential experts to collaborate with. Then we provide rules for "the selected" expert to make decisions on the collaboration. Next, we introduce the rules for preference to collaboration mode.

Two flowcharts, Figure 5.3 and Figure 5.4, are first given to illustrate two sample rule systems, which are followed by the symbolic versions of the same rule systems. The two rule systems illustrated in these flowcharts have been incorporated in rule sets, where, in declarative programming, unlike flowcharts, the order of rules does not matter in rule execution.

5.2.2.1 Rule System for Expert Finding

This rule system is user-centric and is used by secondary expert to find a primary expert to collaborate with. The flowchart showing these rules are shown in Figure 5.3. The expertise ?X and ?Y represented in this paper ranges over the taxonomy for technology transfer of Teclantic.ca.

When a music expert queries for a Computer Science expert in an area, the rule system accesses the experts' profiles. It first checks, in the profile a candidate expert, if he/she meets the qualification of offering the required expertise and if he/she is a person different from the music expert. Rating is also verified of being above some satisfiable threshold. FindXpRT scale ranks from 1 to 5, where 3.0 is considered as a acceptable rating boundary. In Figure 5.3, we focus on persons in that same cooperation (group). If the music expert and the Computer Science expert have collaborated on the same project, which is still in progress, the process ends because they are already collaborating. Otherwise, if the Computer Science expert's offered expertise does not match the music expert's sought expertise according to our taxonomic similarity measure [68] for a user threshold ?T, FindXpRT cannot pair them up for consultation. Otherwise, when this Computer Science expert is not currently involved in any project, the FindXpRT calls another rule system as a subroutine, namely

FindXpRT

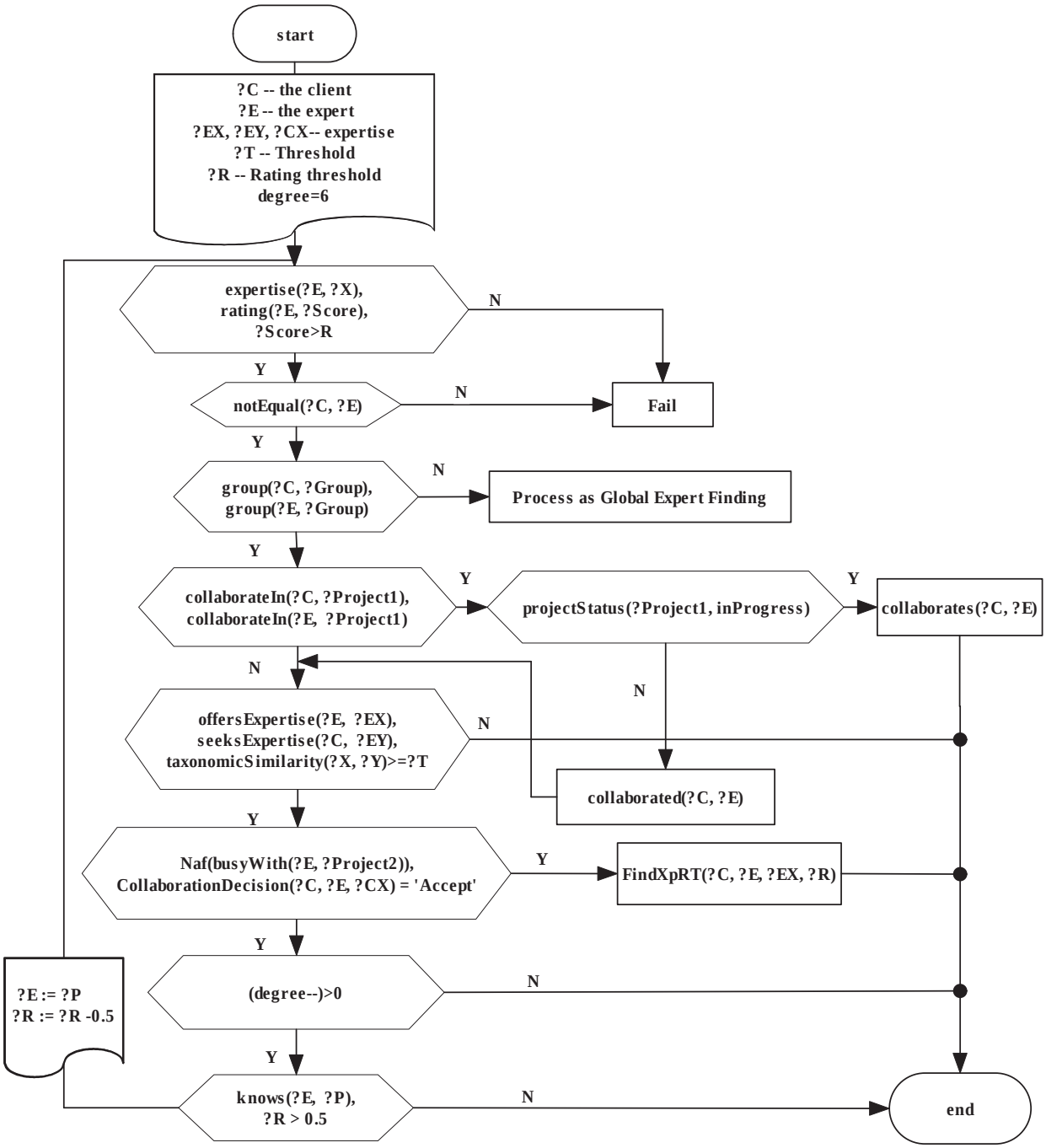

Figure 5.3: Scenario of Expert Finding.

CollaborationDecision. If the result of the CollaborationDecision rule system is 'Accept', then the FindXpRT answers the music expert by providing this Computer Science expert. However, when the Computer Science expert turns out to be unavailable, or the result of the CollaborationDecision rule system is not 'Accept', then this Computer Science expert may still refer the music expert to other potential Computer Science experts in his/her social network[2]. Every time the expert refers to another expert, the predefined rating threshold decreases by threshold decrement (0.5 in our FindXpRT). According to the "six degrees" concept [34], there would be at most six such rounds of referrals. The variable ***degree*** in Figure 5.3 is thus assigned the value ***6***.

The symbolic version of Figure 5.3 is shown below[3]. Note that we have the convention of naming variable: E implies Computer Science expert and C implies the music expert; X expresses offered expertise while Y expresses sought expertise.

```
Input:  Music expert ?C, ?E, Computer Science expert's offered
expertise ?EX, music expert's sought expertise ?CY, and
music expert's offered expertise ?CX, rating threshold ?R,
pre-assigned value 6 for degree.

Output:  Assert FindXpRT(?C,?E) when finding an
appropriate Computer Science expert ?E.

 Step 1:

 IF expertise(?E,?X),
    rating(?E, ?Score),
    ?Score > R
 {
   IF notEqual(?C,?E)
   {
       GOTO Step 2
   }
 }

Step 2:

 IF (group(?C,?Group), group(?E,?Group))
 {
     GOTO Step 3
 }
```

[2]For simplicity (avoiding non-determinism on this level), we assume that each expert can refer to at most to a single other expert.

[3]Negation as failure is written as naf(...).

```
 ELSE
 {
    Process as Global Expert Finding
 }

Step 3:

 IF (collaborateIn(?C,?Project1),
     collaborateIn(?E,?Project1))
 {
     GOTO Step 4
 }
 ELSE GOTO Step 5

Step 4:

 IF projectStatus(?Project1,InProgress)
 {
    collaborates(?C,?E)
 }
 ELSE
 {
    collaborated(?C,?E)
    GOTO Step 5
 }

Step 5:

 IF (offersExpertise(?E,?X),
     seeksExpertise(?C,?Y),
     taxonomicSimilarity(?X,?Y)>=?T)

 {
    GOTO Step 6
 }

Step 6:

 IF (naf(busyWith(?E,?Project2)),
     CollaborationDecision(?C,?E,?X)
     ='Accept')
 {
    Assert FindXpRT(?C,?E)
 }
 ELSE GOTO Step 7

Step 7:

 loop:  while ((degree--)&& (?R > 3.5))>0
        IF (knows(?E,?P),
            offersExpertise(?P,?X))
        {
                ?E := ?P
                ?R := ?R - 0.5
                GOTO Step 1
        }
```

Pseudo-Code 1: Pseudo-Code for Figure 5.3

We give here a match-making example on the basis of the two profiles, of persons Lucy Alm and Peter Pan. Suppose Lucy Alm is the music expert while Peter Pan is the Computer Science expert. Lucy Alm seeks for Logic Programming as the required expertise, with a similarity threshold 0.8, from an expert. Peter Pan first satisfies the criteria that he has the expertise. Then he meets the requirement that he is from the same company, namely BestGroup, as Lucy Alm. The next rule checks if they are not collaborating with each other, succeeding in our case. Peter Pan then meets the condition that he offers Logic Programming as his expertise, with a taxonomic similarity of 1.0, greater or equal 0.8, as Lucy Alm required. The next step is to see if Peter Pan is currently busy with some project. Since Peter is not having any project at this time, the rule system calls another rule system CollaborationDecision. When CollaborationDecision gives the result, in our case 'Accept', a new fact, consultedBy(Lucy_Alm, Peter_Pan), is asserted.

5.2.2.2 Rule System for Decision Making on Collaboration

Rules in this rule system is expert-centric, and helps an expert who is "selected" to collaborate with an arbitrary person to make decisions on participation. Different persons can have different precondition for making a decision. Therefore, in Figure 5.4, the previously open expert, expressed by a variable ?E, is here a constant, Peter_Pan, these rules are local and attached to this specific person.

Figure 5.4 illustrates the scenario of how Peter Pan makes a decision on request by an arbitrary person for participating in a project. The rule system first gets the preferred phone number via the phonePreference rule set. When Peter Pan receives this request, he accesses this music expert, e.g. Lucy Alm's profile. He first checks if this person is from the BestGroup cooperation within the Fredericton region, as he declines all the request outside his company's Fredericton branch. Moreover, Peter Pan is only interested in collaborating with co-experts in Pop music. Peter Pan has criteria on the number of a collaborator's RecordedCDs, period of working and rating (ranked by colleagues with 5 as the best mark). Peter Pan only decides to collaborate when the co-expert meets all of these criteria. After

making his decision, Peter Pan contacts people in two ways, by phone or email, depending on this person being within his social network.

The symbolic version of Figure 5.4 for a specific person Peter Pan is shown below:

```
Input:  Music expert ?C, expertise ?X.
Output:  Accept or Decline the request.

Step 1:

 phonePreference(Peter_Pan,?Tel1),
 call(?C, ?Tel1)

Step 2:

 IF location(?C,Fredericton)
 {
    GOTO Step 3
 }
 ELSE
 {
    Decline the request
 }

Step 3:

 IF offersExpertise(?C,?X)
 {
    GOTO Step 4
 }
 ELSE
 {
    Decline the request
 }

Step 4:

 IF (RecordedCDs(?C,?Amount),
     greaterThan(?Amount,8),
     workDuration(?C,?Year),
     greaterThan(?Year,2.0))
 {
    GOTO Step 5
 }
 ELSE
 {
    Decline the request
 }

Step 5:

 IF rating(?X,?Score) AND
    greaterThan(?Score,4.5)
 {
    Accept the request
    GOTO Step 6
```

CollaborationDecision

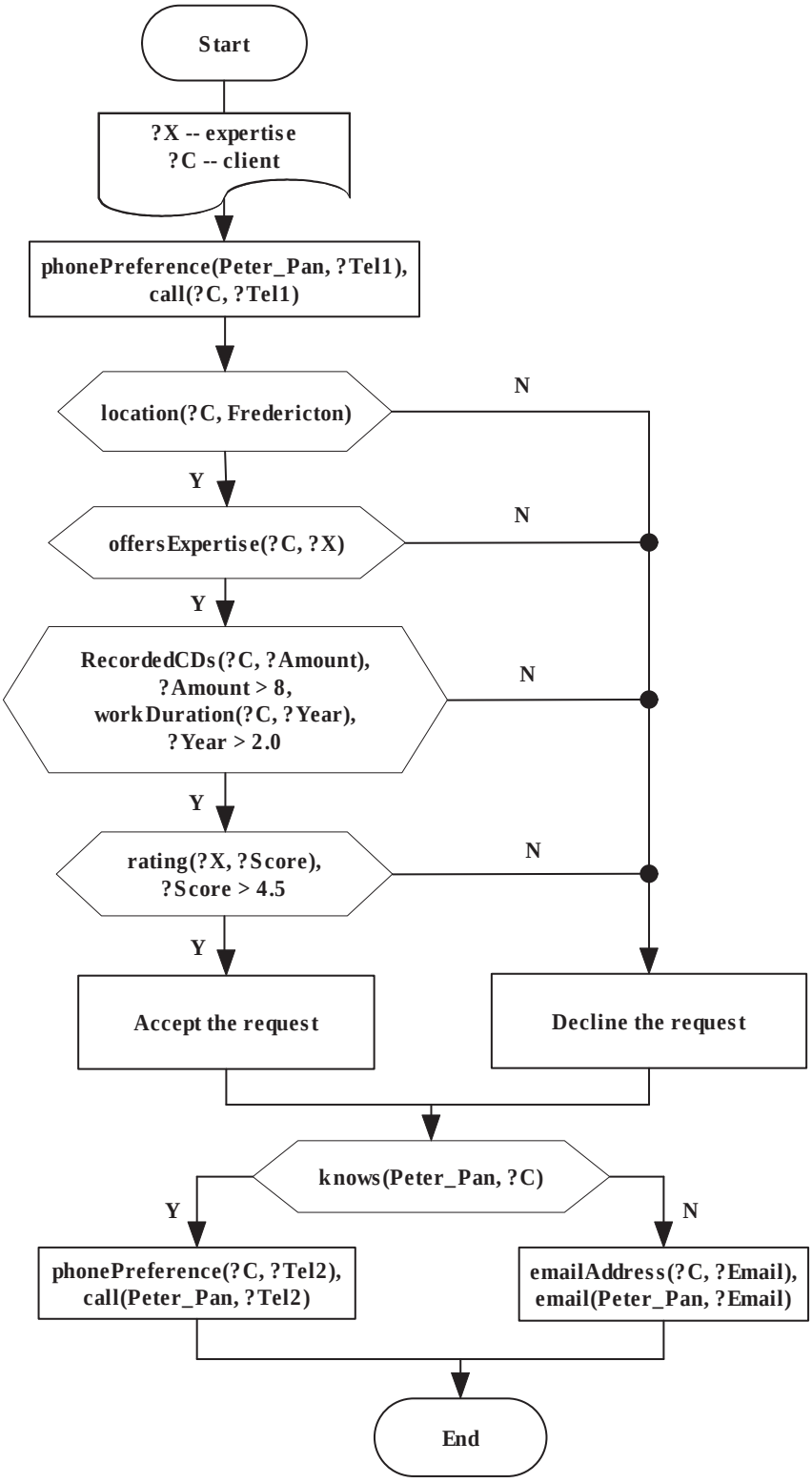

Figure 5.4: Scenario of Decision Making on Possible Collaboration.

```
 }
 ELSE
 {
    Decline the request
 }

Step 6:

 IF knows(Peter_Pan,?C)
 {
    phonePreference(?C,?Tel2),
    call(Peter_Pan,?Tel2)
 }
 ELSE
 {
    emailAddress(?C,?Email),
    email(Peter_Pan, ?Email)
 }
```

Pseudo-Code 2: Pseudo-Code for Figure 5.4

As described previously, Peter Pan is chosen by Lucy Alm as desired collaborator and it calls CollaborationDecision. Peter Pan receives the message in his preferred way providing by the rule set phonePreference. This rule system is for Peter Pan to decide whether he wants to accept this request. Peter Pan wants his prospective collaborator to be in Fredericton, where Lucy Alm satisfies. Lucy Alm also satisfies his criteria as she not only offers expertise in Pop music, but also her RecordedCDs exceed eight, and she has a three-year working experience, which exceeds the two-year's minimum, a 4.5 rating which also exceeds 4 as the minimum. Therefore, Peter Pan accepts the request. The means for him accepting the request is via e-mail since Peter Pan does not know Lucy Alm.

5.2.2.3 Rules for Specifying the Collaboration Mode

This rule system represents preferences of co-experts regarding the collaboration mode. With these rules, a co-expert can collaborate face to face with an expert, or by telephone, or through the Web, according to restrictions on the date, time and distance. We represent these rules first in POSL and then also serialize the first two rules in RuleML[4].

[4]In order to run these rules in OO jDREW, a "`:Integer`" type will be given to all integer constants.

```
(POSL-1) collaborationMode(F2F,?Date,?Time,?Distance):-
    holiday(?Date),
    greaterThanOrEqual(?Time,10:Integer),
    lessThanOrEqual(?Time,16:Integer),
    lessThan(?Distance,20:Integer).
(POSL-2) collaborationMode(F2F,?Date,?Time,?Distance):-
    naf(holiday(?Date)),
    greaterThanOrEqual(?Time,16:Integer),
    lessThanOrEqual(?Time,20:Integer),
    lessThan(?Distance,20:Integer).
(POSL-3) collaborationMode(Tel,?Date,?Time,?Distance):-
    holiday(?Date),
    greaterThanOrEqual(?Time,09:Integer),
    lessThanOrEqual(?Time,22:Integer),
    greaterThanOrEqual(?Distance,20:Integer),
    lessThan(?Distance,100:Integer).
(POSL-4) collaborationMode(Tel,?Date,?Time,?Distance):-
    naf(holiday(?Date)),
    greaterThanOrEqual(?Time,17:Integer),
    lessThanOrEqual(?Time,22:Integer),
    greaterThanOrEqual(?Distance,20:Integer),
    lessThan(?Distance,100:Integer).
(POSL-5) collaborationMode(Web,?Date,?Time,?Distance):-
    holiday(?Date),
    greaterThanOrEqual(?Time,09:Integer),
    lessThanOrEqual(?Time,22:Integer),
    greaterThanOrEqual(?Distance,100:Integer).
(POSL-6) collaborationMode(Web,?Date,?Time,?Distance):-
    naf(holiday(?Date)),
    greaterThanOrEqual(?Time,17:Integer),
    lessThanOrEqual(?Time,22:Integer),
    greaterThanOrEqual(?Distance,100:Integer).
```

Rule (POSL-1) expresses that if it is a holiday, the time is between 10:00 and 16:00 o'clock, and the distance to the collaboration place is less than 20 miles, then the preferred collaboration mode is face to face.

Rule (POSL-2) represents that if it is not a holiday, the time is between 16:00 and 20:00, and the distance to the collaboration place is less than 20 miles, then the preferred collaboration mode is also face to face.

Rule (POSL-3) expresses that if it is a holiday, the time is between 09:00 and 22:00, and the distance to the collaboration place is between 20 miles and 100 miles, then the preferred collaboration mode is by telephone.

Rule (POSL-4) represents that if it is not a holiday, the time is between 17:00 and 22:00, and the distance to the collaboration place is between 20 miles and 100 miles, then the preferred

collaboration mode is also by telephone.

Rule (POSL-5) expresses that if it is a holiday, the time is between 09:00 and 22:00, and the distance to the collaboration place is greater than 100 miles, then the preferred collaboration mode is the Web.

Rule (POSL-6) expresses that if it is not a holiday, the time is between 17:00 and 22:00, and the distance to the collaboration place is greater than 100 miles, then the preferred collaboration mode is also the Web.

To exemplify XML serialization, the rules (POSL-1) and (POSL-2) are marked up as two 'Implies' elements in RuleML 0.9[5] as follows.

```
<Assert>
   <And mapClosure="universal">
      <Implies>
         <And>
            <Atom>
               <Rel>holiday</Rel>
               <Var>Date</Var>
            </Atom>
            <Atom>
               <Rel>greaterThanOrEqual</Rel>
               <Var>Time</Var>
               <Ind type="Integer">10</Ind>
            </Atom>
            <Atom>
               <Rel>lessThanOrEqual</Rel>
               <Var>Time</Var>
               <Ind type="Integer">16</Ind>
            </Atom>
            <Atom>
               <Rel>lessThan</Rel>
               <Var>Distance</Var>
               <Ind type="Integer">20</Ind>
            </Atom>
         </And>
         <Atom>
            <Rel>collaborationMode</Rel>
            <Ind>F2F</Ind>
            <Var>Date</Var>
            <Var>Time</Var>
            <Var>Distance</Var>
         </Atom>
      </Implies>
      <Implies>
         <And>
            <Naf>
```

[5]http://www.ruleml.org/0.9/

```
            <Atom>
               <Rel>holiday</Rel>
               <Var>Date</Var>
            </Atom>
         </Naf>
         <Atom>
            <Rel>greaterThanOrEqual</Rel>
            <Var>Time</Var>
            <Ind type="Integer">16</Ind>
         </Atom>
         <Atom>
            <Rel>lessThanOrEqual</Rel>
            <Var>Time</Var>
            <Ind type="Integer">20</Ind>
         </Atom>
         <Atom>
            <Rel>lessThan</Rel>
            <Var>Distance</Var>
            <Ind type="Integer">20</Ind>
         </Atom>
      </And>
      <Atom>
         <Rel>collaborationMode</Rel>
         <Ind>F2F</Ind>
         <Var>Date</Var>
         <Var>Time</Var>
         <Var>Distance</Var>
      </Atom>
   </Implies>
</And>
</Assert>
```

Chapter 6

The FindXpRT Benchmark for Computer Science and Music Profiles

In this chapter, we describe how to use FindXpRT, its functionality (e.g., how well it can handle different situations), its execution speed, as well as its advantages and disadvantages. Computer Science and music are chosen as expertise domains here for demonstrating FindXpRT and proposed as a benchmark for expert finding in general.

The proposed benchmark consists of a suite for testing expert-finding systems against 10 characteristic expert profiles (cf. section 6.2.1), exemplified with our FindXpRT system tested against Computer Science and music profiles. Through these experiments, we are able to show how FindXpRT can help its users to find appropriate experts under different circumstances. Experimental results show that FindXpRT can find expert(s) not only via direct matches, but also via appropriate expert referrals.

Figure 6.1 shows how FindXpRT works to help a user to find a collaborator. First, the FindXpRT user posts a query to our FindXpRT system. Our FindXpRT system then automatically applies the corresponding rules (including expert rule profiles as well as FindXpRT rule systems and rule sets) to the facts (including expert fact profiles as well as other agent profiles containing centrally stored data). After processing facts with rules, the result of our FindXpRT system may turn out to be either a success or a failure. When it succeeds,

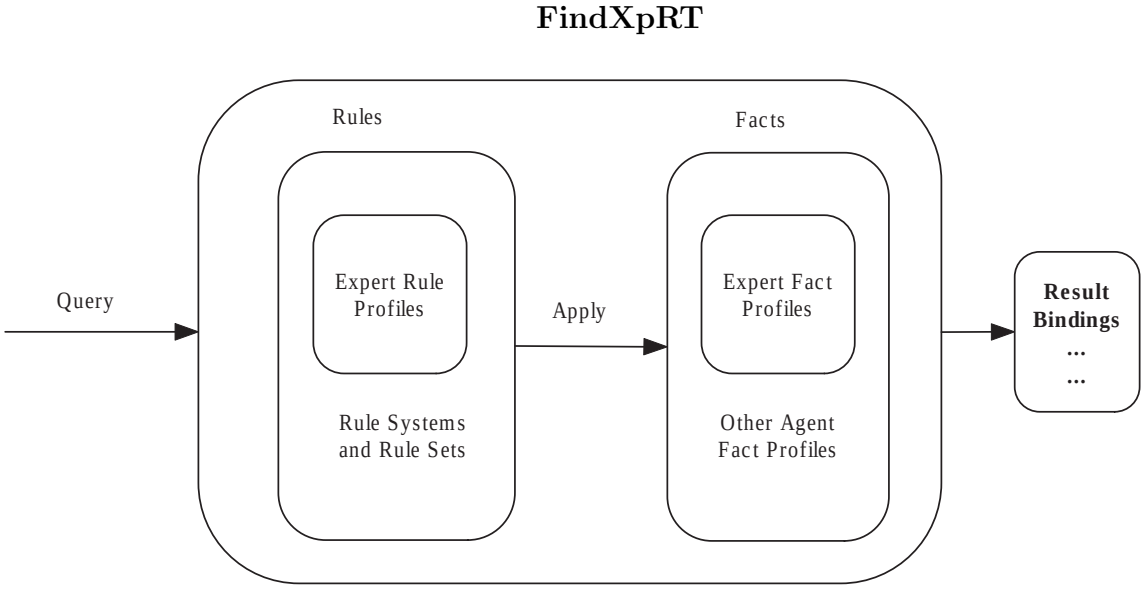

Figure 6.1: FindXpRT System

FindXpRT shows in result bindings the matched expert to its user. Otherwise, FindXpRT notifies its user that there is "No Solution" for him/her by showing empty result bindings.

The FindXpRT system is built on the top of a rule engine, OO jDREW. We explain here how to execute the FindXpRT system according to the OO jDREW TD user interface (cf. the screen shot in Figure 6.2). The uppermost box is where FindXpRT users post their queries for finding an appropriate expert. The FindXpRT system is then executed after user clicks "Issue Query" button. After the FindXpRT computation is completed, result bindings will be shown in the box on the righthand side of FindXpRT window. In the left box of the FindXpRT window, it shows the trace of how the expert is found by our FindXpRT system. However, if FindXpRT system fails to find an expert for its user, the box for showing result bindings will remain blank and the box for showing trace remains "No Solution".

The way to query our FindXpRT system is:

```
FindXpRT(?CoExpert, ?Expert, ?ReferredExpert, ?CoExpertise,
?RatingThreshold, ?UltimateRating, ?Degree)
```

where the meaning of the seven arguments is as follows:

```
?CoExpert: the secondary expert, here the music expert
?Expert: the primary expert, here the Computer Science expert
?ReferredExpert: the referred primary expert,
                 here the Computer Science expert
?CoExpertise: the expertise offered by the ?CoExpert
?Expertise: the expertise offered by the ?Expert
?RatingThreshold: the initial rating threshold
?UltimateRating: the ultimate rating of the referred expert
?Degree: the counter for controlling referral rounds
```

Our FindXpRT system applies rule-based social networking to expert finding. Inspired by Teclantic, the assumed business-service model is bartering-like [54] exchange of expertise between an expert and a co-expert, where the latter initiates the search using our system. The co-expert acts as a buyer in this bartering-like exchange, who seeks an expert with certain expertise. The expert, on the other hand, acts as a seller, who offers certain expertise sought by the co-expert. Furthermore, for establishing a collaboration, the co-expert is required to offer some secondary expertise in collaboration with that expert. Therefore, this bartering-like exchange between co-expert and expert is partially asymmetric.

6.1 Experimental Results under Typical Testing

The results of FindXpRT system are provided in this section, varying with different loaded profiles as well as queries. We consider the following examples to demonstrate each typical case in the FindXpRT system.

6.1.1 Find an Expert via Direct Search

Example 1: *A music specialist, Lucy, acting as the co-expert finds a Computer Science specialist Peter. Peter, who is considered as the expert, accepts Lucy's request.*

In **Example 1.a**, we set degree to 0:Integer, which means the FindXpRT user does not allow expert referral, and in **Example 1.b**, we assign 6:Integer to degree, which means the referral is allowed, but not needed in this case.

Example 1.a: Referral not Allowed

The degree variable is set to 0:Integer, which means Lucy wants to find an expert without referral. With the two expert profiles loaded in OO jDREW (see Figure 5.3 and Figure 5.4), a direct search between Lucy and Peter can be made. Therefore, Lucy, with Pop music as her expertise, can query the system for a Computer Science expert with expertise in Logic Programming and the system finds Peter for her with 0 rounds of referral, as shown in Figure 6.2.

Query (against Lucy's and Peter's profiles in Appendix A):

FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, PopMusic, LogicProgramming, 4.5:Real, ?RatingSought, 0:Integer)

Result Bindings:

```
?CSXpRT = Peter
?RatingSought = 4.5:Real
?ReferredXpRT = Peter
```

Example 1.b: Referral Allowed, But Not Needed

The degree variable is set to 6:Integer (according to six degree of separation), which means Lucy wants to find an expert no matter via direct search or via referral by intermediate experts, as shown in Figure 6.3. On the other hand, the secondary and primary expertise variables are left unspecified, hence will be bound to Lucy's and Peter's expertise, respectively.

Query (against Lucy's and Peter's profiles in Appendix A):

FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, ?LucyExpertise, ?CSXpRTise, 4.5:Real, ?RatingSought, 6:Integer)

Result Bindings:

```
?CSXpRT = Peter
?RatingSought = 4.5:Real
?CSXpRTise = LogicProgramming
?ReferredXpRT = Peter
?LucyExpertise = PopMusic
```

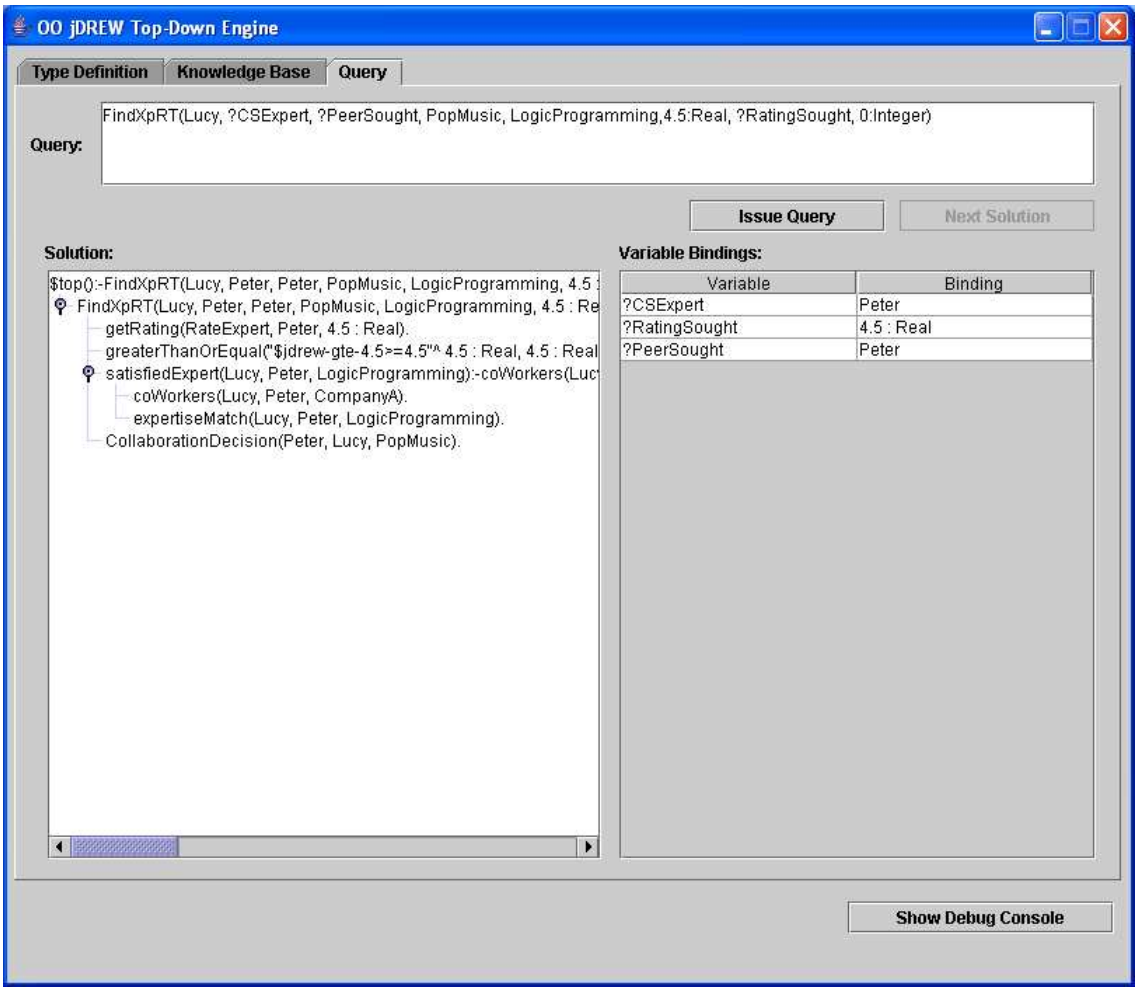

Figure 6.2: Screen Shot for Example 1.a

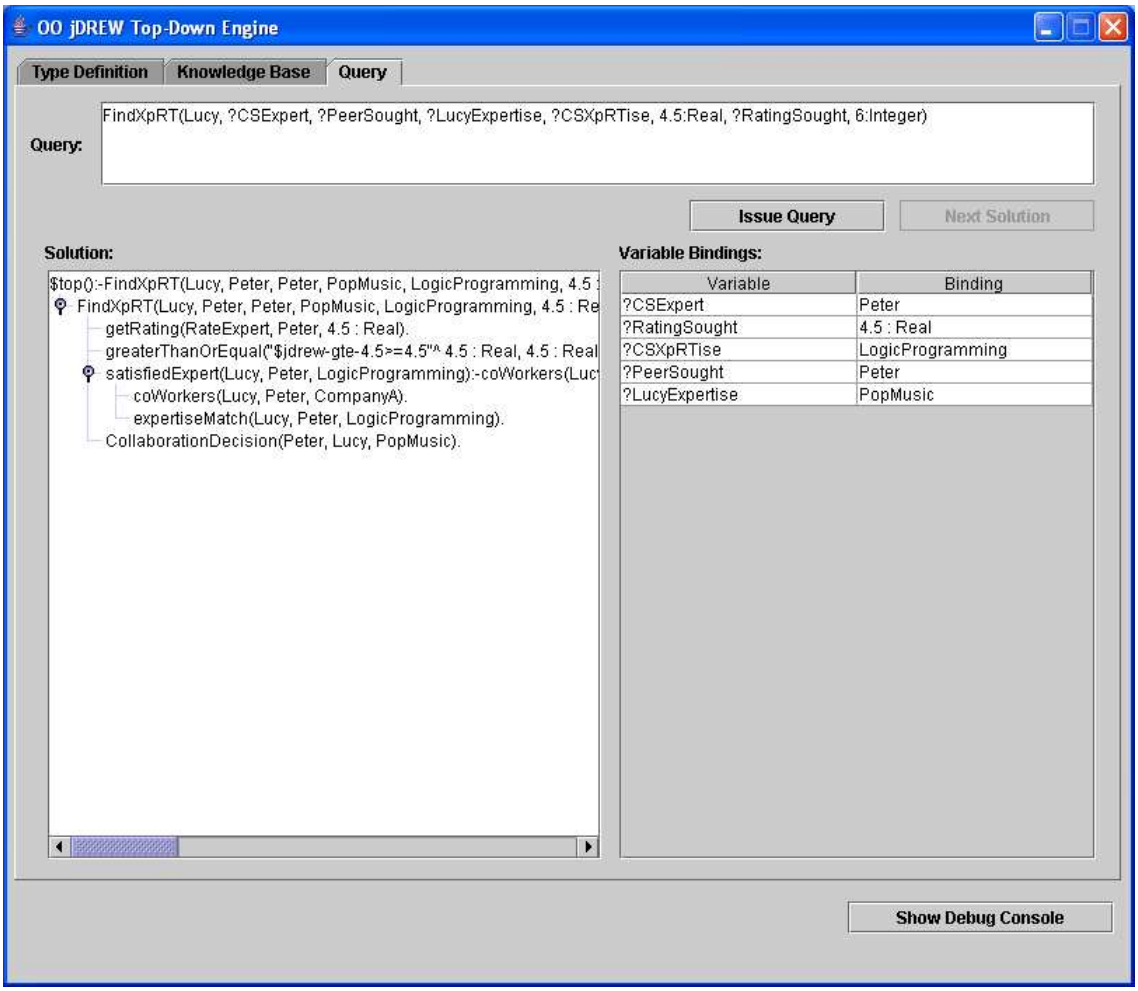

Figure 6.3: Screen Shot for Example 1.b

6.1.2 Find an Expert via One Round of Referral

Example 2: *According to Lucy's criteria as well as the Computer Science expert profiles loaded in the rule engine, the system cannot find a direct match for her. Therefore, an intermediate expert referral is provided.*

In Example 2.a, we set degree to 0:Integer, which causes failure since there is no direct match the between the FindXpRT user and the experts, and no expert referral is allowed. Two examples in Example 2.b both allow expert referral, with degree set to 6:Integer. Example 2.b.1 shows that the directly matched expert declined to collaborate and referral then took place, while Example 2.b.1 shows the referral took place because of the unavailability of the expert.

Example 2.a: Failure Caused by not Allowing Referral

Lucy discovers Annie as an expert. However, Annie wants to collaborate with those who have Classic Music expertise, while Lucy offers Pop Music as her expertise. Therefore, Annie does not accept Lucy's collaboration request: degree 0:Integer -> failure, as shown in Figure 6.4.

Query (against Lucy's, Annie's and Hart's profiles in Appendix A):
FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, ?LucyExpertise, ?CSXpRTise, 4.5:Real, ?RatingSought, 0:Integer)

Result Bindings:

```
No solutions
```

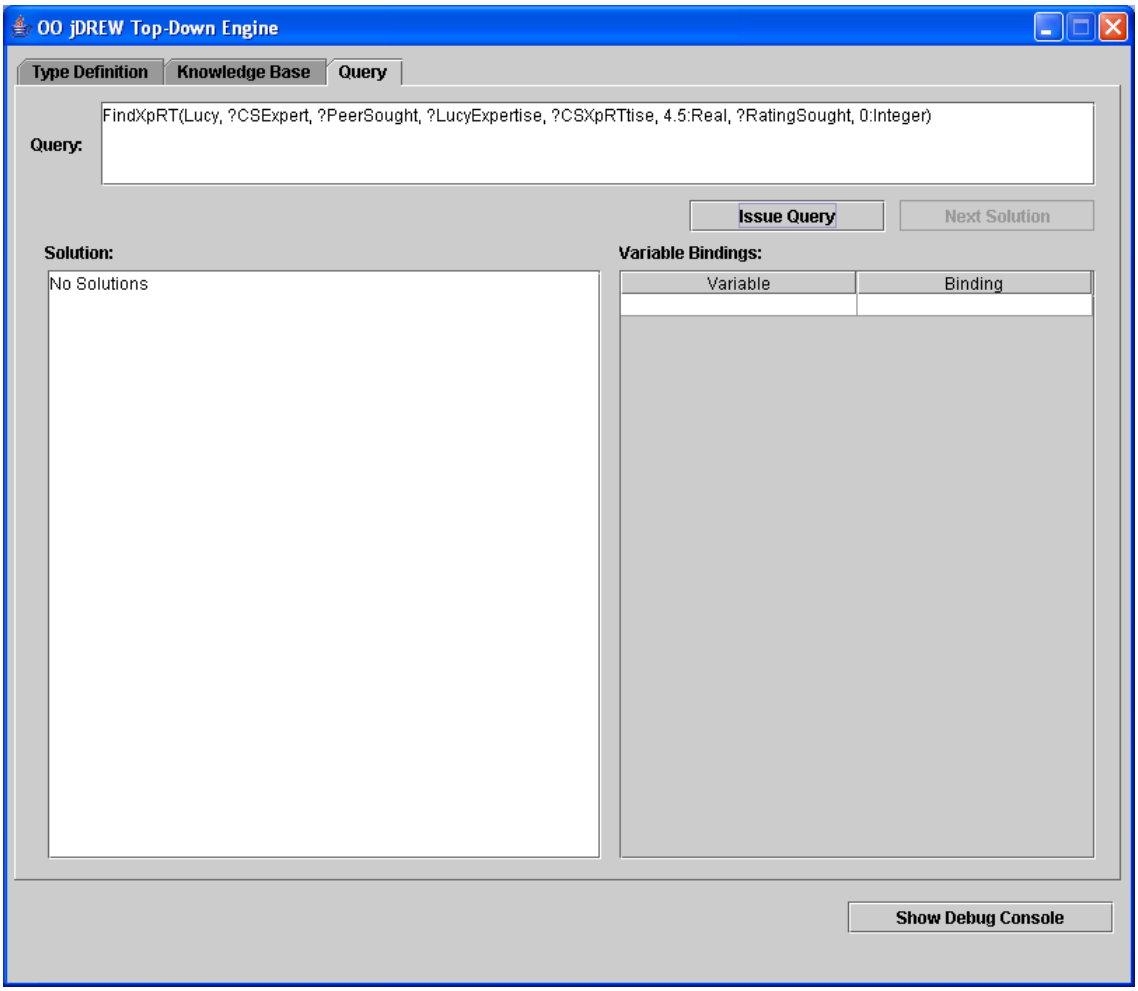

Figure 6.4: Screen Shot for Example 2.a

Example 2.b: Success in One Round of Referral

Two subexamples are involved here: Direct search fails because the Computer Science expert declines the collaboration request or because s/he is not available.

Example b.1: Direct Match Fails Because Computer Science Expert Declines the Collaboration Request

Lucy discovers Annie. However, Annie wants to collaborate with those who have Classic Music expertise, while Lucy offers Pop Music as her expertise. Therefore, Annie does not accept Lucy's collaboration request and she refers to Hart. Hart accepts. The result is shown in Figure 6.5.

Query (against Lucy's, Annie's and Hart's profiles in Appendix A):

FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, ?LucyExpertise, ?CSXpRTise, 4.5:Real, ?RatingSought, 6:Integer)

Result Bindings:

```
?CSXpRT = Annie
?RatingSought = 4.0:Real
?CSXpRTise = LogicProgramming
?ReferredXpRT = Hart
?LucyExpertise = PopMusic
```

Example b.2: Direct Search Fails Because Computer Science Expert Declines since not Available

Lucy discovers Julia. Julia is busy with some other project(s) and she refers to Hart. Hart accepts. The result is shown in Figure 6.6.

Query (against Lucy's, Julia's and Hart's profiles in Appendix A):

FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, ?LucyExpertise, ?CSXpRTise, 4.5:Real, ?RatingSought, 6:Integer)

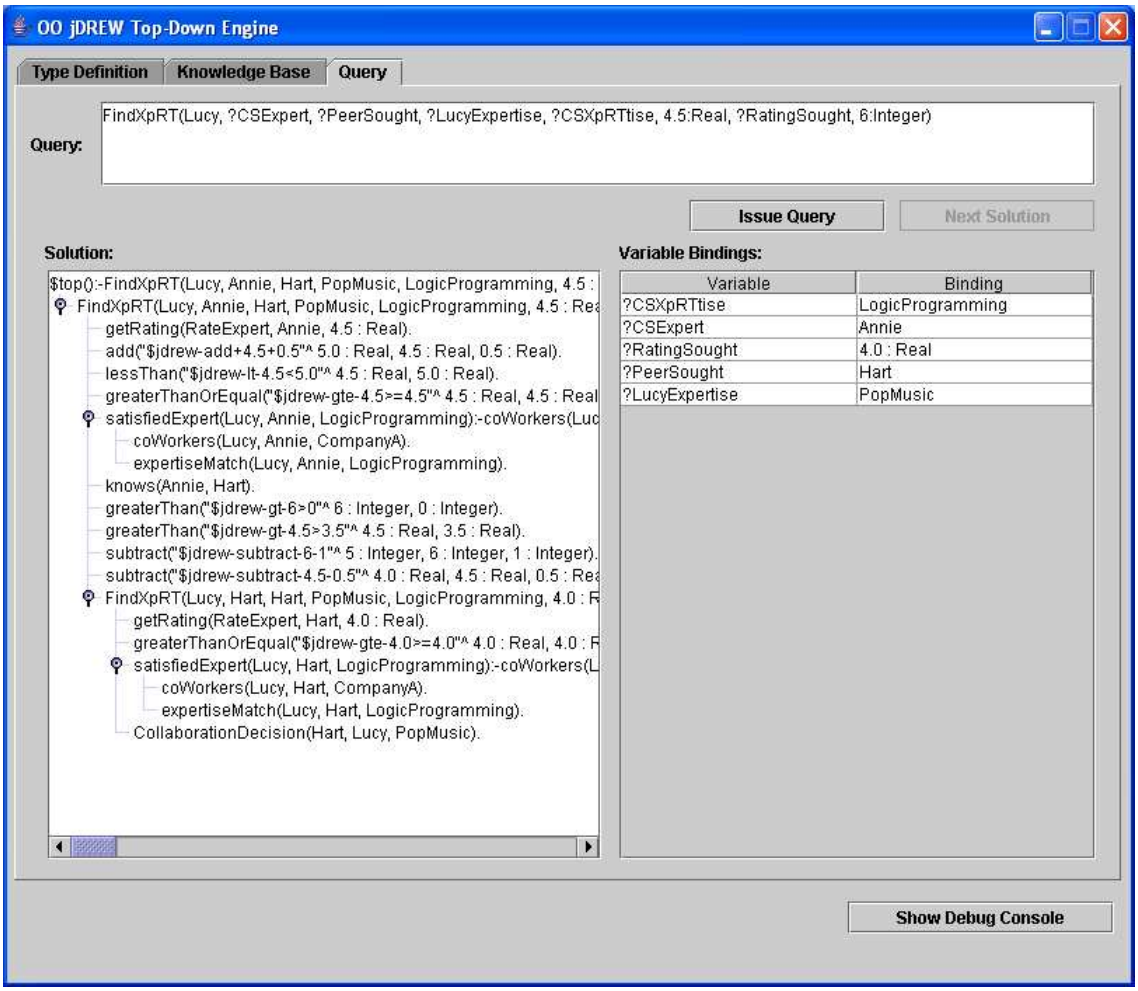

Figure 6.5: Screen Shot for Example 2.b

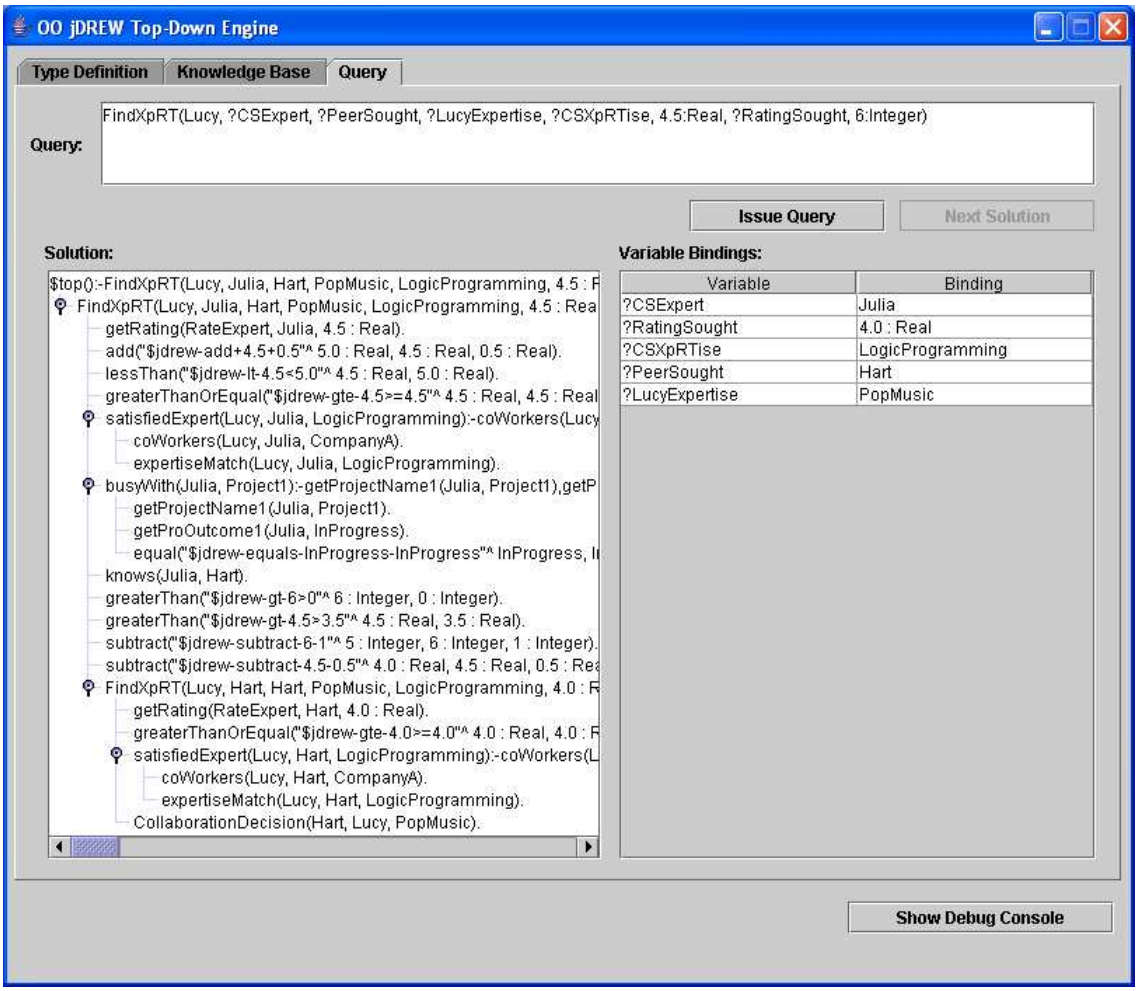

Figure 6.6: Screen Shot for Example 2.c

Result Bindings:

```
?CSXpRT = Julia
?RatingSought = 4.0:Real
?CSXpRTise = LogicProgramming
?ReferredXpRT = Hart
?LucyExpertise = PopMusic
```

6.1.3 Failure to Find an Expert

Example 3: *Sue discovers John. John is busy and he refers to Peter. Peter does not accept and he refers to Hart. Hart fails because the rating threshold fails (less than or equal to 3.0:Real). The result is shown in Figure 6.7.*

Query (against Sue's, John's, Peter's and Hart's profiles in Appendix A):
FindXpRT(Sue, ?CSXpRT, ?ReferredXpRT, ?SueExpertise, ?CSXpRTise, 5.0:Real, ?RatingSought, 6:Integer)

Result Bindings:

```
No solution
```

6.1.4 Find More Than One Expert

Example 4: *Mary discovers Amy and John. Amy accepts. Mary asks for another Computer Science expert. Since John is busy, he refers to Peter. Peter does not accept Mary's collaboration request and he refers to Hart. Hart accepts.*

Two screen shots are shown in the following part, with the first one showing the direct search and the second one showing the matched expert after two rounds of referrals.

Query (against Mary's, Amy's, John's, Peter's and Hart's profiles in Appendix A):
FindXpRT(Mary, ?CSXpRT, ?ReferredXpRT, ?MaryExpertise, ?CSXpRTise, 5.0:Real, ?RatingSought, 6:Integer)

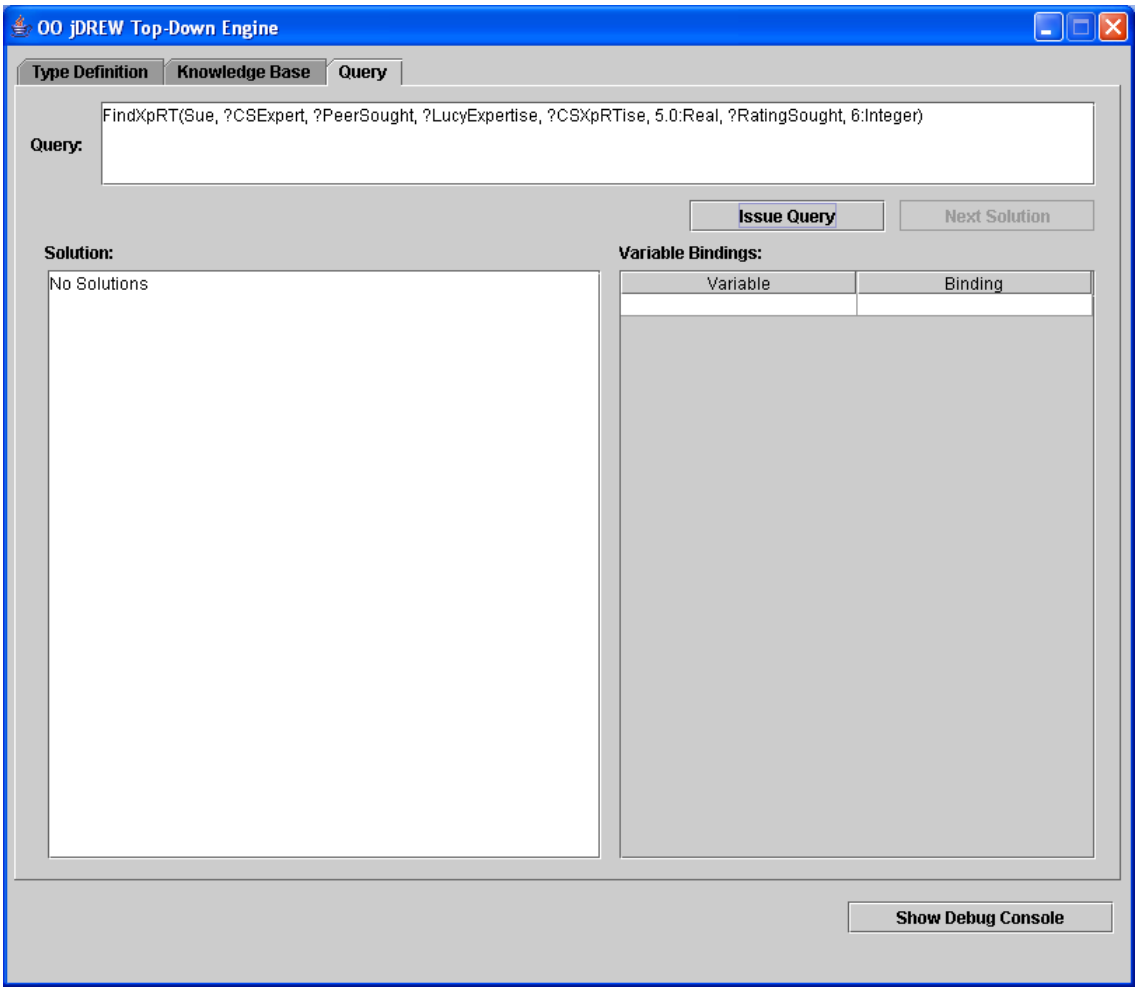

Figure 6.7: Screen Shot for Example 3

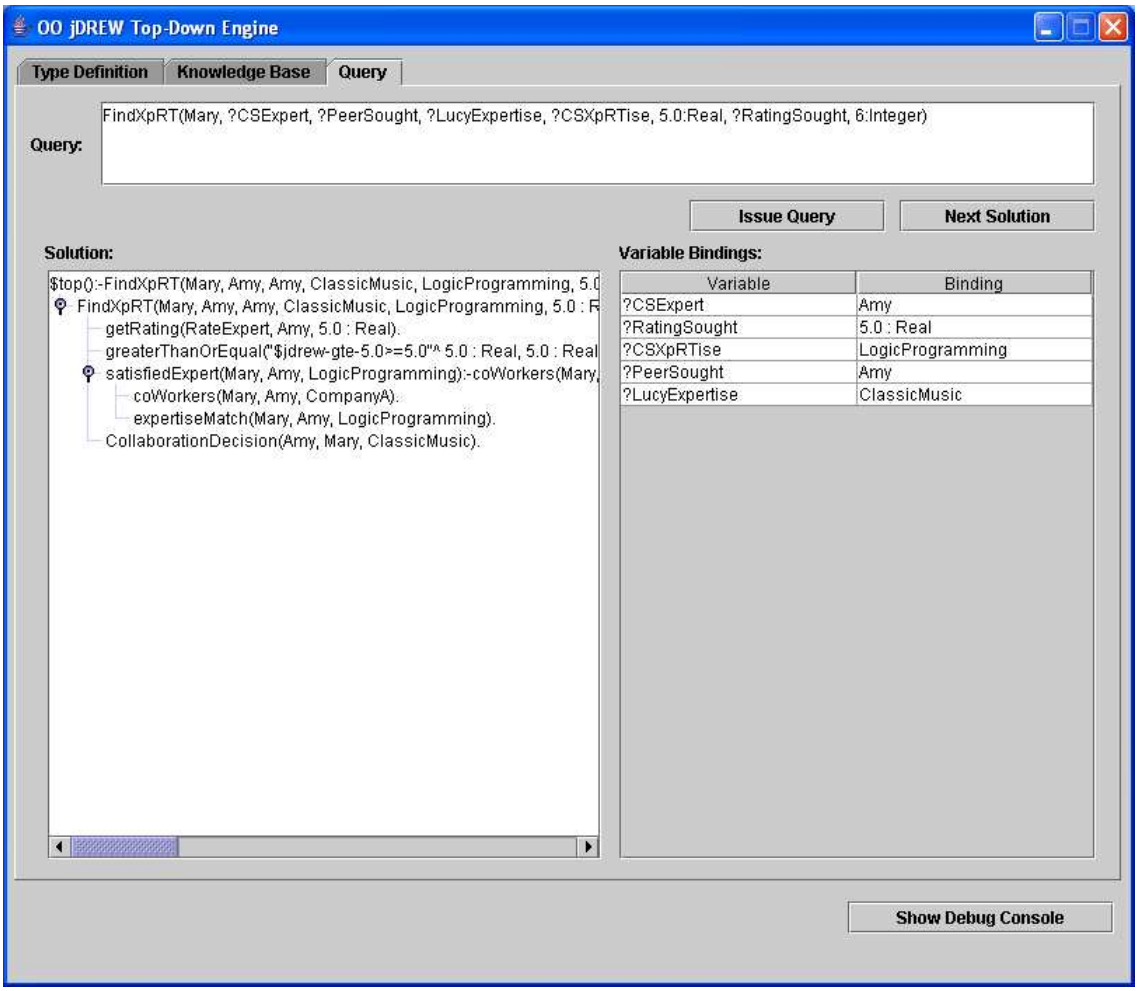

Figure 6.8: Screen Shot for Example 4.a

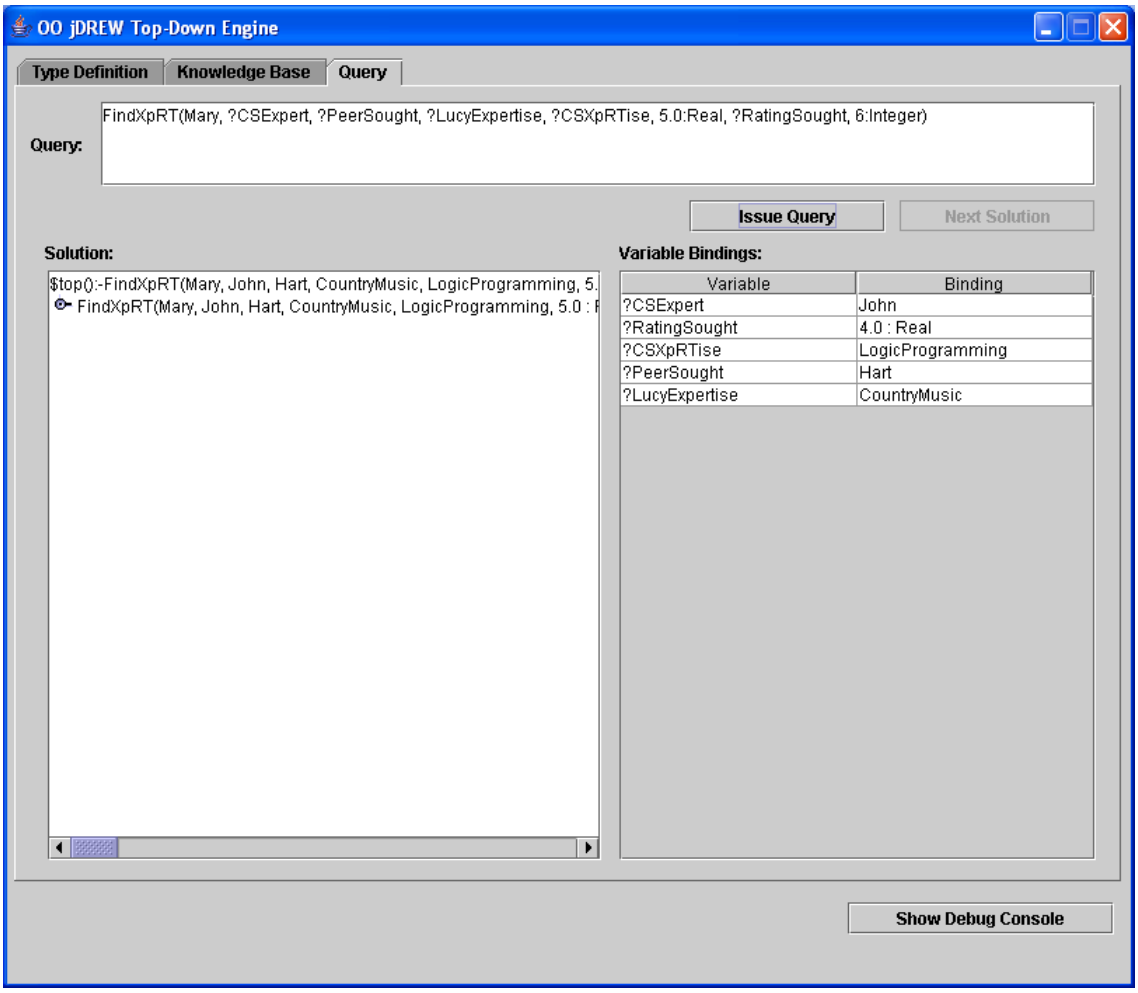

Figure 6.9: Screen Shot for Example 4.b

Result Bindings:

```
?CSXpRT = Amy
?RatingSought = 5.0:Real
?CSXpRTise = LogicProgramming
?ReferredXpRT = Amy
?MaryExpertise = ClassicMusic
```

Figure 6.8 shows the first expert that the system provides to Mary, via direct search. When Mary asks for another expert, as shown in Figure 6.9, the system goes through two rounds of referrals providing Hart to Lucy, referred to by Peter, who is again referred to by John.

Result Bindings:

```
?CSXpRT = John
?RatingSought = 4.0:Real
?CSXpRTise = LogicProgramming
?ReferredXpRT = Hart
?LucyExpertise = ClassicMusic
```

The end user of FindXpRT is considered as the co-expert in the collaboration, e.g., music expert Lucy in our scenario, while the expert the FindXpRT provides to the end user is regarded as the expert, e.g., the Computer Science expert Peter in our scenario.

In this section, we evaluate our FindXpRT system via three aspects: discussing fact and rule variation, query before or after FNF (RNF) generation and query entire profiles for expert referrals.

6.2 Experimental Variations

In this section, we test our FindXpRT system with experimental variations, including the variation of expert profiles and rules, as well as exchange of roles of experts and co-experts.

6.2.1 Experts' Profiles and Rule Variation

Recall Example 1: Lucy finds Peter and Peter accepts. The result would change if we change Lucy's or Peter's fact profile.

For example, changing the fact "foaf.membershipClass -> ComputMusic;" to "foaf.membershipClass -> CSConsult;" in either Lucy's or Peter's profile would cause FindXpRT fails to find any expert. This is because we have the following rule "coWorkers", which ensures that the two collaborators must be within in the same companies. Otherwise, it fails.

```
coWorkers(?Peer1, ?Peer2, ?Group) :-
   getCompany(?Peer1, ?Group),
   getCompany(?Peer2, ?Group),
   notEqual(?Peer1, ? Peer2).
```

Similarly, if we change "ex.offersExpertise -> PopMusic;" to "ex.offersExpertise -> ClassicMusic;", and/or change "ex.seeksExpertise-> LogicProgramming;" to "ex.seeksExpertise-> ArtifitialIntelligence;" in Lucy's fact profile, the rule "expertiseMatch" will fail and hence cause the FindXpRT to fail. The same would happen if we change Peter's fact profile.

```
expertiseMatch(?Peer1, ?Peer2, ?Expertise) :-
   offersExpertise(?Peer2, ?Expertise),
   seeksExpertise(?Peer1, ?Expertise),
   notEqual(?Peer1, ?Peer2).
```

FindXpRT would also fail if we change Peter's ratings in the rating agent profile so that his average rating is lower than the rating threshold issued by Lucy.

Recall Example 2: Lucy discovers Hart via one level of referral. Changing relevant persons' profile(s) can also cause the change of FindXpRT results.

In Example 2 b.1, Lucy discovers Annie, but Annie declines to collaborate with Lucy since Lucy's offered expertise does not match Annie's sought expertise. If we change "ex.offersExpertise -> PopMusic;" in Lucy's fact profile into "ex.offersExpertise -> ClassicMusic;", which matches "ex.seeksExpertise-> ClassicMusic;" in Annie's fact profile. In this example, Annie will be matched to Lucy directly, and Hart will not be discovered since Annie agrees to collaborate with Lucy and hence will not recommend other people any more.

The same will happen if we change "ex.seeksExpertise-> ClassicMusic;" into "ex.seeksExpertise-> PopMusic;" in Annie's fact profile.

In Example 2 b.2, Lucy first discovers Julia, but Julia is busy with other project(s). Therefore, we see that referral can not only be invoked by the declination of the expert, but also the unavailability of him/her. If we change "ex.outcome -> Proposed" to "ex.outcome -> Terminated" or "ex.outcome -> Accomplished", Julia will be found for Lucy for collaboration. As in Example 2 b.1, referral to Hart will not occur.

Also, if we change "ex.seeksExpertise -> PopMusic;" into "ex.seeksExpertise -> ClassicMusic;" in Hart's fact profile, Hart will also declines and the second rounds of referral will then occur. Likewise, change "ex.outcome -> Accomplished" to "ex.outcome -> Proposed" or "ex.outcome -> InProgress" will cause the same result.

```
busyWith(?Peer, ?Project) :-
   getProjectName1(?Peer, ?Project),
   getProOutcome1(?Peer, ?Outcome),
   equal(?Outcome, InProgress).
busyWith(?Peer, ?Project) :-
   getProjectName1(?Peer, ?Project),
   getProOutcome1(?Peer, ?Outcome),
   equal(?Outcome, Proposed).
```

Recall Example 3: Sue fails to find any expert to collaborate with. We analyze different probabilities here due to different changes to people's fact profile or even global and non-individualized rules.

In John's fact profile, if we change "ex.outcome -> InProgress" into "ex.outcome -> Terminated" or "ex.outcome -> Accomplished". Sue will find John to collaborate with and no referral is involved.

If we make changes in Peter's fact profile, replacing "ex.seeksExpertise -> PopMusic;" with "ex.seeksExpertise -> CountryMusic;", which matches "ex.offersExpertise -> CountryMusic;" in Sue's fact profile. Therefore, in this case, Sue finds Peter to collaborate with through one level of referral. Note that changing "ex.offersExpertise -> PopMusic;" instead will get the same result.

We can also keep all the persons' fact profiles unchanged and only changes the lower bound of rating threshold for terminating the FindXpRT recursion (details of FindXpRT rule set, see Appendix C). For example, if we change the premisses "greaterThan(?Rating, 3.5:Real)," to "greaterThan(?Rating, 3.0:Real)," in two of the FindXpRT recursive calls, Hart will probably be found ultimately through two levels of referrals.

Recall Example 4: Mary finds Amy via direct search and Hart via two levels of referrals. Similarly, we can change Amy's or Mary's fact profile because Amy's sought expertise will not match Mary's offered expertise. We can also change Amy's availability to "busy" in her fact profile. In both cases will we invoke Amy's FindXpRT referrals.

Likewise, properly changing John's and Peter's fact profile can also cause a match between Mary and John, or Mary and Peter, respectively.

6.2.2 Exchanging the Roles of Experts and Co-Experts

The FindXpRT system is also tested by exchanging the roles of experts and co-experts.

Recall **Example 1**: Music expert Lucy found Computer Science expert via direct search. We exchange the roles of Lucy and Peter. Peter, acting as the end user, can still be matched to Lucy.

Query: FindXpRT(Peter, ?MusicXpRT, ?ReferredXpRT, ?PeterExpertise, ?MusicXpRTise, 4.5:Real, ?RatingSought, 6:Integer)

Result Bindings:

```
?MusicXpRT = Lucy
?ReferredXpRT = Lucy
?PeterExpertise = LogicProgramming
?MusicXpRTise = PopMusic
?RatingSought = 4.5:Real
```

Recall **Example 2**: Hart, a Computer Science expert, is found by Music expert Lucy via one round of referral. We exchange the roles of Hart and Lucy, making Hart acting as the end user. Hart can find Lucy via direct search in this case.

Query: FindXpRT(Hart, ?MusicXpRT, ?ReferredXpRT, ?HartExpertise, ?MusicXpRTise, 4.5:Real, ?RatingSought, 6:Integer)

Result Bindings:

```
?MusicXpRT = Lucy
?ReferredXpRT = Lucy
?HartExpertise = LogicProgramming
?MusicXpRTise = PopMusic
?RatingSought = 4.5:Real
```

Recall **Example 4**: Mary found Amy and John. We exchange the roles of Mary and Amy, and Mary and John respectively. The result shows that neither Amy nor John can find Mary as her/his collaborator because Mary's rating is lower than the rating threshold and she is not recommended by other experts.

Query: FindXpRT(Amy, ?MusicXpRT, ?ReferredXpRT, ?AmyExpertise, ?MusicXpRTise, 5.0:Real, ?RatingSought, 6:Integer)

Result Bindings:

```
No solution
```

Query: FindXpRT(John, ?MusicXpRT, ?ReferredXpRT, ?JohnExpertise, ?MusicXpRTise, 5.0:Real, ?RatingSought, 6:Integer)

Result Bindings:

```
No solution
```

6.2.3 Taxonomic Expertise Matching

Matching between expertise need not be based on exact matches but can employ taxonomic matches, where a successful match of the expert's and the co-expert's expertise only need to find a common 'super-expertise'.

We have implemented a Computer Science expertise taxonomy in RDFS based on the ACM classification. Supported by the order-sorted types of our rule engine OO jDREW (with RDFS loaded in "Type Definition"), Computer Science expertise taxonomy can be used to help the matched between co-experts and experts if the expertise that the co-experts seeks is a subclass of the expertise the expert offers.

We give an example in Figure 6.10 to show the important special case of taxonomic matches, where the expert's expertise subsumes the co-expert's expertise. Recall Lucy's and Peter's profiles in Appendix A. In Lucy's profile, we change "ex.seeksExpertise-> LogicProgramming;" to "ex.seeksExpertise-> ?Expertise:Logic_Programming;" and in Peter's profile, we change "ex.offersExpertise -> ?Expertise:LogicProgramming;" to "ex.offersExpertise -> ?Expertise:Deduction_and_Theorem_Proving;", where *Deduction_and_Theorem_Proving* is a super class of *Logic_Programming*. Therefore, the co-expert Lucy's sought expertise is a subset of expert Peter's offered expertise and Figure 6.10 shows that our FindXpRT system finally provides Peter to Lucy.

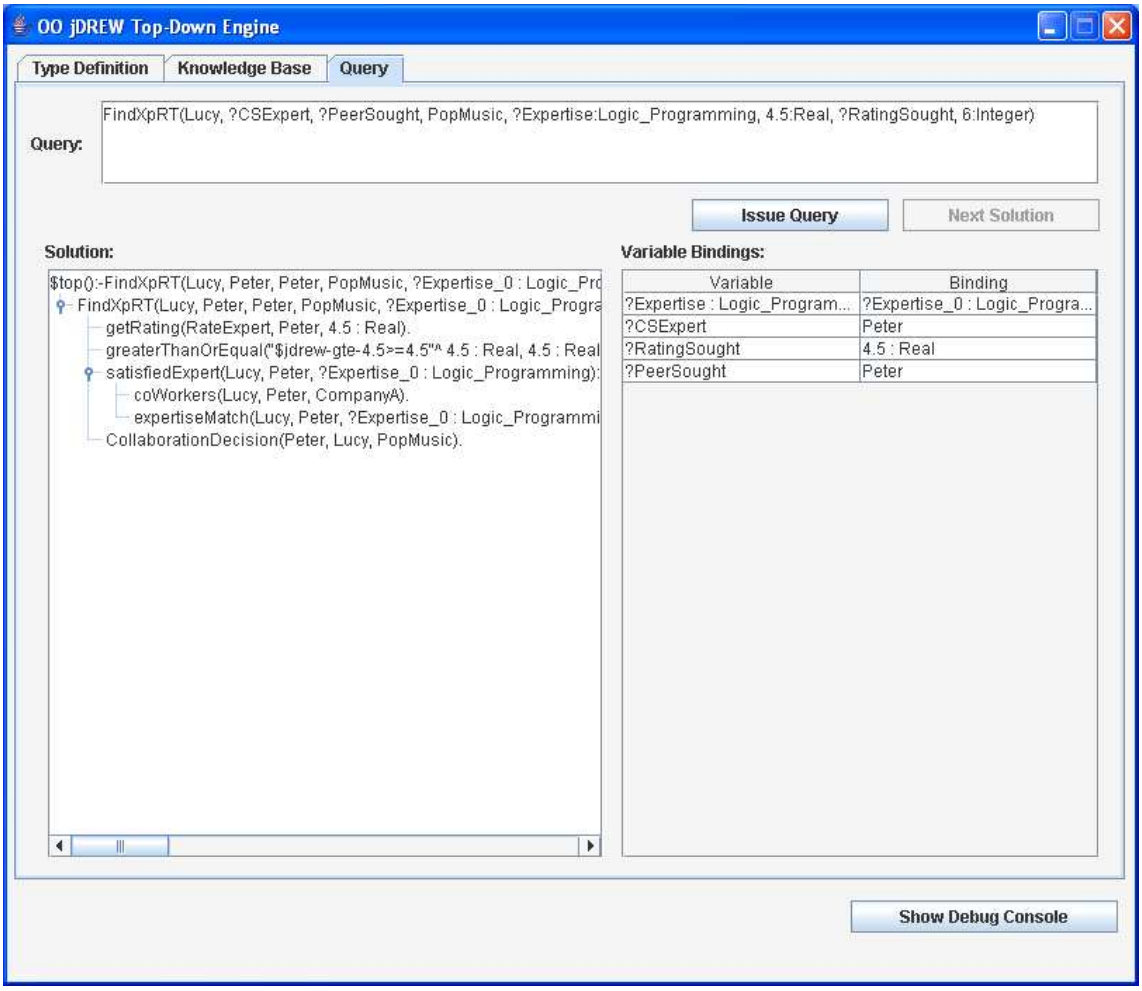

Figure 6.10: Expertise Matching Using RDFS

6.3 Expert Referrals

Expert referrals are featured as only experts recommending experts. An expert referral is asked only if the expert is a qualified expert to the user. Only then do we consider that he is qualified of giving referral. Since we assume the expert, who has the same expertise as the user queried and is qualified in this area against the criteria illustrated in Figure 5.3, may provide more valuable referrals.

Moreover, an expert referral is made also based on the assumption of "Six degree of separation" [39]. According to the "Six degree of separation" assumption that beyond six degree, two persons are unlikely to know each other, so in typical calls expert referrals are terminated after six rounds. This is because beyond six degree, referrals are viewed as invaluable as the user is unlikely to benefit from the recommended experts anyway.

Therefore, our expert referral is very helpful and valuable for the user to find an appropriate collaborator.

6.3.1 Query Entire Profiles for Expert Referrals

Note that expert referral is only requested when this current expert is not available (i.e., busy with other project(s)), or s/he does not accept the co-expert's collaboration request (according to the collaboration decision illustrated in Figure 5.4).

When the system cannot find a direct match towards a query, it asks for expert referrals. The system then tests the referral expert, and provides him/ her to the user if s/he matches with the user. Otherwise, we go to another round of expert referral. We consider that an expert's rating is virtually increased when being recommended by other experts. Therefore the virtual rating is considered to be increased every time in order to find a best match for the user and the expert among all those that previously failed. Every time when the system asks for expert referral, the rating threshold is decreased by the threshold decrement (0.5 in our FindXpRT), until it reaches 3.0 (or less), which is considered to be the boundary for rating of qualified expertise versus non-qualified expertise (recall our FindXpRT scale: 1-5).

The reason why we decrease the rating threshold every time by threshold decrement is that the FindXpRT system exhaustively finds all the experts that matches the user against those profiles that have been loaded in the rule engine. Only when none of the experts matches the user will we go to referral step.

6.3.2 Avoid Detours When Finding An Expert

The FindXpRT system always find a best expert for its end user. Even there is more than one solutions, FindXpRT first suggests its user the expert found by the least rounds of referrals. Only if the user asks for more options will the system gives the user the expert with the second least rounds of referrals. Each time the system provides one more expert to the user who is the second a best match when asking for "Next Solution" until no more experts can be found.

Take the last example shown in Experiment Results for example. FindXpRT first finds Amy for Mary via direct search, shown in Figure 6.8. Then when Mary queries for "Next Solution", the expert Hart is then found (see Figure 6.9) through two rounds od referrals.

Moreover, since "knows" relation is used in FindXpRT referral, cyclic issue need to be avoided. That is, it is likely that the previously failed expert will be recommended back by other experts again and thus causes cyclic problems. We avoid circularity issue by limiting the rating upper bound each time in the FindXpRT recursive call. Experts that are recommended "back" must have already been found earlier because of their higher ratings. Therefore, an expert with a fixed rating cannot be recalled in more than one round since his/her rating cannot satisfy different non-overlapping intervals in different rounds.

In other words, detours are avoided in providing a best match in FindXpRT.

6.4 Execution Times

In this section we measure and discuss the execution times of our FindXpRT system and then explain possible factors causing the execution speed variations.

Note that the FNF is generated before querying, via OO jDREW BU, while the RNF is generated after querying, interactively by OO jDREW TD.

The FNF, generated by OO jDREW BU, includes all the stored and derivable facts. The advantages of generating FNF before querying are listed as follows:

- Scalability of rule computations posted in OO jDREW TD improves considerably when OO jDREW TD offers the pre-computation and provides derivable facts.
- Reuse of derivable facts is enabled by OO jDREW BU pre-computation. Re-computation of derivable facts is thus avoided.

For the RNF, it is generated after querying, since the means to generate it is by interactively posting queries in OO jDREW TD.

Tables 6.11, 6.12, 6.13 and 6.14 together show the execution times of FindXpRT for different expert profiles loaded in the system as well as variations of queries. In the "Expert Profile" column, an *Italics* font indicates that the person is playing the role of a co-expert, while a **Bold** font indicates that the person is playing the role of an expert. The execution time is measured under Windows, on the platform Dell Latitude 610, with a 1.86 GHz processor, 2.00 GB (782 MHz) of RAM. The version of the SUN Java Virtual Machine used here is 1.5.0_05, Java 2 Platform. The rule engine, OO jDREW, used in these measurements is in version 0.92.

Table 6.11 shows the execution time variations when a FindXpRT user, Lucy, queries the system for finding an expert via direct matches, where the number of the matched experts to the user is fixed (1 in this case). Table 6.12 shows, for the FindXpRT user, Lucy, how the execution times vary between direct matches and one round of referrals, where the amount of matched experts is of a fixed number (1 in this case). Table 6.13 also shows, for the FindXpRT user, Lucy, how the execution time varies when the referral round is up to 1 and the number of matched experts to the user is unfixed (1 to 2 in this case). Table 6.14 shows, with Mary being a FindXpRT user, referral rounds from 0 to 2 has been tested, along with the number of matched profiles unfixed (1 to 2 in this case).

From Table 6.11 to Table 6.14, we see that both the variation of expert profiles and queries affect the execution time. As introduced in the beginning of this section, we first use OO jDREW BU to derive pre-processed facts. The increase of expert profiles in POSL causes the generated XML profiles to increase by a factor of about 3.5. It appears that the execution time increases linearly with the profile size. Querying differently against different expert profiles invokes different sets of rules and thus causes changes in the execution time as shown in Table 6.12. Some rules require much longer times to process, for example, the FindXpRT recursive referral rules (cf. Appendix C). We see from Table 6.13 as well as 6.14 that the execution has a superlinear growth with the number of referral rounds.

We summarize the factors that affect the execution time as follows:

- If the user queries a specific expert, instead of an open one, the execution time decreases considerably, approximately 6 times less execution time (e.g., first row vs. second row; third vs. fourth in Table 6.11).

- Exchanging the roles of the expert and the co-expert in the query would also affect the execution time, where two cases are involved:

  - The co-expert finds the expert through the same referral rounds (including 0) as the expert would find the co-expert. In this case, exchanging of the roles would only slightly affect the execution time (e.g., row 1 vs. row 3 and row 2 vs. row 4 in Table 6.11).

  - The referral rounds (including 0) that the co-expert takes to find the expert is different from that of the expert would take to find the co-expert. In this case, exchanging the roles would affect the execution time considerably (e.g. row 7 vs. row 10 and row 8 vs. row 13 in Table 6.12).

- Instantiating the expertise of both co-expert and expert will only slightly improve the execution speed (e.g., row 4 vs. row 5 and row 8 vs. row 15 in Table 6.12).

- Varying permitted referral rounds may also affect the execution time, if among the

expert profiles loaded, more than one experts can be found, via different referral rounds (e.g., row 3 vs. row 4 and row 7 vs. row 8 in Table 6.13, as well as row 2, row 3 and row 5 in Table 6.14).

- With the change of the rating threshold, execution time may also vary, if among the expert profiles loaded, lower the rating threshold would cause more expert(s) be found (e.g., row 1 vs. row 3 and row 2 vs. row 4 in Table 6.13).

- If more expert profiles are loaded in the system, more time will be taken in execution, where also two cases are involved.

  - In this case, more expert profiles are loaded which will not be matched to the co-expert. Here, the time is less affected by the increase of the loaded files (e.g., row 6 vs. row 7 in Table 6.11, where the profile set of Lucy, Peter and Karen is a superset of the profile set of Lucy and Peter).

  - In this case, more expert profiles are loaded which will be matched to the co-expert. Here, however, the increase of the loaded files greatly causes an increase of the execution time (e.g., row 6 in Table 6.11 vs. row 4 in Table 6.13, where the profile set of Lucy, Peter, Annie and Hart is a superset of the profile set of Lucy and Peter, and Hart will also be matched to Lucy via one round of referral).

  - Using Naf (Negation as failure) can also cause an increase of execution time (e.g., row 5 and row 15 in Table 6.12, where more Naf rules are involved in the query of row 5 than in row 15). The execution times can be decreased if Naf is replaced with negative relations such as `not-collaborates`.

  - Because many rules are packaged as person-centric modules, for these, search can be localised.

The longer runtimes of FindXpRT are mostly due to its current execution in the OO jDREW engine (which has been implemented initially to teach how to build (pure) Prolog interpreters in Java), and has been under development as a reference implementation of

Expert Profile	Query Posted	Execution Time (ms)
Lucy, **Peter**	FindXpRT(Lucy, Peter, ?ReferredXpRT, PopMusic, LogicProgramming, 4.5:Real, ?RatingSought, 0:Integer)	13766
Lucy, **Peter**	FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, PopMusic, LogicProgramming, 4.5:Real, ?RatingSought, 0:Integer)	83734
Lucy, **Peter**	FindXpRT(Peter, Lucy, ?ReferredXpRT, LogicProgramming, PopMusic, 4.5:Real, ?RatingSought, 0:Integer)	15786
Lucy, **Peter**	FindXpRT(Peter, ?MusicXpRT, ?ReferredXpRT, LogicProgramming, PopMusic, 4.5:Real, ?RatingSought, 0:Integer)	122317
Lucy, **Peter**	FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, ?MusicXpRTise, ?CSXpRTise, 4.5:Real, ?RatingSought, 0:Integer)	185015
Lucy, **Peter**	FindXpRT(Peter, ?MusicXpRT, ?ReferredXpRT, ?CSXpRTise, ?MusicXpRTise, 4.5:Real, ?RatingSought, 0:Integer)	182849
Lucy, **Peter, Karen**	FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, LogicProgramming, PopMusic, 4.5:Real, ?RatingSought, 0:Integer)	87328
Lucy, **Peter, Hart**	FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, LogicProgramming, PopMusic, 4.5:Real, ?RatingSought, 0:Integer)	87969

Figure 6.11: Experiments on Direct Search, with Fixed Number of Matched Profiles

RuleML since then [23]. Actually, FindXpRT has been the most challenging rule-intensive OO jDREW application so far (NBBizKB [21] was more fact-intensive). Moreover, the longer execution times of OO jDREW TD searches are caused by the semantically "parallel" firing of (textually unordered) rules simulated by the (round-robin-like) iterative-deepening procedure of OO jDREW. In spite of these efficiency issues, OO jDREW has been a success as a deductive prototype engine for the derivation rule executions needed in the FindXpRT experiments, as it helped to discover relevant efficiency variations.

The efficiency of the FindXpRT implementation through an OO jDREW ruleset could be improved by introducing mode declarations in POSL / OO RuleML and in the OO jDREW implementation, to reflect the intended FindXpRT dataflow (for this, POSL could borrow the '::'-syntax used by Mercury,[1] and OO jDREW could prune ill-moded calls).

Since the textual order of conjuncts (e.g., rule premises) is not exploited by our pure logic programs, in the current version the instantiation state of the argument *?ReferralPeer2* of the recursive FindXpRT call of

[1] http://www.cs.mu.oz.au/research/mercury/information/documentation.html

Expert Profile	Query Posted	Execution Time (ms)
Lucy, **Annie, Hart**	FindXpRT(Lucy, Annie, ?ReferredXpRT, PopMusic, LogicProgramming, 4.5:Real, ?RatingSought, 0:Integer)	400352
Lucy, **Annie, Hart**	FindXpRT(Hart, Lucy, ?ReferredXpRT, LogicProgramming, PopMusic, 4.5:Real, ?RatingSought, 0:Integer)	15139
Lucy, **Annie, Hart**	FindXpRT(Hart, ?MusicXpRT, ?ReferredXpRT, LogicProgramming, PopMusic, 4.5:Real, ?RatingSought, 0:Integer)	95436
Lucy, **Annie, Hart**	FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, PopMusic, LogicProgramming, 4.5:Real, ?RatingSought, 1:Integer)	2015041
Lucy, **Annie, Hart**	FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, ?MusicXpRTise, ?CSXpRTise, 4.5:Real, ?RatingSought, 1:Integer)	2229889
Lucy, **Julia, Hart**	FindXpRT(Hart, ?MusicXpRT, ?ReferredXpRT, ?CSXpRTise, ?MusicXpRTise, 4.5:Real, ?RatingSought, 0:Integer)	88505
Lucy, **Julia, Hart**	FindXpRT(Lucy, Julia, ?ReferredXpRT, PopMusic, LogicProgramming, 4.5:Real, ?RatingSought, 1:Integer)	359875
Lucy, **Julia, Hart**	FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, PopMusic, LogicProgramming, 4.5:Real, ?RatingSought, 1:Integer)	1641623
Lucy, **Julia, Hart**	FindXpRT(Hart, Lucy, ?ReferredXpRT, LogicProgramming, PopMusic, 4.5:Real, ?RatingSought, 0:Integer)	14911
Lucy, **Julia, Hart**	FindXpRT(Julia, Lucy, ?ReferredXpRT, LogicProgramming, PopMusic, 4.5:Real, ?RatingSought, 0:Integer)	14679
Lucy, **Julia, Hart**	FindXpRT(Hart, ?MusicXpRT, ?ReferredXpRT, LogicProgramming, PopMusic, 4.5:Real, ?RatingSought, 0:Integer)	68590
Lucy, **Julia, Hart**	FindXpRT(Julia, ?MusicXpRT, ?ReferredXpRT, LogicProgramming, PopMusic, 4.5:Real, ?RatingSought, 0:Integer)	56808
Lucy, **Julia, Hart**	FindXpRT(Hart, ?MusicXpRT, ?ReferredXpRT, ?CSXpRTise, ?MusicXpRTise, 4.5:Real, ?RatingSought, 0:Integer)	68590
Lucy, **Julia, Hart**	FindXpRT(Julia, ?MusicXpRT, ?ReferredXpRT, ?CSXpRTise, ?MusicXpRTise, 4.5:Real, ?RatingSought, 0:Integer)	56808
***Lucy,* Julia, Hart**	FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, ?MusicXpRTise, ?CSXpRTise, 4.5:Real, ?RatingSought, 1:Integer)	1641624
Lucy, Sue, **Julia, Hart**	FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, PopMusic, LogicProgramming, 4.5:Real, ?RatingSought, 1:Integer)	2016514
Lucy, Sue, **Peter, Karen**	FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, PopMusic, LogicProgramming, 4.5:Real, ?RatingSought, 0:Integer)	90438

Figure 6.12: Experiments on up to 1 Round of Referral, with Fixed Number of Matched Profiles

Expert Profile	Query Posted	Execution Time (ms)
Lucy, **Peter, Annie, Hart**	FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, PopMusic, LogicProgramming, 4.0:Real, ?RatingSought, 0:Integer)	35378
Lucy, **Peter, Annie, Hart**	FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, PopMusic, LogicProgramming, 4.0:Real, ?RatingSought, 1:Integer)	36691
Lucy, **Peter, Annie, Hart**	FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, PopMusic, LogicProgramming, 4.5:Real, ?RatingSought, 0:Integer)	97149
Lucy, **Peter, Annie, Hart**	FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, PopMusic, LogicProgramming, 4.5:Real, ?RatingSought, 1:Integer)	1783903
Lucy, **Peter, Julia, Hart**	FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, PopMusic, LogicProgramming, 4.0:Real, ?RatingSought, 0:Integer)	35409
Lucy, **Peter, Julia, Hart**	FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, PopMusic, LogicProgramming, 4.0:Real, ?RatingSought, 1:Integer)	35971
Lucy, **Peter, Julia, Hart**	FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, PopMusic, LogicProgramming, 4.5:Real, ?RatingSought, 0:Integer)	98148
Lucy, **Peter, Julia, Hart**	FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, PopMusic, LogicProgramming, 4.5:Real, ?RatingSought, 1:Integer)	1779498
Lucy, Mary, Sue, **Peter, Hart**	FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, PopMusic, LogicProgramming, 4.5:Real, ?RatingSought, 0:Integer)	129072
Lucy, Mary, Sue, **Julia, Hart**	FindXpRT(Lucy, ?CSXpRT, ?ReferredXpRT, PopMusic, LogicProgramming, 4.5:Real, ?RatingSought, 1:Integer)	2229889

Figure 6.13: Experiments on up to 1 Round of Referral, with Number of Matched Profile Variations

Expert Profile	Query Posted	Execution Time (ms)
Marry, **Amy**	FindXpRT(Mary, ?CSXpRT, ?ReferredXpRT, CountryMusic, LogicProgramming, 5.0:Real, ?RatingSought, 0:Integer)	16975
Marry, **Amy**	FindXpRT(Mary, ?CSXpRT, ?ReferredXpRT, ?MusicXpRTise, ?CSXpRTise, 5.0:Real, ?RatingSought, 0:Integer)	87048
Marry, **Peter, Hart**	FindXpRT(Mary, ?CSXpRT, ?ReferredXpRT, ?MusicXpRTise, ?CSXpRTise, 4.5:Real, ?RatingSought, 1:Integer)	219072
Marry, **John, Peter, Hart**	FindXpRT(Mary, ?CSXpRT, ?ReferredXpRT, ?MusicXpRTise, ?CSXpRTise, 5.0:Real, ?RatingSought, 1:Integer)	228883
Marry, **John, Peter, Hart**	FindXpRT(Mary, ?CSXpRT, ?ReferredXpRT, ?MusicXpRTise, ?CSXpRTise, 5.0:Real, ?RatingSought, 2:Integer)	7363799

Figure 6.14: Experiments on up to 2 Rounds of Referral, with Number of Matched Profile Variations

```
FindXpRT(?Peer1,?Peer2,...) :-
        . . .
        knows(?Peer2, ?ReferralPeer2),
        . . .
        FindXpRT(?Peer1, ?ReferralPeer2, ...).
```

is not forced to be of mode 'in' (ground), although *?ReferralPeer2* is supposed to have been bound through the knows call, with mode 'out' (free), by the time this referring FindXpRT call is executed. This means that FindXpRT unnecessarily searches for candidate bindings of *?ReferralPeer2*, although only the one referred to by *?Peer2* will ultimately be eligible.

The following optimised moded version would achieve the intended flow of the *?ReferralPeer2* data from knows to FindXpRT:

```
FindXpRT(?Peer1,?Peer2,...) :-
        . . .
        knows(?Peer2, ?ReferralPeer2::out),
        . . .
        FindXpRT(?Peer1, ?ReferralPeer2::in, ...).
```

Chapter 7

Conclusions

In this thesis, we first put forward an approach of combining RuleML and FOAF, namely RuleML FOAF, and showed how it can enhance the current RDF FOAF. We have implemented and introduced our FindXpRT system using RuleML FOAF, followed by a benchmark for FindXpRT for Computer Science and music profiles.

In the following sections, we first discuss the contributions of this thesis, followed by future work.

7.1 Contributions

We have proposed a combination of RDF FOAF facts and RuleML FOAF rules. We have extended the current FOAF vocabulary to RuleML FOAF via the human-oriented syntax for facts and rules, POSL as an intermediary language. We have then designed the rule vocabulary specification for RuleML FOAF, as an extension to the current FOAF (fact-only) vocabulary. With the help of RuleML FOAF, users can derive FOAF data by employing person-centric rules, either before (RDF) FOAF publication or, on demand, from published (RuleML FOAF) pages.

Next, we have implemented our FindXpRT, a prototypical system expert finding using RuleML FOAF, for semantically finding an expert via rules and taxonomies. The FindXpRT

system is featured as a intermediary in matching the co-experts with experts for future collaboration. Our FindXpRT system can not only offer its users the expert that exactly meets his/her every criteria, but also can provide expert referrals, which can help the users to find other possible and appropriate experts. Our FindXpRT system also assist its users to find the best mode to collaborate according to the date, the day as well as the distance between collaborators.

We have tested our FindXpRT system in the rule engine OO jDREW to compute the result for expert finding, where both bottom-up and top-down execution are needed. Bottom-up execution provides us with the all the newly derived facts as required for the Fact-oriented Normal Form (FNF). Top-down execution enables users intending to find a collaboration expert to query specific information on demand, as requested by the Rule-oriented Normal Form (RNF). All the facts provided by FNF, given and derivable, can be mapped back to RDF syntax on demand via our existing XSLT translator. This translation can benefit the RDF FOAF community in developing enriched FOAF vocabulary specifications as called for by concrete use cases. Moreover, our FindXpRT using RuleML FOAF has been the most challenging rule-intensive OO jDREW application so far.

Finally, we have proposed a benchmark for FindXpRT in the domains of Computer Science and music.Variations of experiment has been carried on, followed by a discussion of the factors that affect the execution time.

7.2 Future Work

Expert finding has been a really popular area for both researchers and practitioners over the last few years. In particular, expert finding using FOAF has also become an attractive and promising area. A joint initiative, ExpertFinder, on extending the FOAF vocabulary for expert finding, has just been founded in July, 2006 (http://rdfweb.org/topic/ExpertFinder).

Currently, our FindXpRT system supports mainly exact expertise match, although Computer Science expertise based on the ACM classification has already been implemented. As

the taxonomic expertise match alone can hardly satisfy users by simply find a common 'super-expertise' between the users' sought expertise and the experts' offered expertise. Preference of the users on how similar the experts expertise would be with their sought expertise should be supported. Therefore, coupling taxonomic expertise match with similarity matching [68], i.e., Teclantic.ca, which offers similarity matching would be our next research step.

"Stress testing" of the FindXpRT on a large number of artificial experts profiles is also considered as our future work to evaluate our FindXpRT system, where the artificial expert profiles will also be generated automatically.

The longer runtimes of FindXpRT are mostly due to its current execution in the OO jDREW engine, which has been under development as a reference implementation of RuleML [23]. One major factor that causes the longer execution times of OO jDREW TD searches is the semantically "parallel" firing of (textually unordered) rules simulated by the (round-robin-like) iterative-deepening procedure of OO jDREW. Parallel processing of the profiles of FindXpRT users against corresponding rules in a distributed system will be needed, which is regarded as one of the future work items extending this thesis to improve the efficiency of FindXpRT.

Moreover, the FNF and RNF are derived interactively, essentially by running OO jDREW BU for the FNF, and checking fact drivability by running OO jDREW TD for the RNF. A control loop for automatically generating the FNF and RNF is planned for the future. More use cases drawn from Teclantic constitute another area of future work.

In this thesis, we used artificial expert profiles when introducing our FindXpRT system. Applying FindXpRT to real-world experts should be also considered as future work.

Similar to FOAF, issues are such as trust and data protection. As RuleML FOAF page is maintained by each individual, how can we trust this information? An earlier step we made, in FindXpRT, is that we store some sensitive data, e.g., expert rating, in a centralized database instead of in self-maintained profiles hold by each expert. Systematic evaluation of the sensitivity of data is needed. For data protection issue, implications of data protection legislation should be carried out as the next research step.

References

[1] Technology Reports: Rule Markup Language (RuleML). http://xml.coverpages.org/ruleML.html, 2002.

[2] FAQ - FOAF Wiki. http://www.nichepal.com/cgiproxy/nph-proxy.pl/010110A/http/rdfweb.org/topic/FAQ, 2005.

[3] Friendster. http://www.friendster.com/, October 2005.

[4] Mandarax. http://java-source.net/open-source/rule-engines/mandarax, September 2005.

[5] StumbleUpon. http://www.stumbleupon.com/, September 2005.

[6] SweetRules. http://sweetrules.projects.semwebcentral.org/, September 2005.

[7] All Experts. http://www.allexperts.com/, January 2006.

[8] AskanExpert. http://www.askanexpert.com/, January 2006.

[9] Expert Finder Home Page. http://web.media.mit.edu/ lieber/Lieberary/Expert-Finder/ Expert-Finder-Intro.html, January 2006.

[10] Expertise Search. http://www.expertisesearch.com/, January 2006.

[11] ExpertWitness. http://www.expertwitness.com/, January 2006.

[12] Look-up Latitude and Longitude - Canada. http://www.bcca.org/misc/qiblih/latlong_ca.html, July 2006.

[13] Music. http://www.aaai.org/AITopics/html/music.html, March 2006.

[14] Purdue Experts. http://purduenews.uns.purdue.edu/UNS/expertlists/ expert.current.html, January 2006.

[15] ReferralWeb. http://www.cs.washington.edu/homes/kautz/referralweb/, March 2006.

[16] RIF RuleML FOAF. W3C RIF-WG Wiki, 2006.

[17] Teclantic. http://teclantic.cs.unb.ca/index.jsp, January 2006.

[18] The ACM Computing Classification System [1998 Version]. http://www.acm.org/class/1998, 2006.

[19] The story of how foaf came to be . http://beta.foaf-project.org/2004/us/history.html, May 2006.

[20] Michelle Anderson, Marcel Ball, Harold Boley, Stephen Greene, Nancy Howse, Daniel Lemire, and Sean McGrath. RACOFI: A Rule-Applying Collaborative Filtering System. In *Proceedings of COLA'03*, pages 13–23. IEEE/WIC, October 2003.

[21] Anna Maclachlan and Harold Boley. Semantic Web Rules for Business Information. In *Proc. International Conference on Web Technologies, Applications, and Services (WTAS 2005), Calgary, Canada*. IASTED, 2005.

[22] Marcel Ball. OO jDREW. http://www.jdrew.org/oojdrew/, September 2005.

[23] Marcel Ball, Harold Boley, David Hirtle, Jingf Mei, and Bruce Spencer. The OO jDREW Reference Implementation of RuleML. In *Proc. Rules and Rule Markup Languages for the Semantic Web (RuleML-2005)*, pages 218–223. LNCS 3791, Springer-Verlag, November 2005.

[24] Nick Bassiliades. DR-DEVICE. http://iskp.csd.auth.gr/systems/dr-device.html, January 2006.

[25] Tim Berners-Lee. Semantic Web - XML2000. http://www.w3.org/2000/Talks/1206-xml2k-tbl, 2000.

[26] Harold Boley. RuleML for the Semantic Web. http://www.dfki.uni-kl.de/ boley/RuleML-RDF-OntoWeb/sld001.htm, June 2001.

[27] Harold Boley. The Rule Markup Language: RDF-XML Data Model, XML Schema Hierarchy, and XSL Transformations. In *Proc. 14th International Conference on Applications of Prolog (INAP2001)*. The University of Tokyo, LNAI 2543, Springer-Verlag, October 2002.

[28] Harold Boley. Integrating Positional and Slotted Knowledge on the Semantic Web. http://www.ruleml.org/posl/poslintweb-talk.pdf, March 2005.

[29] Lorne H. Bouchard. MCETECH 2006 Tutorial: The Semantic Web. http://tesniere.info.uqam.ca/MCETECH/The18th 2006.

[30] Dan Brickley. RDFWeb and Friend of a Friend (FOAF)– Getting started with FOAF. http://rdfweb.org/mt/foaflog/archives/000051.html, 2003.

[31] Dan Brickley. The Friend of a Friend (FOAF) project. http://www.foaf-project.org/, November 2005.

[32] Dan Brickley and Libby Miller. An Introduction to RDFWeb and FOAF . http://rdfweb.org/2000/08/why, 2002.

[33] Dan Brickley and Libby Miller. FOAF Vocabulary Specification. http://xmlns.com/foaf/0.1/, October 2005.

[34] Elizabeth F. Churchill and Christine A. Halverson. *IEEE Internet Computing*, volume 9, chapter Social Networks and Social Networking, pages 14–19. IEEE Computer Society, 2005.

[35] Sarah Currier and Lorna M. Campbell. The SeSDL Taxonomy. http://www.sesdl.scotcit.ac.uk, 2006.

[36] Leigh Dodds. An Introduction to FOAF. http://www.xml.com/pub/a/2004/02/04/foaf.html, 2004.

[37] Edd Dumbill. XML Watch: Finding friends with XML and RDF. http://www-128.ibm.com/developerworks/xml/library/x-foaf.html, June 2002.

[38] Tim Finin, James Mayfield, Anupam Joshi, R. Scott Cost, and Clay Fink. Information Retrieval and the Semantic Web. In *Proceedings of the 38th International Conference on System Sciences*, 2005.

[39] Stefania Ghita, Wolfgang Nejdl, and Raluca Paiu. Semantically Rich Recommendations in Social Networks for Sharing and Exchanging Semantic Context. In *Proc.4th International Semantic Web Conference (ISWC2005)*, pages 293–307, Galway, Ireland, November 2005.

[40] Gunnar AAstrand Grimnes, Pete Edwards, and Alun Preece. Learning Meta-Descriptions of the FOAF Network. In *ISWC 2004, LNCS 3298*, pages 152–165. Springer-Verlag, Berlin, 2004.

[41] RSS-DEV Working Group. RDF Site Summary 1.0. http://web.resource.org/rss/1.0/spec, October 2005.

[42] T. R. Gruber. Towards Principles for the Design of Ontologies Used for Knowledge Sharing. In N. Guarino and R. Poli, editors, *Formal Ontology in Conceptual Analysis and Knowledge Representation*, Deventer, The Netherlands, 1993. Kluwer Academic Publishers.

[43] Masahiro Hamasaki, Junichiro Mori, Hideaki Takeda, and Koiti Hasida. Ontological Consideration on Human Relationship Vocabulary for FOAF. 1st Workshop on Friend of a Friend, Social Networking and the Semantic Web, 2004.

[44] Elliotte Rusty Harold. *The XML Bible, 2nd Edition*, chapter 17: XSL Transformations. John Wiley & Sons, 2001.

[45] Jie. Li, Harold. Boley, Virendrakumar C. Bhavsar . Ruleml foaf: Web rules for social networking. In *RuleML 2006*, November 2006. To Appear.

[46] Jie Li, Harold Boley, Virendrakumma C. Bhavsar, Jing Mei. Expert Finding in eCollaboration Using FOAF with RuleML Rules. In *Montreal Conference of eTechnologies 2006*, pages 53–65, 2006.

[47] Jie Li, Virendrakumma C. Bhavsar, Harold Boley. RuleML FOAF: A Web Rule Language for Social Networking. MITACS Internship - Case Study, 2005.

[48] Jing Mei, Harold Boley, Jie Li, Virendrakumar C. Bhavsar, and Zuoquan Lin. DatalogDL: Datalog Rules Parameterized by Description Logics. In *Canadian Semantic Web*, volume 2 of *Semantic Web and Beyond*, pages 171–187, 2006.

[49] Jinghai Rao and Norman Sadeh. Interleaving Semantic Web Reasoning and Service Discovery to Enforce Context-Sensitive Security and Privacy Policies. In *The International Conference on Rules and Rule Markup Languages for the Semantic Web, Nov. 2005 (RuleML-2005)* , 2005.

[50] Michael Kifer, George Lausen, and James Wu. Logic Foundations of Object Oriented and Frame Based Languages. Journal of the Association for Computing Machinary, 1995.

[51] Ruben Laura. Semantic Web School. http://www.semantic-web.at/10.36.46.article.ruben-lara-semantic-web-technologien-helfen-das-web-zu-ordnen.htm, 2006.

[52] Shiyong Lu, Ming Dong, and Farshad Fotouhi. The Semantic Web: opportunities and challenges for next-generation Web applications. Information Research 7(4), 2002.

[53] Simone Ludwig. Introduction to Ontology Development. MCETECH 2006, 2006.

[54] Sebastien Mathieu. Match-making in bartering scenarios. Master thesis, University of New Brunswick, December 2005.

[55] Luke McDowell, Oren Etzioni, Steven D. Gribble, Alon Halevy, Hank Levy, William Pentney, Deepak Verma, and Stani Vlasseva. Mangrove: Enticing Ordinary People onto the Semantic Web via Instant Gratification. In *Proc. 2nd International Semantic Web Conference (ISWC2003)*, pages 754–770, Sanibel Island, Florida, USA, October 2003. Springer.

[56] Luke McDowell, Oren Etzioni, Steven D. Gribble, Alon Halevy, Henry Levy, William Pentney, Deepak Verma, and Stani Vlasseva. Evolving the Semantic Web with Mangrove. Technical Report UWCSE030201, Department of Computer Science and Engineering, University of Washington, Seattle, WA, February 2003.

[57] Ikki Ohmukai1, Hideaki Takeda, Masahiro Hamasaki, Kosuke Numa, and Shin Adachi. Metadata-Driven Personal Knowledge Publishing. In *ISWC 2004, LNCS 3298*, pages 591–604. Springer-Verlag, Berlin, 2004.

[58] National Information Standards Organization. Understanding Metadata. NISO Press, 2004.

[59] Sean B. Palmer. The Semantic Web: An Introduction. http://infomesh.net/2001/swintro/, 2001.

[60] Paul Niquette. 101 Words I Don't Use. Sophisticated:The Magazine, 1997.

[61] W3C Candidate Recommandation. XSL Transformations (XSLT) Version 2.0. http://www.w3.org/TR/xslt, November 2005.

[62] W3C Recommendation. RDF Vocabulary Description Language 1.0: RDF Schema. http://www.w3.org/TR/rdf-schema/, February 2004.

[63] Urvi Shah, Tim Finin, Anupam Joshi, R. Scott Cost, and James Mayfield. Information Retrieval On The Semantic Web. In *Proceedings of the eleventh international conference on Information and knowledge management*, pages 461–468. ACM Press, 2002.

[64] Bruce Spencer. jDREW. http://www.jdrew.org/jDREWebsite/jDREW.html, October 2005.

[65] Michael Stollberg, Uwe Keller, and Dieter Fensel. Partner and service discovery for collaboration establishment with semantic web services. In *ICWS*, pages 473–480, 2005.

[66] M. Uschold and M Gruninger. ONTOLOGIES:. Principles, Methods and Applications. In *Knowledge Engineering Review*, volume 11, pages 93–155, 1996.

[67] Stuart Weibel and Eric Miller. An Introduction to Dublin Core. http://www.xml.com/pub/a/2000/10/25/dublincore/index.html, October 2000.

[68] Lu Yang, Marcel Ball, Virendrakumar C. Bhavsar, and Harold Boley. Weighted Partonomy-Taxonomy Trees with Local Similarity Measures for Semantic Buyer-Seller Match-Making. Business Agents and the Semantic Web (BASeWEB) Workshop, 2005.

Appendix A: Expert Profiles

This appendix gives the extended FOAF profiles of ten typical (fictitious) persons who are experts in either music or computer science.

```
% Namespace URIs:
% foaf: http://www.foaf-project.org/
% SESDL: http://www.sesdl.scotcit.ac.uk/
% ex: www.ruleml.org/usecases/foaf/ex

%% Acting as the music experts

% URI of the POSL FOAF page: www.ruleml.org/usecases/foaf/Lucy.posl
% URI of the RuleML FOAF page: www.ruleml.org/usecases/foaf/Lucy.ruleml

%% Fact profile of Lucy

foaf.person(Lucy^
% Should have page URI: .../Lucy/index.html
% Basic information
        foaf.name->Lucy;
        foaf.title->Dr;
        foaf.membershipClass->ComputMusic;
        foaf.mbox->Lucy_AT_ComputMusic.com;
        foaf.knows->Ann;
        ex.languages->English;
        ex.workPlace->Canada;
        ex.address->Fredericton;
        ex.occupation->professor;
        ex.atWork->Details[
            ex.from->09.00:Real;
            ex.to->17.00: Real];
        ex.phones->Tel[
           ex.office->0235;
           ex.cell->0357;
           ex.home->7654;
           ex.voicemail->6665];
        ex.onBusiness->Details[
```

```
            ex.from->07.30:Real;
            ex.to->08.12:Real];
% Information about Person's expertise, including
% personal development, professional development and
% project development, where this taxonomy is from
% SeSDL (Scottish electronic Staff Development Library)
        ex.expertise->Taxonomy[
            ex.offersExpertise -> PopMusic;
            ex.seeksExpertise-> LogicProgramming;
            sesdl.personalDevelopment->Taxonomy[
              sesdl.communication->Taxonomy[
                foaf.name->synchronousCommunication[
                  foaf.name->mailingLists]];
              sesdl.training->Taxonomy[
                ex.learningMethod->deepLearning;
                ex.workDuration->4.0:Real;
                ex.expertiseType->teachingSkills;
                ex.student->Taxonomy[
                  ex.PHD->1:Integer;
                  ex.Master->3:Integer;
                  ex.Undergrad->4:Integer];
                  ex.area->LogicProgramming]];
            sesdl.profesionalDevelopment->Taxonomy[
              sesdl.mentors ->Taxonomy[
                sesdl.experiment->Details[
                  ex.item1->Details[
                    foaf.name->Exp1;
                    ex.teachingMethod->peerTeaching;
                    ex.level->Graduate;
                    ex.area->Deduction;
                    ex.studentNum->20:Integer]];
                  ex.totalTeachingHours->30:Integer];
              sesdl.lectures-> Details[
                sesdl.webcasting->Details[
                  ex.amount->30:Integer;
                  ex.item1->Details[
                    foaf.name->Lecture1;
                    ex.area->Deduction;
                    ex.range->International;
                    ex.location->Paris]];
                sesdl.radioBroadcasting ->Details[
                  ex.amount->100:Integer;
                  ex.item2->Details[
                    foaf.name->Lecture2;
                    ex.area->Deduction;
                    ex.range->International;
                    ex.participants->100:Integer]]];
              sesdl.courses->Details[
                ex.item1->Details[
```

```
    foaf.name->Course1;
    ex.mode->online;
    ex.level->Undergraduate;
    ex.area->LogicProgramming;
    ex.studentNum->30:Integer]];
sesdl.qualifications->Taxonomy[
  sesdl.degrees->PHD];
sesdl.publication->Details[
  sesdl.peerReviewed ->Details[
    ex.item1->Details[
        bibtex.title->Article1;
        ex.area->LogicProgramming;
        ex.type->JournalPaper;
        bibtex.bookTitle->Journal1;
        bibtex.year->1991:Integer;
        bibtex.pages->"12-16";
        bibtex.month->Sep;
        sesdl.intellectualProperty->Details[
          sesdl.copyright->Publisher1]];
    ex.item2->Details[
        bibtex.title->Article2;
        ex.area->MachineLearning;
        ex.type->ConferencePaper;
        bibtex.bookTitle->Journal3;
        bibtex.year->2000:Integer;
        bibtex.pages->"32-43";
        bibtex.month->Oct;
        sesdl.intellectualProperty->Details[
          sesdl.copyright ->Publisher2]];
    ex.item3->Details[
        bibtex.title->Article3;
        ex.area->LogicProgramming;
        ex.type->JournalPaper;
        bibex.bookTitle->Journal1;
        bibtex.year->2005:Integer;
        bibtex.pages->"21-29";
        bibtex.month->Feb;
        sesdl.intellectualProperty->Details[
          sesdl.copyright ->Publisher1]];
    ex.item4->Details[
        bibtex.title->Article4;
        ex.area->Deduction;
        ex.type->ConferencePaper;
        bibex.bookTitle->Journal8;
        bibtex.year->2005:Integer;
        bibtex.pages->"25-33";
        bibtex.month->May;
        sesdl.intellectualProperty->Details[
          sesdl.copyright ->Publisher2]]];
```

```
                ex.amount->4:Integer];
            sesdl.sabbaticals ->Details[
              ex.from->5.01:Real;
              ex.to->7.31:Real]];
          sesdl.projectDevelopment ->Taxonomy[
            sesdl.evaluation->Taxonomy[
              ex.type->summativeAssessment];
            ex.desiredSalary->Taxonomy[
              ex.price->15000:Real;
              ex.range->0.15:Real];
            sesdl.projectDevelopments ->Details[
              ex.collaboratesIn->Details[
                  ex.item1->Details[
                    foaf.name->Project1;
                    ex.mode->mixedMode;
                    ex.collaborate->Details[
                      ex.type->groupWork;
                      ex.amount->8:Integer;
                      ex.funds->15000:Integer];
                    ex.area->MachineLearning;
                    ex.range->Local;
                    ex.position->TeamLeader;
                    ex.outcome->Accomplished]]]]]).
% Rule profile of Lucy

% Phone preference of Lucy during different time of the day, attached
% to Lucy's profile (24 hour system represented with Real types, since
% TimeOfDay type not implemented)
phonePreference(Lucy, ?Time, office) :-
    greaterThanOrEqual(?Time, 9.00:Real),
    lessThan(?Time, 12.00:Real).
phonePreference(Lucy, ?Time, office) :-
    greaterThanOrEqual(?Time, 13.00:Real),
    lessThan(?Time, 17.00:Real).
phonePreference(Lucy, ?Time, cell) :-
    greaterThanOrEqual(?Time, 12.00:Real),
    lessThan(?Time, 13.00:Real).
phonePreference(Lucy, ?Time, cell) :-
    greaterThanOrEqual(?Time, 17.00:Real),
    lessThan(?Time, 18.00:Real).
phonePreference(Lucy, ?Time, home) :-
    greaterThanOrEqual(?Time, 18.00:Real),
    lessThan(?Time, 21.00:Real).
phonePreference(Lucy, ?Time, voicemail) :-
    greaterThanOrEqual(?Time, 21.00:Real),
    lessThan(?Time, 24.00:Real).
phonePreference(Lucy, ?Time, voicemail) :-
    greaterThanOrEqual(?Time, 0.00:Real),
    lessThan(?Time, 9.00:Real).
```

```
% URI of the POSL FOAF page: www.ruleml.org/usecases/foaf/Mary.posl
% URI of the RuleML FOAF page: www.ruleml.org/usecases/foaf/Mary.ruleml

%% Fact profile of Mary

foaf.person(Maryˆ
% Should have page URI: .../Mary/index.html
        foaf.name->Mary;
        foaf.title->Dr;
        foaf.membershipClass->ComputMusic;
        foaf.mbox->Mary_AT_ComputMusic.com;
        foaf.knows->Hart;
        ex.languages->English;
        ex.workPlace->Canada;
        ex.address->Toronto;
        ex.occupation->professor;
        ex.atWork->Details[
            ex.from->09.00:Real;
            ex.to->17.00: Real];
        ex.phones->Tel[
           ex.office->0235;
           ex.cell->0357;
           ex.home->7654;
           ex.voicemail->6665];
        ex.onBusiness->Details[
            ex.from->07.30:Real;
            ex.to->08.12:Real];
        ex.expertise->Taxonomy[
            ex.offersExpertise -> ClassicMusic;
            ex.seeksExpertise-> LogicProgramming;
            sesdl.profesionalDevelopment->Taxonomy[
                sesdl.publication->Details[
                  ex.amount->4:Integer];
                sesdl.sabbaticals ->Details[
                  ex.from->5.01:Real;
                  ex.to->7.31:Real]];
            sesdl.projectDevelopment ->Taxonomy[
                ex.desiredSalary->Taxonomy[
                  ex.price->10000:Real;
                  ex.range->0.25:Real];
                sesdl.projectDevelopments ->Details[
                  ex.collaboratesIn->Details[
                    ex.item1->Details[
                      foaf.name->Project1;
                      ex.outcome->Accomplished]]]]]).

% Rule profile of Mary

phonePreference(Mary, ?Time, office) :-
    greaterThanOrEqual(?Time, 8.00:Real),
```

```
    lessThan(?Time, 11.00:Real).
phonePreference(Mary, ?Time, office) :-
    greaterThanOrEqual(?Time, 12.00:Real),
    lessThan(?Time, 17.00:Real).
phonePreference(Mary, ?Time, cell) :-
    greaterThanOrEqual(?Time, 11.00:Real),
    lessThan(?Time, 12.00:Real).
phonePreference(Mary, ?Time, cell) :-
    greaterThanOrEqual(?Time, 17.00:Real),
    lessThan(?Time, 18.00:Real).
phonePreference(Mary, ?Time, home) :-
    greaterThanOrEqual(?Time, 18.00:Real),
    lessThan(?Time, 20.00:Real).
phonePreference(Mary, ?Time, voicemail) :-
    greaterThanOrEqual(?Time, 20.00:Real),
    lessThan(?Time, 24.00:Real).
phonePreference(Mary, ?Time, voicemail) :-
    greaterThanOrEqual(?Time, 0.00:Real),
    lessThan(?Time, 8.00:Real).
```

```
% URI of POSL FOAF page: www.ruleml.org/usecases/foaf/Sue.posl
% URI of RuleML FOAF page: www.ruleml.org/usecases/foaf/Sue.ruleml

%% Fact profile of Sue
```

```
foaf.person(Sue^
% Should have page URI: .../Sue/index.html
        foaf.name->Sue;
        foaf.title->Dr;
        foaf.membershipClass->ComputMusic;
        foaf.mbox->Sue_AT_ComputMusic.com;
        foaf.knows->Annie;
        ex.languages->English;
        ex.workPlace->Canada;
        ex.address->Ottawa;
        ex.occupation->professor;
        ex.atWork->Details[
            ex.from->09.00:Real;
            ex.to->17.00: Real];
        ex.phones->Tel[
           ex.office->0235;
           ex.cell->0357;
           ex.home->7654;
           ex.voicemail->6665];
        ex.onBusiness->Details[
            ex.from->07.30:Real;
            ex.to->08.12:Real];
        ex.expertise->Taxonomy[
            ex.offersExpertise -> CountryMusic;
            ex.seeksExpertise-> LogicProgramming;
```

```
            sesdl.profesionalDevelopment->Taxonomy[
                sesdl.publication->Details[
                  ex.amount->4:Integer];
                sesdl.sabbaticals ->Details[
                  ex.from->5.01:Real;
                  ex.to->7.31:Real]];
            sesdl.projectDevelopment ->Taxonomy[
                ex.desiredSalary->Taxonomy[
                  ex.price->8000:Real;
                  ex.range->0.2:Real];
                sesdl.projectDevelopments ->Details[
                  ex.collaboratesIn->Details[
                    ex.item1->Details[
                      foaf.name->Project1;
                      ex.outcome->Accomplished]]]]]).

% Rule profile of Sue

% Phone preference of Sue during different time of the day, attached
% to Sue's profile
phonePreference(Sue, ?Time, office) :-
    greaterThanOrEqual(?Time, 8.00:Real),
    lessThan(?Time, 11.00:Real).
phonePreference(Sue, ?Time, office) :-
    greaterThanOrEqual(?Time, 12.00:Real),
    lessThan(?Time, 17.00:Real).
phonePreference(Sue, ?Time, cell) :-
    greaterThanOrEqual(?Time, 11.00:Real),
    lessThan(?Time, 12.00:Real).
phonePreference(Sue, ?Time, cell) :-
    greaterThanOrEqual(?Time, 17.00:Real),
    lessThan(?Time, 18.00:Real).
phonePreference(Sue, ?Time, home) :-
    greaterThanOrEqual(?Time, 18.00:Real),
    lessThan(?Time, 20.00:Real).
phonePreference(Sue, ?Time, voicemail) :-
    greaterThanOrEqual(?Time, 20.00:Real),
    lessThan(?Time, 24.00:Real).
phonePreference(Sue, ?Time, voicemail) :-
    greaterThanOrEqual(?Time, 0.00:Real),
    lessThan(?Time, 8.00:Real).

%%%% Acting as Computer Science Expert

% URI of the POSL FOAF page: www.ruleml.org/usecases/foaf/Peter.posl
% URI of the RuleML FOAF page: www.ruleml.org/usecases/foaf/Peter.ruleml

%% Fact profile of Peter
```

```
foaf.person(Peter^
% Should have page URI: .../Peter/index.html
        foaf.name->Peter;
        foaf.title->Dr;
        foaf.membershipClass->ComputMusic;
        foaf.mbox->Peter_AT_ComputMusic.com;
        foaf.knows->Ann;
        ex.languages->English;
        ex.workPlace->Canada;
        ex.address->Calgary;
        ex.occupation->professor;
        ex.atWork->Details[
            ex.from->09.00:Real;
            ex.to->17.00: Real];
        ex.phones->Tel[
           ex.office->0235;
           ex.cell->0357;
           ex.home->7654;
           ex.voicemail->6665];
        ex.onBusiness->Details[
            ex.from->07.30:Real;
            ex.to->08.12:Real];
        ex.expertise->Taxonomy[
            ex.offersExpertise -> LogicProgramming;
            ex.seeksExpertise-> PopMusic;
            sesdl.personalDevelopment->Taxonomy[
              sesdl.communication->Taxonomy[
                foaf.name->synchronousCommunication[
                  foaf.name->mailingLists]];
              sesdl.training->Taxonomy[
                ex.learningMethod->deepLearning;
                ex.workDuration->4.0:Real;
                ex.expertiseType->teachingSkills;
                ex.student->Taxonomy[
                  ex.PHD->1:Integer;
                  ex.Master->3:Integer;
                  ex.Undergrad->4:Integer];
                  ex.area->LogicProgramming]];
          sesdl.profesionalDevelopment->Taxonomy[
            sesdl.mentors ->Taxonomy[
              sesdl.experiment->Details[
                ex.item1->Details[
                  foaf.name->Exp1;
                  ex.teachingMethod->peerTeaching;
                  ex.level->Graduate;
                  ex.area->Deduction;
                  ex.studentNum->20:Integer]];
                  ex.totalTeachingHours->30:Integer];
            sesdl.lectures-> Details[
```

```
  sesdl.webcasting->Details[
     ex.amount->30:Integer;
     ex.item1->Details[
       foaf.name->Lecture1;
       ex.area->Deduction;
       ex.range->International;
       ex.location->Paris]];
  sesdl.radioBroadcasting ->Details[
     ex.amount->100:Integer;
     ex.item2->Details[
       foaf.name->Lecture2;
       ex.area->Deduction;
       ex.range->International;
       ex.participants->100:Integer]]];
sesdl.courses->Details[
  ex.item1->Details[
    foaf.name->Course1;
    ex.mode->online;
    ex.level->Undergraduate;
    ex.area->LogicProgramming;
    ex.studentNum->30:Integer]];
sesdl.qualifications->Taxonomy[
  sesdl.degrees->PHD];
sesdl.publication->Details[
  sesdl.peerReviewed ->Details[
    ex.item1->Details[
          bibtex.title->Article1;
          ex.area->LogicProgramming;
          ex.type->JournalPaper;
          bibtex.bookTitle->Journal1;
          bibtex.year->1991:Integer;
          bibtex.pages->"12-16";
          bibtex.month->Sep;
          sesdl.intellectualProperty->Details[
            sesdl.copyright->Publisher1]];
    ex.item2->Details[
          bibtex.title->Article2;
          ex.area->MachineLearning;
          ex.type->ConferencePaper;
          bibtex.bookTitle->Journal3;
          bibtex.year->2000:Integer;
          bibtex.pages->"32-43";
          bibtex.month->Oct;
          sesdl.intellectualProperty->Details[
            sesdl.copyright ->Publisher2]];
    ex.item3->Details[
          bibtex.title->Article3;
          ex.area->LogicProgramming;
          ex.type->JournalPaper;
```

```
                    bibex.bookTitle->Journal1;
                    bibtex.year->2005:Integer;
                    bibtex.pages->"21-29";
                    bibtex.month->Feb;
                    sesdl.intellectualProperty->Details[
                      sesdl.copyright ->Publisher1]];
                ex.item4->Details[
                    bibtex.title->Article4;
                    ex.area->Deduction;
                    ex.type->ConferencePaper;
                    bibex.bookTitle->Journal8;
                    bibtex.year->2005:Integer;
                    bibtex.pages->"25-33";
                    bibtex.month->May;
                    sesdl.intellectualProperty->Details[
                      sesdl.copyright ->Publisher2]]];
              ex.amount->4:Integer];
          sesdl.sabbaticals ->Details[
                ex.from->5.01:Real;
                ex.to->7.31:Real]];
        sesdl.projectDevelopment ->Taxonomy[
          sesdl.evaluation->Taxonomy[
            ex.type->summativeAssessment];
          ex.desiredSalary->Taxonomy[
            ex.price->5000:Real;
            ex.range->0.1:Real];
          sesdl.projectDevelopments ->Details[
            ex.collaboratesIn->Details[
                  ex.item1->Details[
                    foaf.name->Project1;
                    ex.mode->mixedMode;
                    ex.collaborate->Details[
                      ex.type->groupWork;
                      ex.amount->8:Integer;
                      ex.funds->15000:Integer];
                    ex.area->MachineLearning;
                    ex.range->Local;
                    ex.position->TeamLeader;
                    ex.outcome->Accomplished]]]]]).

% URI of the POSL FOAF page: www.ruleml.org/usecases/foaf/John.posl
% URI of the RuleML FOAF page: www.ruleml.org/usecases/foaf/John.ruleml

%% Fact profile of John

foaf.person(John^
% Should have page URI: .../John/index.html
        foaf.name->John;
        foaf.title->Dr;
```

```
        foaf.membershipClass->ComputMusic;
        foaf.mbox->John_AT_ComputMusic.com;
        foaf.knows->Ann;
        ex.languages->English;
        ex.workPlace->Canada;
        ex.address->Moncton;
        ex.occupation->professor;
        ex.atWork->Details[
            ex.from->09.00:Real;
            ex.to->17.00: Real];
        ex.phones->Tel[
           ex.office->0235;
           ex.cell->0357;
           ex.home->7654;
           ex.voicemail->6665];
        ex.onBusiness->Details[
            ex.from->07.30:Real;
            ex.to->08.12:Real];
        ex.expertise->Taxonomy[
            ex.offersExpertise -> LogicProgramming;
            ex.seeksExpertise-> CountryMusic;
            sesdl.profesionalDevelopment->Taxonomy[
              sesdl.publication->Details[
                  ex.amount->4:Integer];
              sesdl.sabbaticals ->Details[
                  ex.from->5.01:Real;
                  ex.to->7.31:Real]];
            sesdl.projectDevelopment ->Taxonomy[
              ex.desiredSalary->Taxonomy[
                  ex.price->7000:Real;
                  ex.range->0.05:Real];
              sesdl.projectDevelopments ->Details[
                  ex.collaboratesIn->Details[
                    ex.item1->Details[
                      foaf.name->Project1;
                      ex.outcome->InProgress]]]]]).

% URI of the POSL FOAF page: www.ruleml.org/usecases/foaf/Hart.posl
% URI of the RuleML FOAF page: www.ruleml.org/usecases/foaf/Hart.ruleml

%% Fact profile of Hart

foaf.person(Hart^
% Should have page URI: .../Hart/index.html
        foaf.name->Hart;
        foaf.title->Dr;
        foaf.membershipClass->ComputMusic;
        foaf.mbox->Hart_AT_ComputMusic.com;
        foaf.knows->Sue;
        ex.languages->English;
```

```
        ex.workPlace->Canada;
        ex.address->Montreal;
        ex.occupation->professor;
        ex.atWork->Details[
            ex.from->09.00:Real;
            ex.to->17.00: Real];
        ex.phones->Tel[
            ex.office->0235;
            ex.cell->0357;
            ex.home->7654;
            ex.voicemail->6665];
        ex.onBusiness->Details[
            ex.from->07.30:Real;
            ex.to->08.12:Real];
        ex.expertise->Taxonomy[
            ex.offersExpertise -> LogicProgramming;
            ex.seeksExpertise-> PopMusic;
            sesdl.profesionalDevelopment->Taxonomy[
              sesdl.publication->Details[
                  ex.amount->4:Integer];
              sesdl.sabbaticals ->Details[
                  ex.from->5.01:Real;
                  ex.to->7.31:Real]];
            sesdl.projectDevelopment ->Taxonomy[
              ex.desiredSalary->Taxonomy[
                  ex.price->2500:Real;
                  ex.range->0.15:Real];
              sesdl.projectDevelopments ->Details[
                  ex.collaboratesIn->Details[
                    ex.item1->Details[
                      foaf.name->Project1;
                      ex.outcome->Accomplished]]]]]).

% URI of the POSL FOAF page: www.ruleml.org/usecases/foaf/Karen.posl
% URI of the RuleML FOAF page: www.ruleml.org/usecases/foaf/Karen.ruleml

%% Fact profile of Karen

foaf.person(Karen^
% Should have page URI: .../Karen/index.html
        foaf.name->Hart;
        foaf.title->Dr;
        foaf.membershipClass-> CBS;
        foaf.mbox->Karen_AT_ComputMusic.com;
        foaf.knows->Hart;
        ex.languages->English;
        ex.workPlace->Canada;
        ex.address->Ottawa;
        ex.occupation->professor;
        ex.atWork->Details[
```

```
            ex.from->09.00:Real;
            ex.to->17.00: Real];
         ex.phones->Tel[
            ex.office->5284;
            ex.cell->0125;
            ex.home->3838;
            ex.voicemail->1985];
         ex.onBusiness->Details[
            ex.from->05.30:Real;
            ex.to->06.12:Real];
         ex.expertise->Taxonomy[
            ex.offersExpertise -> LogicProgramming;
            ex.seeksExpertise-> CountryMusic;
            sesdl.profesionalDevelopment->Taxonomy[
              sesdl.publication->Details[
                ex.amount->4:Integer];
              sesdl.sabbaticals ->Details[
                  ex.from->5.01:Real;
                  ex.to->7.31:Real]];
            sesdl.projectDevelopment ->Taxonomy[
              ex.desiredSalary->Taxonomy[
                ex.price->2500:Real;
                ex.range->0.15:Real];
              sesdl.projectDevelopments ->Details[
                ex.collaboratesIn->Details[
                    ex.item1->Details[
                      foaf.name->Project1;
                      ex.outcome->Accomplished]]]]).
% URI of the POSL FOAF page: www.ruleml.org/usecases/foaf/Annie.posl
% URI of the RuleML FOAF page: www.ruleml.org/usecases/foaf/Annie.ruleml

%% Fact profile of Annie

foaf.person(Annie^
% Should have page URI: .../Annie/index.html
        foaf.name->Annie;
        foaf.title->Dr;
        foaf.membershipClass->ComputMusic;
        foaf.mbox->Annie_AT_ComputMusic.com;
        foaf.knows->Hart;
        ex.languages->English;
        ex.workPlace->Canada;
        ex.address->SaintJohn;
        ex.occupation->professor;
        ex.atWork->Details[
            ex.from->09.00:Real;
            ex.to->17.00: Real];
        ex.phones->Tel[
           ex.office->0235;
           ex.cell->0357;
```

```
           ex.home->7654;
           ex.voicemail->6665];
        ex.onBusiness->Details[
            ex.from->07.30:Real;
            ex.to->08.12:Real];
        ex.expertise->Taxonomy[
            ex.offersExpertise -> LogicProgramming;
            ex.seeksExpertise-> ClassicMusic;
            sesdl.profesionalDevelopment->Taxonomy[
                sesdl.publication->Details[
                  ex.amount->4:Integer];
                sesdl.sabbaticals ->Details[
                  ex.from->5.01:Real;
                  ex.to->7.31:Real]];\
            sesdl.projectDevelopment ->Taxonomy[
                ex.desiredSalary->Taxonomy[
                  ex.price->12000:Real;
                  ex.range->0.3:Real];
                sesdl.projectDevelopments ->Details[
                  ex.collaboratesIn->Details[
                    ex.item1->Details[
                      foaf.name->Project1;
                      ex.outcome->Accomplished]]]]]).

% URI of the POSL FOAF page: www.ruleml.org/usecases/foaf/Amy.posl
% URI of the RuleML FOAF page: www.ruleml.org/usecases/foaf/Amy.ruleml

%% Fact profile of Amy

foaf.person(Amy^
% Should have page URI: .../Amy/index.html
        foaf.name->Amy;
        foaf.title->Dr;
        foaf.membershipClass->ComputMusic;
        foaf.mbox->Amy_AT_ComputMusic.com;
        foaf.knows->Hart;
        ex.languages->English;
        ex.workPlace->Canada;
        ex.address->SaintJohn;
        ex.occupation->professor;
        ex.atWork->Details[
            ex.from->09.00:Real;
            ex.to->17.00: Real];
        ex.phones->Tel[
           ex.office->0235;
           ex.cell->0357;
           ex.home->7654;
           ex.voicemail->6665];
        ex.onBusiness->Details[
```

```
            ex.from->07.30:Real;
            ex.to->08.12:Real];
        ex.expertise->Taxonomy[
            ex.offersExpertise -> LogicProgramming;
            ex.seeksExpertise-> ClassicMusic;
            sesdl.profesionalDevelopment->Taxonomy[
                sesdl.publication->Details[
                  ex.amount->14:Integer];
                sesdl.sabbaticals ->Details[
                  ex.from->5.01:Real;
                  ex.to->7.31:Real]];
            sesdl.projectDevelopment ->Taxonomy[
                ex.desiredSalary->Taxonomy[
                  ex.price->12000:Real;
                  ex.range->0.3:Real];
                sesdl.projectDevelopments ->Details[
                  ex.collaboratesIn->Details[
                    ex.item1->Details[
                      foaf.name->Project1;
                      ex.outcome->Accomplished]]]]]).

% URI of the POSL FOAF page: www.ruleml.org/usecases/foaf/Julia.posl
% URI of the RuleML FOAF page: www.ruleml.org/usecases/foaf/Julia.ruleml

%% Fact profile of Julia

foaf.person(Julia ^
% Should have page URI: .../Julia/index.html
        foaf.name->Julia;
        foaf.title->Dr;
        foaf.membershipClass->CompanyA;
        foaf.mbox-> Julia_AT_CompanyA.com;
        foaf.knows->Hart;
        ex.languages->English;
        ex.workPlace->Canada;
        ex.address->SaintJohn;
        ex.occupation->professor;
        ex.atWork->Details[
            ex.from->09.00:Real;
            ex.to->17.00: Real];
        ex.phones->Tel[
           ex.office->0235;
           ex.cell->0357;
           ex.home->7654;
           ex.voicemail->6665];
        ex.onBusiness->Details[
            ex.from->07.30:Real;
            ex.to->08.12:Real];
        ex.expertise->Taxonomy[
```

```
ex.offersExpertise -> LogicProgramming;
ex.seeksExpertise-> ClassicMusic;
sesdl.profesionalDevelopment->Taxonomy[
    sesdl.publication->Details[
      ex.amount->4:Integer];
    sesdl.sabbaticals ->Details[
      ex.from->5.01:Real;
      ex.to->7.31:Real]];
sesdl.projectDevelopment ->Taxonomy[
    ex.desiredSalary->Taxonomy[
      ex.price->12000:Real;
      ex.range->0.3:Real];
    sesdl.projectDevelopments ->Details[
      ex.collaboratesIn->Details[
        ex.item1->Details[
          foaf.name->Project1;
          ex.outcome->Proposed]]]]]).
```

Appendix B: Agent Profiles

This appendix gives the agent profiles of globally stored facts.

```
% Namespace URIs as for the expert profiles

% Profiles of rating agents

 staffRating(RateExpert^Lucy,
            Rating[sesdl.personalDevelopment->3.5:Real;
                   sesdl.profesionalDevelopment->4.0:Real;
                   sesdl.projectDevelopments->4.5:Real]).
 staffRating(RateExpert^Mary,
            Rating[sesdl.personalDevelopment->4.0:Real;
                   sesdl.profesionalDevelopment->4.0:Real;
                   sesdl.projectDevelopments->5.0:Real]).
 staffRating(RateExpert^John,
            Rating[sesdl.personalDevelopment->5.0:Real;
                   sesdl.profesionalDevelopment->5.0:Real;
                   sesdl.projectDevelopments->5.0:Real]).
 staffRating(RateExpert^Peter,
            Rating[sesdl.personalDevelopment->4.5:Real;
                   sesdl.profesionalDevelopment->4.5:Real;
                   sesdl.projectDevelopments->4.5:Real]).
 staffRating(RateExpert^Hart,
            Rating[sesdl.personalDevelopment->4.5:Real;
                   sesdl.profesionalDevelopment->3.5:Real;
                   sesdl.projectDevelopments->4.0:Real]).
 staffRating(RateExpert^Annie,
            Rating[sesdl.personalDevelopment->5.0:Real;
                   sesdl.profesionalDevelopment->4.5:Real;
                   sesdl.projectDevelopments->4.5:Real]).
 staffRating(RateExpert^Amy,
            Rating[sesdl.personalDevelopment->5.0:Real;
                   sesdl.profesionalDevelopment->5.0:Real;
                   sesdl.projectDevelopments->5.0:Real]).
 staffRating(RateExpert^Sue,
            Rating[sesdl.personalDevelopment->3.5:Real;
                   sesdl.profesionalDevelopment->3.0:Real;
                   sesdl.projectDevelopments->5.0:Real]).
 staffRating(RateExpert^Julia,
```

```
            Rating[sesdl.personalDevelopment->4.0:Real;
                   sesdl.profesionalDevelopment->5.0:Real;
                   sesdl.projectDevelopments->4.5:Real]).
 staffRating(RateExpert^Karen,
            Rating[sesdl.personalDevelopment->4.7:Real;
                   sesdl.profesionalDevelopment->4.9:Real;
                   sesdl.projectDevelopments->4.3:Real]).

% staffRating(Portal4Experts^Lucy,
%             Rating[sesdl.personalDevelopment->5.0:Real;
%                    sesdl.profesionalDevelopment->4.0:Real;
%                    sesdl.projectDevelopments->4.5:Real]).
% ... ...
% staffRating(Portal4Experts^Karen,
%             Rating[sesdl.personalDevelopment->5.0:Real;
%                    sesdl.profesionalDevelopment->4.0:Real;
%                    sesdl.projectDevelopments->4.5:Real]).

% Longitude and latitude of selected cities in Canada
% Source from Qiblih Query Page
% (http://www.bcca.org/misc/qiblih/latlong.html)


 address(Qiblih^
 % Should have URI http://www.bcca.org/misc/qiblih/latlong.html
    Fredericton, Location[
         latitude->
            Detail[degree->45:Real;minute->52:Real];
         longitude->
            Detail[degree->66:Real;minute->32:Real]]).
 address(Qiblih^Moncton, Location[
         latitude->
            Detail[degree->46:Real;minute->7:Real];
         longitude->
            Detail[degree->64:Real;minute->41:Real]]).
 address(Qiblih^SaintJohn, Location[
         latitude->
            Detail[degree->45:Real;minute->19:Real];
         longitude->
            Detail[degree->65:Real;minute->53:Real]]).
 address(Qiblih^Edmundston, Location[
         latitude->
            Detail[degree->47:Real;minute->22:Real];
         longitude->
            Detail[degree->68:Real;minute->20:Real]]).
 address(Qiblih^Chatham, Location[
         latitude->
            Detail[degree->47:Real;minute->1:Real];
         longitude->
            Detail[degree->65:Real;minute->27:Real]]).
```

```
 address(Qiblih^Campbellton, Location[
         latitude->
            Detail[degree->48:Real;minute->0:Real];
         longitude->
            Detail[degree->66:Real;minute->40:Real]]).
 address(Qiblih^Calgary, Location[
         latitude->
            Detail[degree->51:Real;minute->6:Real];
         longitude->
            Detail[degree->114:Real;minute->1:Real]]).
 address(Qiblih^Vancouver, Location[
         latitude->
            Detail[degree->49:Real;minute->11:Real];
         longitude->
            Detail[degree->123:Real;minute->10:Real]]).
 address(Qiblih^Ottawa, Location[
         latitude->
            Detail[degree->45:Real;minute->19:Real];
         longitude->
            Detail[degree->75:Real;minute->40:Real]]).
 address(Qiblih^Toronto, Location[
         latitude->
            Detail[degree->43:Real;minute->41:Real];
         longitude->
            Detail[degree->79:Real;minute->38:Real]]).
 address(Qiblih^Montreal, Location[
         latitude->
            Detail[degree->45:Real;minute->28:Real];
         longitude->
            Detail[degree->73:Real;minute->45:Real]]).
% Canadian weekends and holidays
 weekend(Saturday).
 weekend(Sunday).

 holiday(1.01:Real).
 holiday(4.10:Real).
 holiday(5.22:Real).
 holiday(7.01:Real).
 holiday(8.01:Real).
 holiday(9.07:Real).
 holiday(10.14:Real).
 holiday(11.11:Real).
 holiday(12.24:Real).
 holiday(12.25:Real).
 holiday(12.26:Real).

% Auxiliary relation inInterval
 inInterval(?Var, ?LowerBound, ?UpperBound) :-
    greaterThanOrEqual(?Var, ?LowerBound),
    lessThanOrEqual(?Var, ?UpperBound).
```

```
% Fictitious Canadian Companies

canadianOrg(FreGroup).
canadianOrg(bestGroup).
canadianOrg(CBS).
canadianOrg(CSConsult).
canadianOrg(ComputMusic).
canadianOrg(CrossCanada).

% Meeting history between experts

meetingHistory(Lucy, Hart).
meetingHistory(Mary, Hart).
meetingHistory(Lucy, Amy).
meetingHistory(Peter, Lucy).
meetingHistory(Peter, Jessica).
```

Appendix C: Rule Sets

The CollaborationDecision, FindXpRT and collaborationMode programs are given below:

```
% Selecting leaf values from a person's fact profile
 knows(?Peer1, ?Peer2) :-
    foaf.person(?Peer1^foaf.knows->?Peer2!?).

 offersExpertise(?Peer, ?Expertise) :-
    foaf.person(?Peer^ex.expertise->Taxonomy[
        ex.offersExpertise ->?Expertise!?]!?).
 seeksExpertise(?Peer, ?Expertise) :-
    foaf.person(?Peer^ex.expertise->Taxonomy[
        ex.seeksExpertise ->?Expertise!?]!?).

 getPlace(?Peer, ?Place) :-
    foaf.person(?Peer^ex.workPlace->?Place!?).
 getAddress(?Person, ?Address) :-
    foaf.person(?Person^ex.address->?Address!?).
 getCompany(?Peer, ?Company) :-
    foaf.person(?Peer^foaf.membershipClass->?Company!?).
 getLanguages(?Peer, ?Language) :-
    foaf.person(?Peer^ex.languages->?Language!?).
 getWorkDuration(?Peer, ?Year) :-
    foaf.person(?Peer^ex.expertise->Taxonomy[
        sesdl.personalDevelopment->Taxonomy[
            sesdl.training->Taxonomy[
            ex.workDuration->?Year!?]!?]!?]!?).
 getWorkStartHour(?Peer, ?Time) :-
    foaf.person(?Peer^ex.atWork->Details[ex.from->?Time!?]!?).
 getWorkEndHour(?Peer, ?Time) :-
    foaf.person(?Peer^ex.atWork->Details[ex.to->?Time!?]!?).
 getBusinessStartDate(?Peer, ?Date) :-
    foaf.person(?Peer^ex.onBusiness->Details[ex.from->?Date!?]!?).
 getBusinessEndDate(?Peer, ?Date) :-
    foaf.person(?Peer^ex.onBusiness->Details[ex.to->?Date!?]!?).
 getSabbFrom(?Peer, ?Date):-
    foaf.person(?Peer^ex.expertise->Taxonomy[
        sesdl.profesionalDevelopment->Taxonomy[
            sesdl.sabbaticals->Details[ex.from->?Date!?]!?]!?]!?).
 getSabbEnd(?Peer, ?Date):-
```

```
   foaf.person(?Peer^ex.expertise->Taxonomy[
       sesdl.profesionalDevelopment->Taxonomy[
           sesdl.sabbaticals->Details[ex.to->?Date!?]!?]!?]!?).
getPublication(?Peer, ?Amount) :-
   foaf.person(?Peer^ex.expertise->Taxonomy[
       sesdl.profesionalDevelopment->Taxonomy[
           sesdl.publication->Details[
               ex.amount->?Amount!?]!?]!?]!?).

getPhoneNumber(?Peer, office, ?PhoneNumber) :-
   foaf.person(?Peer^ex.phones-> Tel[ex.office->? PhoneNumber!?]!?).
getPhoneNumber(?Peer, home, ?PhoneNumber) :-
   foaf.person(?Peer^ex.phones-> Tel[ex.cell->? PhoneNumber !?]!?).
getPhoneNumber(?Peer, cell, ?PhoneNumber) :-
   foaf.person(?Peer^ex.phones-> Tel[ex.home->? PhoneNumber!?]!?).
getPhoneNumber(?Peer, voicemail, ?PhoneNumber) :-
   foaf.person(?Peer^ex.phones-> Tel[ex.voicemail->? PhoneNumber!?]!?).

getProjectName(?Peer,?Name):-
   foaf.person(?Peer^ex.expertise->Taxonomy[
   sesdl.projectDevelopment->Taxonomy[
   sesdl.projectDevelopments->Details[
   ex.collaboratesIn->Details[ex.item1->Details[
   foaf.name->?Name!?]!?]!?]!?]!?]!?).
getProOutcome(?Peer, ?Outcome):-
   foaf.person(?Peer^ex.expertise->Taxonomy[
   sesdl.projectDevelopment->Taxonomy[
   sesdl.projectDevelopments->Details[ex.collaboratesIn->Details[
   ex.item1->Details[
   foaf.name->?Name;
   ex.outcome->?Outcome!?]!?]!?]!?]!?]!?).

getDesiredPrice(?Peer, ?Price) :-
   foaf.person(?Peer^ex.expertise->Taxonomy[
       sesdl.projectDevelopment ->Taxonomy[
           ex.desiredSalary->Taxonomy[
               ex.price->?Price!?]!?]!?]!?).
getDesiredPriceRange(?Peer, ?Range) :-
   foaf.person(?Peer^ex.expertise->Taxonomy[
       sesdl.projectDevelopment ->Taxonomy[
           ex.desiredSalary->Taxonomy[
               ex.range->?Range!?]!?]!?]!?).

% Selecting values from centralized facts

getPersonalRating(?RatingAgent, ?Person, ?PersonalRating) :-
   staffRating(?RatingAgent^?Person,
   Rating[sesdl.personalDevelopment->?PersonalRating!?]).
getResearchRating(?RatingAgent, ?Person, ?PersonalRating) :-
   staffRating(?RatingAgent^?Person,
```

```
    Rating[sesdl.profesionalDevelopment->?PersonalRating!?]).
 getProfessionRating(?RatingAgent, ?Person, ?PersonalRating) :-
    staffRating(?RatingAgent^?Person,
    Rating[sesdl.projectDevelopments->?PersonalRating!?]).

% From centralized stored fact about the longitude and latitude
% of Canadian cities
 getLatiLoc(?Place, ?Degree, ?Minute) :-
    address(Qiblih^?Place,  Location[
    latitude->Detail[degree->?Degree; minute->?Minute]!?]).
 getLongtiLoc(?Place, ?Degree, ?Minute) :-
    address(Qiblih^?Place,  Location[
    longitude->Detail[degree->?Degree;  minute->?Minute]!?]).


% Compute the overall rating of each person from a certain
% rating agent
 getRating(?RatingAgent, ?Person, ?AverageRating) :-
    getPersonalRating(?RatingAgent,?Person, ?Rating1),
    getResearchRating(?RatingAgent,?Person, ?Rating2),
    getProfessionRating(?RatingAgent, ?Person, ?Rating3),
    add(?Temp, ?Rating1:Real, ?Rating2:Real),
    add(?Rating, ?Temp:Real, ?Rating3:Real),
    divide(?AverageRating, ?Rating:Real, 3.0:Real).

% Get the phone number according different time
 call(?Peer1, ?Peer2, ?Time, ?PhoneNumber) :-
    phonePreference(?Peer2, ?Time, ?Preference),
    getPhoneNumber(?Peer2, ?Preference, ?PhoneNumber).

% Get email address
 email(?Peer1, ?Peer2, ?EmailAddress) :-
    foaf.person(?Peer2^foaf.mbox->?EmailAddress!?).

% Primary expert's criteria for his/her potential collaborator
% according to Figure 5.4
 checkLocation(?Peer2, ?Peer1) :-
    getPlace(?Peer1, ?Place),
    equal(?Place, Canada).
 expertiseMatch(?Peer1, ?Peer2, ?Expertise) :-
    offersExpertise(?Peer2, ?Expertise),
    seeksExpertise(?Peer1, ?Expertise),
    notEqual(?Peer1, ?Peer2).
 checkPublication(?Peer2, ?Peer1) :-
    getPublication(?Peer1, ?Amount),
    greaterThan(?Amount, 3:Integer).
 checkWorkDuration(?Peer2, ?Peer1) :-
    getWorkDuration(?Peer1, ?Year),
    greaterThan(?Year, 2.0:Real).
```

```
 checkRating(?Peer2, ?Peer1) :-
    getPersonalRating(?RatingAgent, ?Peer1, ?Rating1),
    getResearchRating(?RatingAgent, ?Peer1, ?Rating2),
    getProfessionRating(?RatingAgent, ?Peer1, ?Rating3),
    add(?Temp, ?Rating1:Real, ?Rating2:Real),
    add(?Rating, ?Temp:Real, ?Rating3:Real),
    divide(?Result, ?Rating:Real, 3.0:Real),
    greaterThan(?Result, 3.0:Real).

% Decision of whether to collaborate or not
 CollaborationDecision(?Peer2, ?Peer1, ?Expertise) :-
    expertiseMatch(?Peer2, ?Peer1, ?Expertise),
    checkLocation(?Peer2, ?Peer1),
    checkPublication(?Peer2, ?Peer1),
    checkWorkDuration(?Peer2, ?Peer1),
    checkRating(?Peer2, ?Peer1).

% Get the phone number according different time
 call(?Peer1, ?Peer2, ?Time, ?PhoneNumber) :-
    phonePreference(?Peer2, ?Time, ?Preference),
    getPhoneNumber(?Peer2, ?Preference, ?PhoneNumber).
% Get email address
 email(?Peer1, ?Peer2, ?EmailAddress) :-
    foaf.person(?Peer2^foaf.mbox->?EmailAddress!?).

% Inform the requester after making a decision, either by phone or
% email, depending on if ?Peer1 knows ?Peer2 or not
 informRequester(?Peer1, ?Peer2, ?Time, ?PhoneNumber) :-
    getNetworks(?Peer1, ?Peer2),
    call(?Peer1, ?Peer2, ?Time, ?PhoneNumber).
 informRequester(?Peer1, ?Peer2, ?Time,?EmailAddress) :-
    naf(getNetworks(?Peer1, ?Peer2)),
    email(?Peer1, ?Peer2, ?EmailAddress).

% Professional relation between two persons
 coWorkers(?Peer1, ?Peer2, ?Group) :-
    getCompany(?Peer1, ?Group),
    getCompany(?Peer2, ?Group),
    notEqual(?Peer1, ? Peer2).
 colleagues(?Peer1, ? Peer2, ?Project) :-
    coWorkers(?Peer1, ? Peer2),
    getProjectName(?Peer1, ?Project),
    getProjectName(?Peer2, ?Project).
 collaborates(?Peer1, ?Peer2, ?Project) :-
    busyWith(?Peer1, ?Project),
    busyWith(?Peer2, ?Project).
 collaborated(?Peer1, ?Peer2, ?Project) :-
    colleagues(?Peer1, ?Peer2, ?Project),
    naf(collaborates(?Peer1, ?Peer2, ?Project)).
```

```
 busyWith(?Peer, ?Project) :-
    getProjectName(?Peer, ?Project),
    getProOutcome(?Peer, ?Outcome),
    equal(?Outcome, InProgress).
 busyWith(?Peer, ?Project) :-
    getProjectName(?Peer, ?Project),
    getProOutcome(?Peer, ?Outcome),
    equal(?Outcome, Proposed).

% Secondary expert's criteria for his/her potential collaborator
% according to Figure 5.3
 satisfiedExpert(?Peer1, ?Peer2, ?Peer2Expertise) :-
    colleagues(?Peer1, ?Peer2, ?Project),
    naf(collaborates(?Peer1, ?Peer2, ?Project)),
    expertiseMatch(?Peer1, ?Peer2, ?Peer2Expertise).
 satisfiedExpert(?Peer1, ?Peer2, ?Peer2Expertise) :-
    coWorkers(?Peer1, ?Peer2, ?Group),
    naf(colleagues(?Peer1, ?Peer2, ?Project)),
    naf(collaborates(?Peer1, ?Peer2, ?Project)),
    expertiseMatch(?Peer1, ?Peer2, ?Peer2Expertise).

% According to the criteria of both experts, find the most appropriate
% match, via possible referral.
% Experts' rating is scaled from 0 to 5.0, where greater than
% 3.0 is considered beyond average
 FindXpRT(?Peer1, ?Peer2, ?Peer2, ?Peer1Expertise, ?Peer2Expertise,
          ?UltimateRating, ?UltimateRating, ?Degree) :-
    getRating(?RatingAgent, ?Peer2, ?Peer2Rating),
    greaterThanOrEqual(?Peer2Rating, ?UltimateRating),
    satisfiedExpert(?Peer1, ?Peer2, ?Peer2Expertise),
    naf(busyWith(?Peer2, ?Project)),
    CollaborationDecision(?Peer2, ?Peer1, ?Peer1Expertise).
 FindXpRT(?Peer1, ?Peer2, ?UltimatePeer, ?Peer1Expertise,
          ?Peer2Expertise, ?Rating, ?UltimateRating, ?Degree) :-
    getRating(?RatingAgent, ?Peer2, ?Peer2Rating),
    add(?RatingTmp, ?Rating, 0.5:Real),
    lessThan(?Peer2Rating, ?RatingTmp),
    greaterThanOrEqual(?Peer2Rating, ?Rating),
    satisfiedExpert(?Peer1, ?Peer2, ?Peer2Expertise),
    busyWith(?Peer2, ?Project),
    % cannot do it, but:
    knows(?Peer2, ?ReferralPeer2),
    greaterThan(?Degree, 0:Integer),
    greaterThan(?Rating, 3.5:Real),
    subtract(?DegreeNew, ?Degree, 1:Integer),
    subtract(?RatingNew, ?Rating, 0.5:Real),
    FindXpRT(?Peer1, ?ReferralPeer2, ?UltimatePeer, ?Peer1Expertise,
             ?Peer2Expertise, ?RatingNew, ?UltimateRating, ?DegreeNew).
```

```
 FindXpRT(?Peer1, ?Peer2, ?UltimatePeer, ?Peer1Expertise,
          ?Peer2Expertise, ?Rating, ?UltimateRating, ?Degree) :-
    getRating(?RatingAgent, ?Peer2, ?Peer2Rating),
    add(?RatingTmp, ?Rating, 0.5:Real),
    lessThan(?Peer2Rating, ?RatingTmp),
    greaterThanOrEqual(?Peer2Rating, ?Rating),
    satisfiedExpert(?Peer1, ?Peer2, ?Peer2Expertise),
    naf(CollaborationDecision(?Peer2, ?Peer1, ?Peer1Expertise)),
    % cannot do it, but:
    knows(?Peer2, ?ReferralPeer2),
    greaterThan(?Degree, 0:Integer),
    greaterThan(?Rating, 3.5:Real),
    subtract(?DegreeNew, ?Degree, 1:Integer),
    subtract(?RatingNew, ?Rating, 0.5:Real),
    FindXpRT(?Peer1, ?ReferralPeer2, ?UltimatePeer, ?Peer1Expertise,
             ?Peer2Expertise, ?RatingNew, ?UltimateRating, ?DegreeNew).

% Rules for identifying the collaborationMode
 onSabbatical(?Peer, ?Date) :-
    getSabbFrom(?Peer, ?Date1),
    getSabbEnd(?Peer, ?Date2),
    inInterval(?Date, ?Date1, ?Date2).
 onHoliday(?Peer, ?Date) :-
    getCompany(?Peer, ?Company),
    canadianOrg(?Company),
    holiday(?Date).
 onWeekend(?Peer, ?Day) :-
    getCompany(?Peer, ?Company),
    canadianOrg(?Company),
    weekend(?Day).
 onBusiness(?Peer, ?Date) :-
    getBusinessStartDate(?Peer, ?Date1),
    getBusinessEndDate(?Peer, ?Date2),
    inInterval(?Date, ?Date1, ?Date2).
 atWork(?Peer, ?Date, ?Day) :-
    naf(onSabbatical(?Peer, ?Date)),
    naf(onHoliday(?Peer, ?Date)),
    naf(onBusiness(?Peer, ?Date)),
    naf(onWeekend(?Peer, ?Day)).

 availability(?Peer, ?Date, ?Day, 0.9:Real)  :-
    atWork(?Peer, ?Date,?Day).
 availability(?Peer, ?Date, ?Day, 0.7:Real) :-
    onWeekend(?Peer, ?Day).
 availability(?Peer, ?Date, ?Day, 0.5:Real) :-
    naf(onWeekend(?Peer, ?Day)),
    onHoliday(?Peer, ?Date).
 availability(?Peer, ?Date, ?Day,  0.3:Real) :-
    naf(onWeekend(?Peer, ?Day)),
```

```
    onBusiness(?Peer, ?Date).
 availability(?Peer, ?Date, ?Day, 0.1:Real) :-
    naf(onWeekend(?Peer, ?Day)),
    onSabbatical(?Peer, ?Date).

 scale(?Ranking, veryHigh) :-
    greaterThan(?Ranking, 0.8:Real),
    lessThanOrEqual(?Ranking, 1.0:Real).
 scale(?Ranking, high) :-
    greaterThan(?Ranking, 0.6:Real),
    lessThanOrEqual(?Ranking, 0.8:Real).
 scale(?Ranking, medium) :-
    greaterThan(?Ranking, 0.4:Real),
    lessThanOrEqual(?Ranking, 0.6:Real).
 scale(?Ranking, low) :-
    greaterThan(?Ranking, 0.2:Real),
    lessThanOrEqual(?Ranking, 0.4:Real).
 scale(?Ranking, veryLow) :-
    greaterThan(?Ranking, 0:Real),
    lessThanOrEqual(?Ranking, 0.2:Real).

% Compute longitude and latitude of a Canadian city
 getLatitude(?Place, ?Latitude) :-
    getLatiLoc(?Place, ?Degree, ?Minute),
    divide(?Tmp1, ?Minute, 60:Real),
    add(?Tmp2, ?Degree, ?Tmp1),
    multiply(?Latitude, ?Tmp2, 100:Real).
 getLongtitude(?Place, ?Longtitude) :-
    getLongtiLoc(?Place, ?Degree, ?Minute),
    divide(?Tmp1, ?Minute, 60:Real),
    add(?Tmp2, ?Degree, ?Tmp1),
    multiply(?Longtitude, ?Tmp2, 100:Real).
% Compute the distance between two Canadian cities
 computeDistance(?Distance, ?Address1, ?Address2) :-
    getLatitude(?Address1, ?Latitude1),
    getLongtitude(?Address1, ?Longtitude1),
    getLatitude(?Address2, ?Latitude2),
    getLongtitude(?Address2, ?Longtitude2),
    subtract(?Sub1, ?Latitude1, ?Latitude2),
    pow(?Pow1, ?Sub1, 2:Real),
    subtract(?Sub2, ?Longtitude1, ?Longtitude2),
    pow(?Pow2, ?Sub2, 2:Real),
    add(?Sum, ?Pow1, ?Pow2),
    pow(?Distance, ?Sum, 0.5:Real).
% Compute the distance between two persons' address
 geoDistance(?Peer1, ?Peer2, ?Distance):-
    getAddress(?Peer1, ?Address1),
    getAddress(?Peer2, ?Address2),
    computeDistance(?Distance, ?Address1, ?Address2).
```

```
 languageMatch(?Peer1, ?Peer2, ?Language) :-
    getLanguages(?Peer1, ?Language),
    getLanguages(?Peer2, ?Language).
 properPay(?Peer1, ?Peer2, ?Price) :-
    getDesiredPrice(?Peer2, ?Price1),
    getDesiredPriceRange(?Peer2, ?Range),
    notEqual(?Peer1, ?Peer2),
    multiply(?Temp, ?Price1, ?Range),
    subtract(?LowerBound, ?Price1, ?Temp),
    greaterThanOrEqual(?Price, ?LowerBound).
 collaborationEstablished(?Peer1, ?Peer2, ?Price):-
    properPay(?Peer1, ?Peer2, ?Price),
    languageMatch(?Peer1, ?Peer2, ?Language).

% Suggest an appropriate communication mode to
% collaborators according to the distance between them
 communication(?Peer1, ?Peer2, ?Price, F2F) :-
    collaborationEstablished(?Peer1, ?Peer2, ?Price),
    geoDistance(?Peer1, ?Peer2, ?Distance),
    inInterval(?Distance, 0:Real, 200:Real),
    meetingHistory(?Peer1, ?Peer2).
 communication(?Peer1, ?Peer2, ?Price, VideoTel) :-
    collaborationEstablished(?Peer1, ?Peer2, ?Price),
    geoDistance(?Peer1, ?Peer2, ?Distance),
    inInterval(?Distance, 0:Real, 200:Real),
    naf(meetingHistory(?Peer1, ?Peer2)).
 communication(?Peer1, ?Peer2, ?Price, VoiceTel) :-
    collaborationEstablished(?Peer1, ?Peer2, ?Price),
    geoDistance(?Peer1, ?Peer2, ?Distance),
    inInterval(?Distance, 200:Real, 2000:Real),
    meetingHistory(?Peer1, ?Peer2).
 communication(?Peer1, ?Peer2, ?Price, WebIM) :-
    collaborationEstablished(?Peer1, ?Peer2, ?Price),
    geoDistance(?Peer1, ?Peer2, ?Distance),
    inInterval(?Distance, 200:Real, 2000:Real),
    naf(meetingHistory(?Peer1, ?Peer2)).
 communication(?Peer1, ?Peer2, ?Price, Email) :-
    collaborationEstablished(?Peer1, ?Peer2, ?Price),
    geoDistance(?Peer1, ?Peer2, ?Distance),
    greaterThan(?Distance, 2000:Real),
    meetingHistory(?Peer1, ?Peer2).
 communication(?Peer1, ?Peer2, ?Price, Email) :-
    collaborationEstablished(?Peer1, ?Peer2, ?Price),
    geoDistance(?Peer1, ?Peer2, ?Distance),
    greaterThan(?Distance, 2000:Real),
    naf(meetingHistory(?Peer1, ?Peer2)).

% Provide the proper collaboration mode to the two collaborators
```

```
% according to their availability and distance
 collaborationMode(?Peer1, ?Peer2, ?Mode, ?Date, ?Day, ?Price,
                   ?P1Avail, ?P2Avail) :-
    communication(?Peer1, ?Peer2, ?Price, ?Mode),
    availability(?Peer1, ?Date, ?Day, ?Ranking1),
    availability(?Peer2, ?Date, ?Day, ?Ranking2),
    scale(?Ranking1, ?P1Avail),
    scale(?Ranking2, ?P2Avail),
    notEqual(?Peer1, ?Peer2).

% MatchXpRT finds the Computer Science expert for the music expert
% and suggust the collaboration mode between them
 MatchXpRT(?Peer1, ?UltimatePeer, ?Peer1Expertise,
                ?Peer2Expertise, ?Rating, ?Degree, ?Mode, ?Date,
                ?Day, ?Price) : -
    FindXpRT(?Peer1, ?Peer2, ?UltimatePeer, ?Peer1Expertise,
             ?Peer2Expertise, ?Rating, ?UltimateRating, ?Degree),
    collaborationMode(?Peer1, ?Peer2, ?Mode, ?Date, ?Day, ?Price,
                      ?P1Avail, ?P2Avail).
```

Vita

Candidate's full name:

Jie Li

University attended (with dates and degrees obtained):

Taiyuan University of Technology, P. R. China
September, 2000 - July, 2004
Bachelor of Science
Bachelor of Arts

Publications:

Jie Li, Harold Boley, Virendrakumar C. Bhavsar, Jing Mei, Expert Finding in eCollaboration Using FOAF with RuleML Rules, *MCeTech 2006*, pages 53-65, May 17-19, 2006

Jie Li, Harold Boley, Virendrakumar C. Bhavsar, RuleML FOAF: Web Rules for Social Networking, *RuleML-2006 poster session*, Athens, Geogia, U.S.A, November 10- 11, 2006

Jie Li, Harold Boley, Virendrakumar C. Bhavsar, RuleML FOAF: Web Rules for Social Networking, http://www.cs.unb.ca/itc/ResearchExpo/posters/2006/abs32.pdf, *UNB Computer Science 2006 Research Expo Poster*, April 10, 2006

Jie Li, Harold Boley, Virendrakumar C. Bhavsar, David Hirtle, Jing Mei, Website: RuleML FOAF: *A Use Case for Web-based Social Networking*,
http://www.ruleml.org/usecases/foaf/, March 31, 2006

Jing Mei, Harold Boley, Jie Li, Virendrakumar C. Bhavsar, and Zuoquan Lin, Datalog^{DL}: Datalog Rules Parameterized by Description Logics, *Canadian Semantic Web*, Springer Series: Semantic Web and Beyond , Vol. 2, pages 171-187, 2006

Conference Presentations:

Jie Li, Harold Boley, Virendrakumar C. Bhavsar, Jing Mei, Montreal Conference of eTechnologies 2006, Montreal, Canada, May 17 -19, 2006, *"Expert Finding in eCollaboration Using FOAF with RuleML Rules"*