D2.3.1 Specification of a methodology for syntactic and semantic versioning

Wolf Winkler (National University of Ireland, Galway)

with contributions from:
Max Völkel and York Sure (University of Karlsruhe, Germany)
Vincent Schickel-Zuber and Walter Binder (Artificial Intelligence Laboratory - EPFL)
Vassilis Tzouvaras (ITI-Certh)
Diego Ponte and Chiara Zini and Marco Cruciani and Matteo Bonifacio (University of Trento)
Sebastian Ryszard Kruk (National University of Ireland, Galway, Ireland)
Marcin Synak (Gdansk University of Technology, Poland)

Abstract.
EU-IST Network of Excellence (NoE) IST-2004-507482 KWEB
Deliverable D2.3.1 (WP2.3)

Document Identifier	KWEB/2004/D2.3.1/v1.0
Project	KWEB EU-IST-2004-507482
Version	v1.0
Date	December 15th, 2004
State	final
Distribution	public

Knowledge Web Consortium

This document is part of a research project funded by the IST Programme of the Commission of the European Communities as project number IST-2004-507482.

University of Innsbruck (UIBK) - Coordinator
Institute of Computer Science
Technikerstrasse 13
A-6020 Innsbruck
Austria
Fax: +43(0)5125079872, Phone: +43(0)5125076485/88
Contact person: Dieter Fensel
E-mail address: dieter.fensel@uibk.ac.at

France Telecom (FT)
4 Rue du Clos Courtel
35512 Cesson Sévigné
France. PO Box 91226
Fax: +33 2 99124098, Phone: +33 2 99124223
Contact person : Alain Leger
E-mail address: alain.leger@rd.francetelecom.com

Free University of Bozen-Bolzano (FUB)
Piazza Domenicani 3
39100 Bolzano
Italy
Fax: +39 0471 315649, Phone: +39 0471 315642
Contact person: Enrico Franconi
E-mail address: franconi@inf.unibz.it

Centre for Research and Technology Hellas / Informatics and Telematics Institute (ITI-CERTH)
1st km Thermi - Panorama road
57001 Thermi-Thessaloniki
Greece. Po Box 361
Fax: +30-2310-464164, Phone: +30-2310-464160
Contact person: Michael G. Strintzis
E-mail address: strintzi@iti.gr

National University of Ireland Galway (NUIG)
National University of Ireland
Science and Technology Building
University Road
Galway
Ireland
Fax: +353 91 526388, Phone: +353 87 6826940
Contact person: Christoph Bussler
E-mail address: chris.bussler@deri.ie

Ècole Polythechnique Fédérale de Lausanne (EPFL)
Computer Science Department
Swiss Federal Institute of Technology
IN (Ecublens), CH-1015 Lausanne
Switzerland
Fax: +41 21 6935225, Phone: +41 21 6932738
Contact person: Boi Faltings
E-mail address: boi.faltings@epfl.ch

Freie Universität Berlin (FU Berlin)
Takustrasse 9
14195 Berlin
Germany
Fax: +49 30 83875220, Phone: +49 30 83875223
Contact person: Robert Tolksdorf
E-mail address: tolk@inf.fu-berlin.de

Institut National de Recherche en Informatique et en Automatique (INRIA)
ZIRST - 655 avenue de l’Europe -
Montbonnot Saint Martin
38334 Saint-Ismier
France
Fax: +33 4 7661 5207, Phone: +33 4 7661 5366
Contact person: Jérôme Euzenat
E-mail address: Jerome.Euzenat@inrialpes.fr

Learning Lab Lower Saxony (L3S)
Expo Plaza 1
30539 Hannover
Germany
Fax: +49-511-7629779, Phone: +49-511-76219711
Contact person: Wolfgang Nejdl
E-mail address: nejdl@learninglab.de

The Open University (OU)
Knowledge Media Institute
The Open University
Milton Keynes, MK7 6AA
United Kingdom
Fax: +44 1908 653169, Phone: +44 1908 653506
Contact person: Enrico Motta
E-mail address: e.motta@open.ac.uk

Universidad Politécnica de Madrid (UPM)
Campus de Montegancedo sn
28660 Boadilla del Monte
Spain
Fax: +34-913524819, Phone: +34-913367439
Contact person: Asunción Gómez Pérez
E-mail address: asun@fi.upm.es

University of Liverpool (UniLiv)
Chadwick Building, Peach Street
L697ZF Liverpool
United Kingdom
Fax: +44(151)7943715, Phone: +44(151)7943667
Contact person: Michael Wooldridge
E-mail address: M.J.Wooldridge@csc.liv.ac.uk

University of Sheffield (USFD)
Regent Court, 211 Portobello street
S14DP Sheffield
United Kingdom
Fax: +44 114 2221810, Phone: +44 114 2221891
Contact person: Hamish Cunningham
E-mail address: hamish@dcs.shef.ac.uk

Vrije Universiteit Amsterdam (VUA)
De Boelelaan 1081a
1081HV. Amsterdam
The Netherlands
Fax: +31842214294, Phone: +31204447731
Contact person: Frank van Harmelen
E-mail address: Frank.van.Harmelen@cs.vu.nl

University of Karlsruhe (UKARL)
Institut für Angewandte Informatik und Formale
Beschreibungsverfahren - AIFB
Universität Karlsruhe
D-76128 Karlsruhe
Germany
Fax: +49 721 6086580, Phone: +49 721 6083923
Contact person: Rudi Studer
E-mail address: studer@aifb.uni-karlsruhe.de

University of Manchester (UoM)
Room 2.32. Kilburn Building, Department of Computer
Science, University of Manchester, Oxford Road
Manchester, M13 9PL
United Kingdom
Fax: +44 161 2756204, Phone: +44 161 2756248
Contact person: Carole Goble
E-mail address: carole@cs.man.ac.uk

University of Trento (UniTn)
Via Sommarive 14
38050 Trento
Italy
Fax: +39 0461 882093, Phone: +39 0461 881533
Contact person: Fausto Giunchiglia
E-mail address: fausto@dit.unitn.it

Vrije Universiteit Brussel (VUB)
Pleinlaan 2, Building G10
1050 Brussels
Belgium
Fax: +32 2 6293308, Phone: +32 2 6293308
Contact person: Robert Meersman
E-mail address: robert.meersman@vub.ac.be

Changes

Version	Date	Author	Changes
1.1	10.03.04	Vincent Schickel-Zuber, Walter Binder (EPFL)	Section Usage Scenario
1.2	14.06.04	Vassilis Tzouvaras (ITI-Certh)	Section 3.1 Neurofuzzy Inferencing and Evolution
1.3	17.12.04	Diego Ponte, Chiara Zini, Marco Cruciani, Matteo Bonifacio (UniTn)	Section 3.2 Economic Aspects of Versioning and Evolution
1.4	15.12.04	Max Völkel (UKarl), Wolf Winkler (NUIG)	Chapter 2 Versioning Methodology
1.5	20.12.04	Wolf Winkler (NUIG), York Sure (UKarl)	Chapter 1 Introduction
1.6	20.12.04	Wolf Winkler (NUIG)	Chapter 4 Conclusions

Executive Summary

Knowledge domains and their semantic representations via ontologies are typically subject to change in practical applications. Additionally, engineering of ontologies often takes place in distributed settings where multiple independent users interact. Therefore, change management for ontologies becomes a crucial aspect for any kind of ontology management environment.

One of the main problems is that there are many different ontology languages and no standard versioning system or methodology has arisen yet, that could provide a common way to handle versioning issues.

Therefor this deliverable introduces a new RDF-centric versioning approach and an implementation called *SemVersion*. SemVersion provides structural and semantic versioning for RDF models and RDF-based ontology languages like RDFS. It separates language-specific from language-neutral features in design and implementation. This way SemVersion offers a common approach for already widely used RDF models and a wide range of ontology languages. But SemVersion will also be further developed as an actual versioning system. That means beside dealing with ontology versioning specific topics like calculating diffs, practical aspects like data and metadata management for an applicable system are also considered. The requirements for our system are derived from a practical scenario in the librarian domain, i.e. the MarcOnt scenario.

In some domains, the knowledge is to vague, to be formulated in current ontology languages. Therefore fuzzy knowledge representations and adequate reasoning methods have been investigated in this deliverable. Fuzzy reasoning can capture and exploit uncertain knowledge. Fuzzy ontologies can be versioned by SemVersion as well, as they can be modelled with RDF.

Reasons and patterns of changes in ontologies should also be considered in an ontology versioning context. Therefore this deliverable explains the relations of language, ontologies and economic forces, that describe the mechanisms of usage and change.

Contents

Chapter 1

Introduction

1.1 Towards a general ontology versioning system

For many practical applications, ontologies (cf. [SS04]) can not be seen as static entities, they rather change over time. Support for change management is crucial to support uncontrolled, decentralised and distributed engineering of ontologies. First approaches have been described in [Kle04, Sto04]. But, there is no system yet that functions as a standard versioning system for ontologies like CVS does in the field of software development.

Reviewing the current approaches to ontology versioning we feel that what is currently lacking most, is a thoroughly designed, usable ontology versioning system that meets the basic requirements for ontology versioning and is adaptable for different languages.

An ontology versioning methodology has three main aspects

- *versioning data management*, that is the systematic storage and retrieval of the data that make up the versions and the corresponding metadata
- *versioning protocols and processes*, that is the specification of the interfaces and the rules of interaction in the versioning system
- *specific versioning functionality* for ontologies, e.g. calculating semantic diffs, semantic merging etc.

Unfortunately most contributions to ontology versioning focus only on the third point. This might be one reason, why there is no widely deployed ontology versioning system out there yet.

This deliverable introduces a new RDF-based versioning approach and describes the new versioning system SemVersion that provides versioning for RDF models and RDF-based ontology languages like RDFS, OWL flavours or TRIPLE [SD02]. We present a working methodology accompanied by its implementation in the system SemVersion. The

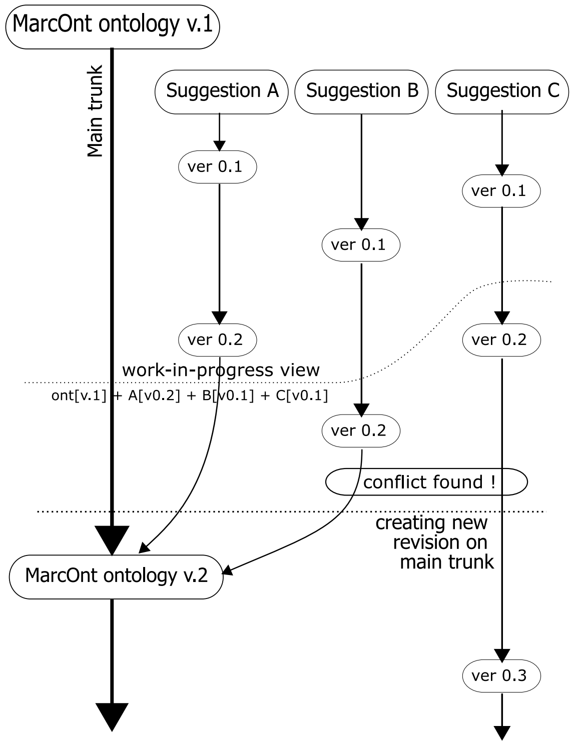

Figure 1.1: Lifelines of main ontology and separate suggestions

methodology and the system provide a well-defined core functionality for ontology versioning. We captured the requirements for the methodology and our system in a real-life scenario from the librarian domain and from a usage scenario for Semantic Web Services.

Our approach is inspired by the classical CVS system for version management of textual documents (e.g. Java code). Core element of our approach is the separation of language-specific features (the diff) from general features (such as structural diff, branch and merge, management of projects and metadata). A speciality of RDF is the usage of so-called blank nodes. As part of our approach we present a method for blank node enrichment which is required for the versioning of such blank nodes.

1.2 Usage scenario 1 – versioning an ontology for digital libraries

With the use of computers computer-based librarian systems became popular. They are used to catalogue and search books and other resources stored in libraries all over the World. Today the Internet is a place where more and more documents of all kinds are published. They are often organised within digital libraries. Searching them often produces poor results and is too slow, because traditional systems are ineffective when adapted to suit the needs of a digital library.

The MarcOnt Initiative[1] is an attempt to address some of these problems by creating an extendable and versatile description system for librarian purposes which will take advantage of the Semantic Web. One result of the MarcOnt Initiative will be a standard ontology for digital libraries that incorporates existing standards like MARC21[2] and Dublin Core[3]. The ontology will be easy to merge with other bibliographic ontologies or to translate to them. It will also support searching[4] and browsing (cf. [Y^+03]) using ontology concepts and semantic user profiles e. g. to deduce personal preferences.

To achieve these goals, there is a need to include a wide community of librarians, computer analysts, software developers, programmers etc. in the development process. Only a community effort can create a recognised standard, as standards are at the end based on social agreements. Such a community needs the appropriate collaborative ontology engineering tools.

The core feature of the MarcOnt Initiative collaboration portal is an integrated ontology builder. Concurrent versions of the MarcOnt ontology are built out of suggestions proposed by community members (see Fig. 1.1). These suggestions themselves can evolve over time. After a voting process, some are applied and a new ontology version is created.

As soon as organisations start to use the current ontology version, the problem arises, how to develop the ontology further while retaining interoperability between systems based on different versions of the MarcOnt ontology. There are two possible solutions: Either by allowing only monotonic extensions or by developing accompanying mapping/translation rules between versions. To be able to do so, the ontology builder must be capable of presenting differences between various versions.

The ontology builder of the MarcOnt portal requires not only a GUI for building the ontology through submitting changes. It also needs the ability to: manage a main trunk of the ontology (M1); manage versions of suggestions (M2); generate snapshots of the main ontology with some suggestions applied (M3); detect and resolve conflicts (M4); add suggestions to the main trunk (M5) and attach mapping/translation rules (M6).

We generalised these requirements in order to: (i) support the CVS core functions, (ii) create a system which can be easily integrated into existing ontology engineering environments such as the ontology builder. Thus our requirements are:

R1 Basic management functions for projects

R2 Retrieve and commit versions, either as full ontologies or as diffs

R3 Branch and merge operations

---

[1] `http://www.marcont.org/`

[2] `http://www.loc.gov/marc/`

[3] `http://dublincore.org/`

[4] e.g. `http://www.jeromedl.org/`

R4 Flexible annotation of versioning artefacts with arbitrary metadata such as “top voted”, “released” or “temporal view”

1.3 Usage scenario 2 – Ontology Versioning for Semantic Web Services

Web Services, at their core, are technologies designed to improve the interoperability between many diverse applications and platforms that exist today (overcoming the limitation of CORBA, RMI, and similar platform dependent technologies). Although, one main objective of Web Services is to allow the automatic composition of existing Web Services in order to create some new services that are able to perform complex tasks (example: making the travel arrangement for a KnowledgeWeb workshop). The syntax of Web Services is pretty well defined and standards have been widely adopted (i.e.: WSDL, UDDI, SOAP) but they lack semantics. This reduces the interoperability between services in a heterogeneous environment like the web.

The Semantic Web allows one to define the semantic of a service via content markup languages, such as OWL and DAML+OIL. Thus, the fundamental component of the Semantic Web will be the markup of Web Services via ontologies in order to facilitate computer-interpretability, sharing, and mapping. The automation tasks that a semantic markup of Web Services must enable are: Automatic Web Service Discovery, Automatic Web Service Execution, and Automatic Web service composition and interoperation.

Automatic Web Service discovery involves automatically locating Web Services that provide a particular service and that adhere to a requested set of properties. With semantic markup of services, we can specify the information necessary for Web service discovery as computer-interpretable semantic markup at the service Web sites and for the service registry. Ontology enhanced search engine can help automatically locating appropriate services given a set of requests (properties).[SAMZ01]

Automatic Web Service composition and interoperation involves the automatic selection, composition, and interoperation of appropriate Web Services in order to perform some tasks, given a high-level description of the tasks’ objectives. With the semantic markup of Web Services, the information necessary to select, compose, and respond to services is enclosed at the service Web sites.[SAMZ01]

Automatic Web service execution involves a computer program or agent automatically executing an identified web service. Semantic markup of Web Services provides a declarative, computer-interpretable API for executing services. The markup precisely defines the necessary inputs, outputs, and possible side effects of the execution of the service. [SAMZ01]

Markup exploits ontologies in order to create a distributed knowledge base of a specific domain. However, those pieces of knowledge are not static, but evolve over time, which means that the ontologies will also change. The evolution of ontologies causes operability problems, which will seriously hamper the effective reuse and sharing. The problem gets even worse when there are a lot of dependencies between data sources, applications and the ontologies. Thus, it is necessary to have a versioning process that helps to keep track of the different evolutions. As discussed in [KF01], backward compatibility should be the main characteristics of an ontology.

1.3.1 Ontology definition

This section gives two different revisions of a seaplane's ontology. Those ontologies were inspired from the examples given in [KF01].

Revision n1 : Seaplane-v1 This ontology defines a seaplane as a plane that must land on the sea, where the sea is modelled as a runway.

```
class-def Runway
class-def Sea
    subclass-of Runway
class-def Plane
class-def Seaplane
    subclass-of Plane
    slot-constraint type-of-runway
        has-value Sea
```

Revision n2 : Seaplane-v2 The definition of a seaplane is updated, and allowing any plane that land either on a classic tarmac runway or on the sea to be called a seaplane.

```
class-def Runway
class-def Sea
    subclass-of Runway
class-def Tarmac
    subclass-of Runway
class-def Plane
class-def Seaplane
    subclass-of Plane
    slot-constraint type-of-runway
        has-value (Sea or Tarmac)
```

Remarks:

1. Version 2 of the ontology is not a backward compatible revision (it is incompatible).

2. Seaplane-v1 is included in Seaplane-v2, meaning that the interpretation of the data will not be incorrect, but it will not be complete. (i.e.: We cannot interpret with Seaplane-v2 that every ”Seaplane” in the data sources (that was constructed with Seaplane-v1) can actually land on the sea.

1.3.2 Usage scenario ontology versioning

This is an attempt to describe the possible event that may happen during the automatic composition of Semantic Web Services described by ontologies. A user will connect to a service that organises travel arrangement. This service must then 1) find, 2) combine, and 3) execute services in order to complete the user’s task.

During the discovery, composition, and execution process, the service will have to take into account that the ontologies may have changed. We will look at extreme cases and practice in order to show the necessity of a versioning system. In the first case, the ontologies are silently changed; the previous version is replaced by the new version without any formal notification. In the second case, the ontologies are changed following a well specified versioning framework (like the one in [KF01], section 4.4).

The purpose of the next paragraphs is to explore the inconsistency and/or incompleteness of Web Service data when the ontologies are changed. We will focus on the Service Discovery process.

Actors & Goals of the use case:

1. A Client that want a travel arrangement for the KnowledgeWeb workshop
2. A Travel Arrangement Web Service (TAWS) to which the client connect to.
3. A Directory Service (DS) containing the list of all callable Web Services.
4. A set of Web Services invocated by the TAWS found in the DS.

The goal is for the TAWS to organise the client’s travel arrangement to a friend’s house in Canada (the house is actually on the Ontario Lake border). We will focus on getting the plane ticket. This scenario can be made more complex by studying automatic service composition and execution.

Use Case: No Ontology Versioning

Issues: Explore the incompleteness of Web Services data when the ontologies are changed with new one (erasing the old one) without any kind of formal notification.

Assumptions: The Service that books plane tickets has changed its ontology from Seaplane-v1 to Seaplane-v2 (replacing the old one). The TAWS do not know that the ontology has been changed.

Scenario & Steps:

1. A client starts by connecting to the TAWS and then enters his desired travel arrangement.
2. The TAWS decompose user's task in sub-task so that each one can be executed by one specific service.
3. The TWAS will call the service in charge of the plane ticket booking.
4. Because in this case, a Seaplane-v1 is a subset of Seaplane-v2, we cannot interpret anymore that every ”Seaplane” in the data sources can actually land on the sea.
5. Thus, we might miss an opportunity to take the plane and the TWAS will book a car instead (meaning we will have to drive for hours).

Use Case: With Ontology Versioning

Issues: Show that with correct ontology versioning we are able to correctly interpret the Web Service Data

Assumptions: The Service that book plane ticket has changed its ontology from Seaplane-v1 to Seaplane-v2. This time, the Web Service uses an ontology versioning methodology like the one described in [KF01]. Thus TAWS will now know that the ontology has been changed.

Scenario & Steps:

1. A client starts by connecting to the TAWS and then enters his desired travel arrangement.
2. The TAWS decompose user's task in sub-task so that each one can be executed by one specific service.
3. The TWAS will call the service in charge of the booking the plane ticket.
4. Given the versioning mechanism, TAWS discovers that in version 2 of the ontology, the relationship type_of_runway has been completed. Knowing that, and knowing that our data source was constructed with Seaplane-v1, we can still interpret that every ”Seaplane” in the data sources can actually land on the sea.
5. Thus, the TWAS will book the plane ticket (meaning that we won't have to drive for hours).

1.4 Current versioning approaches

1.4.1 Source Code Versioning

The very popular Concurrent Version System[5] (CVS) [Ber90] initially was a collection of scripts to simplify the handling of the Revision Control System (RCS) [Tic85]. RCS operates in a file-centric way by using a “lock-modify-unlock”-style. However, CVS works on the syntactical level, not on the conceptual. *I.e.*, it is not capable of versioning objects and in particular not capable of versioning ontological entities and their complex structure. The underlying `diff` operation is capable of showing the syntactical differences between two files (based on the differences of text lines). Subversion [sub04] recently gained a lot of attention as designated CVS successor.

As theses code versioning system already solved the problem of syntactic versioning, we will focus on structural and semantic versioning in our approach.

The data and metadata management of the code versioning systems will be a starting point for design the corresponding ontology versioning system.

1.4.2 Database Schema Versioning and Evolution

As ontologies are also often schemas for instances data, similar problems occur as already known from database schema versioning (cf. e.g. [Hue97, BKKK87] etc.). [Rod96] gives an overview of important problems in database schema versioning. A semantic approach for schema evolution and versioning can be found in [FGM00].

A critical discussion of the relationships between ontology evolution and these approaches can be found in [Sto04].

1.4.3 Ontology Versioning and Evolution

Following terminology from the database community (cf. [Rod96]) we mainly distinguish between „ontology versioning” and „ontology evolution”. The difference between schema evolution and ontology evolution is shown in [NK03]. Ontology versioning is accommodated when an ontology management system allows for handling of ontology changes by creating and managing different versions of it. Ontology evolution is accommodated when an ontology management system facilitates the modification of an ontology by preserving its consistency.

A first survey on causes and consequences of changes in an ontology have been described in [KF01], followed by an implementation for ontology versioning (cf. [Kle04]) that is based on the comparison of two ontology versions in order to detect changes. Even

---

[5] Available freely for download at `http://www.cvshome.org/`

though there are many ways to transfer an ontology into a new version, this system generates only one solution based on the set of heuristics. It mainly addresses OWL and OKBC as representation languages and does not deal with RDF/S specific problems as we do.

An more detailed survey on ontology versioning has already been given in the DIP project [MRA04].

Chapter 2

Economic Aspects of Versioning and Evolution

As the above chapters have shown, evolution and versioning of ontologies are considered important issues in current computer science research since ontologies inevitably change over time. In this area a versioning system is to be considered software that should allow the maintenance of identity relations between evolved ontologies. Each new version of an ontology should be connected to the other versions in a possibly complex structure of ancestors and descendants.

Unfortunately, even if in the last years important work has been done in versioning problems, both the supporting technology and the processes associated with ontology life cycle management have been focusing on the dynamics of change only from an engineering and formal perspective.

Conversely, the fact that people need to share meanings in order to perform social and economic processes (along with meaning) fundamentally embedded into social activity and economic dynamics has been generally ignored by versioning studies. Anyway, along this pragmatic perspective, others research fields have observed how economic interests, practices, and social identities concretely determine, contribute, and facilitate semantic resistance or change. Intuitively, when meanings change, such changes are affected by and affect these concrete dimensions of economic and social life. A particular configuration of a system of meanings should be considered as a particular configuration of social and economic relationships.

The main goal of this section is properly to analyse the underlining causes of change of ontologies from a socio-economic perspective. In fact, we think that the understanding of such factors will result in a better capability of developing versioning software tools that will be able to consider how changes in the ontologies are interwoven with those social accounts and economic interests that are attached to ontologies. In general, we say that to understand how mapping changes in order to make consistent to systems of knowledge versioning should adopt a pragmatic point of view based on social practice.

2.0.4 Ontologies: description of concepts, relationships and agents

A brief review of the concept of ontology is necessary in order to focalise better on its main parts. The concept of ontology was taken from the philosophy field and currently is utilised in technological domains. More precisely, in the latter domain, ontologies are considered as applied ontologies since they are used for practical reasons i.e. they serve as means to communicate between different entities. In particular, since an ontology has as requisite a well defined formal semantic it can be used from computer machines that are able to interpret data and exchange information according to the ontology. Here a series of relevant definitions of an ontology are reviewed. One of the first main accepted definitions of ontology was proposed by Gruber: An ontology is a formal, explicit specification of a shared conceptualisation that holds in a particular context. [Gru93]. Another interesting definition is taken from Russel and Norvig that focus on the important role of agents: an ontology is a formal description of the concepts and relations which can exist in a community of agents. [RN95]. Furthermore, Swartout focuses on the structure of an ontology: an ontology is a hierarchically structured set of terms to describe a domain that can be used as a skeletal foundation for a knowledge base [SPKR96]. In conclusion, Fensel defines an ontology as a common, shared and formal description of important concepts in a specific domain. [Fen00], [Fen01].

In more general terms from these definitions we can summarise the characteristics of an ontology: it is a representation of a set of concept and their relations assumed to exist in a certain domain from the point of view of a community. The representation is build up through a particular language that is used to describe the environment and to allow information exchange between agents. Some example of artifacts with ontological characteristics can help to clearly understand the topic: glossaries, thesauruses, dictionaries and encyclopedias can be used as a reference points to commonly understand words and to communicate. A successful communication in a multi-agent system requires not only that communicating agents share a common language, but also that they are committed to the same intended model for the semantics of this language. As we have seen from the above considerations, an ontology properly satisfy these two dimensions [SC03]. In synthesis, from a technological point of view, the kind of language (that is used to formally describe concepts, instances and relations) and the view of the world evoked through language embody the very essence of an ontology: its meaning[1].

Starting from this consideration, in the next section we will review some important socio-economic area that study the change in meaning.

---

[1] An ontology necessarily includes a specification of the terms used, (”terminology”) and agreements to determine the meaning of these terms, along with the relationships between them [Sta03].

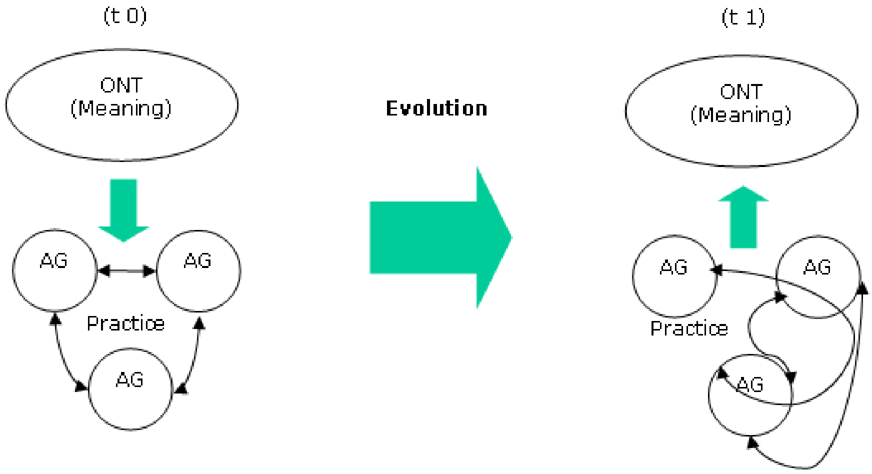

Figure 2.1: Meaning influences practices while practices influence meaning.

2.0.5 Socio-economic approaches and their relevance for meaning versioning and evolution

The socio-economic issues used here focus on the particular field of economics called Economics of Language and several sociological approaches. Even if there is no direct relation between ontology studies in computer sciences, economics of language and sociological argumentations, we think that there is a strong correlation among these fields. In fact, although with different goals, the objects of study of these areas are the same: how knowledge representation (ontology) is developed and modified during time, and how language as the basic element of knowledge representation is produced and modified. In particular, the socio-economic perspective underlines that meaning change and evolves during time due to the practices performed by users (see figure 1). In fact, an ontology provides a shared vocabulary (language) and the representation of the type of concepts and relationships that exist within a particular domain (knowledge model).

More in particular, the socio-economic perspective emphasises some more general thought about the peoples needs to share meanings in order to perform social and economic processes. Some consideration can be taken into account; people attach to meanings their concrete interests, practices, and identities. In this sense, when meanings change, such changes are affected by and affect these concrete dimensions of economic and social life. As a consequence, the way in which meanings evolve cannot be understood without considering how such changes are caused and impact on social and economic accounts. Then, a particular configuration of a system of meanings should be considered as a particular configuration of social and economic relationships.

In addition, some non-exhaustive but concrete question on ontologies versioning can be briefly formulated: how does the software medium used for the ontologies influence the development and modification of languages and meanings used among users? How do people model an ontology able to encompass the change of meaning in the daily human practices? What does it mean that an ontology is considered valid from a socio-economic perspective? How the needs of the ontology users are considered in the development of an ontology from a socio-economic perspective? That is why we think that the evolution of an ontology cannot fully understood as a logic, abstract and formal problem. More in particular, this section wants to investigate how social evolution of meaning can be reflected as changes in an ontology.

The perspective used here tries to give a response to these questions and to understand, trough socio-economic considerations, how individuals and groups generate and modify their language and knowledge in communicating, and to communicate. In the following, different socio-economic approaches to knowledge representation and language are presented.

SSK (Sociology of Scientific Knowledge)

Sociology of scientific knowledge is an empiric sociology aimed at analysing the production of knowledge within a particular social context of reference. In particular, this school of thought focuses on the production of knowledge starting from the different practices within the scientific communities [Blo76], [Col85]. The principal element that has been highlight by the different approaches within SSK is that production and sharing of knowledge is based on social consensus. Consensus is dependent from the interests (that is to be understood not only as economic but as a general social dimension) that the different communities have. Consensus also determines the correctness of a particular practice. As a result, communities interests determine the practice to be used in a domain. In general terms, a practice can be considered as the set of actions used to do something that agents perform in a determined community [Pic92]. In conclusion, SSKs scholars state that one of the main independent variables that influence the production of knowledge is the concept of interests.

In this sense, from the SSK perspective, using a particular logic-based, language for representing problems can be considered a practice within a particular community. Indeed, the use of a particular language depends on contextual factors. Furthermore, the particular artifacts (instruments) used for laboratory experiments and practices not only depend on the particular culture of the community, but influence the results of the experiments.

Ethnomethodology

A different branch of social studies is Ethnomethodology. Garfinkel [Gar67] argues that in praxis there is the possibility to explain the way trough which social members construct and shared social reality. In particular, the aim of Garfinkel is to explore intentionality related to theory of praxis. In other words, he tries to figure out the concrete content of concepts such as: common sense, inter-subjectivity, meaning, etc. In this sense, Ethnomethodology can be defined as a particular discipline interested in discern how common sense is used to determine reality and justify behaviours. It studies resources, practices, and procedures of common sense through which members of a specific culture produce and recognise objects, events, and courses of action in ways that are mutually comprehensible [Her84], [Her92]. Therefore, analysis copes with daily life creation and use of shared meaning (labelling) about world events and objects as unconscious and never-ending use of methods (accounts) to establish reasonable bases to behave in the world and claims that this methods have to be explain by social researchers [Lin99].

While, initially, Ethnomethodologist addressed their attention towards the natural attitude and practice of common sense and argued that actions are realised by actors in routine ways and are responsible of the production and justification of social reality, to date, the aim of Ethnomethodology is to study scientific activity (Study of Work) [Lin93]. In this direction it tries to show, comparably to the SSK, how the production of scientific knowledge is merely social and is based on repeated mechanisms of interaction in which cognitive activity is not only the effect but also the source of social reality. Said differently, also Ethnomethodology assumes that social practices, for example in scientific research as well as a martial activity (kung-fu), are essential in the processes of knowledge production and legitimisation [Gar88], [Gar96].

Situated Learning

The Situated Learning perspective is a socio-practical approach which is linked to both the constructivist tradition and the more radical approach of post-modern thinking. It considers knowledge production as a process through which negotiation of meaning takes place continuously through interaction among social actors. Such process, when stably repeated, can produce a set of regular actions correlated to a collection of meanings that identify a social structure, namely, a community of practice. This context represents knowledge because all the involved elements, both human and artefacts, have symbolic function because they embed some interpretation on what needs to be done in respect to the others. In other words, elements represent some established meanings in virtue of some mutual and stable relations and these meanings imply a sort of rules for subsequent interactions. As a consequence, the production of both collective knowledge and a shared identity are considered as two faces of the same social process base on tasks accomplishment [BS99], [BD91]. In this sense, there is a collective learning that is a process of participation in contexts and implies people develop a sense of belonging [LW91]. Knowing how to interact in order to produce meaningful contents implies knowing how to behave in relation to other members expectations.

Situated Learning, arguing that knowledge is a social and contextual construct, advises that there are many knowledges that depend on specific actions. As a consequence, a pluralistic perspective emerges, given that a knowledge cannot be said to be better than another as far as meaning depends on contextual conditions. This suggests that centralisation or standardisation are not correct solutions in order to express the wealth of meanings and perspectives that populates an organisation when red as a knowledge system. Moreover, such redundancy should be accepted and exploited as an opportunity to generate value. Moreover, the link between learning and identity formation suggests the opportunity to emphasise how every intervention about knowledge -far from being neutral-influences, impacts, and is driven by the interests of different communities.

Economics of Language

As said, trying to comprehend how individuals and groups generate and modify their knowledges, we need to take in to account various perspectives in language studying. And, besides philosophical, sociological, or cognitive theories, we advise that also Economics seems to explain human behaviours related to language. Indeed, according to the discipline named Economics of Language, theoretical frame of economics can be useful to recognise new factors that can be put in relation to knowledge processes. In particular [Gri99] gives a definition of this branch of study: The economics of language [] refers to the paradigm of mainstream theoretical economics and uses the concepts and tools of economics in the study of relationships featuring linguistic [] variables; it focuses principally, but not exclusively, on those relationships in which economic variables also play a part. Along these lines, in this section our objective is to understand how cost-benefits logic affects language considerations.

Reviewing Economic of Language researches [Har04], [Gri02], [Bre97], [GV97], [Chi91], [GV83], [Hoc97], we can recognise different traditions of study. As [Gri02] points out there are two main approaches, the first is referred to the researches conducted primarily in the North-America area and considers language as an explanatory factor of economics (e.g. language as a determinant of labour income); the second is related to more recent European studies which stress the role of economic aspects on affecting language variables (e.g. income as a determinant of language maintenance).

Our purpose is to identify from this literature some detected language factors that can suggest an economic interpretation of knowledge changes.

Language as Human Capital

One of the main approaches is based on Human Capital Paradigm [Bre64]. Human capital is defined as the stock of skills, physical and mental health, knowledge, and so on, which are a contributing source to a persons earning power. Human capital growth is the result of investments of resources in that form of capital and is possible if current consumption is less than current actual and potential income. Namely, human capital is the accumulated stock of produced means of production. From this point of view, learning one language is an investment of resources (tuition fees, dictionaries, grammars, cassettes, books and, of central importance, time not allocated to other valuable activities including leisure) and instruction is an example of this investment.

Such Economics studies on language are remarkable since they have detected important factors on the costs and yields of modifying language. Analysing costs, the ability of the learner is the first factor that has an impact on investment; secondly theres the influence of the family and especially of teachers and schools in terms of competence to perform language and ability to motivate learner; finally, another element is mentioned that may be called linguistic intolerance: its proved that a status of inferiority acts as a deterrent or as a barrier to investments in quality improvements. Besides these cost factors, there is a gross yield on improve languages investments. It includes income, social status, and a cultural component (which allows the individual, after a certain volume of investment, to enjoy literature, the arts, and a rich social interaction with other ”cultivated” members of her society, and of other societies as well). Its to note that, as [Breton, 1999] argued, when the cultural yield (say identification function) is too low, the extinction of languages is inevitable.

Especially, the human capital approach has been applied to bilingualism studies. Researches state that, in a market equilibrium, prices will be set in labour markets and in the language training market such that individuals learning a second language will be compensated by a favourable wage differential, such that the present value of the additional earnings just compensates for the cost of learning that language (included in ”cost” are the foregone earnings due to time spent learning the second language). But this dynamic version of the paradigm, it is held [Gri90] and [Gri92] that the marginal resource cost of investing in a second language is reduced, in a second period, by the positive effect of the human capital (the second language) that has been accumulated in the first period. In other words, the positive effect on subsequent capital formation is like an investment return from the production of some amount of language capital at time t that reduces the investment costs at t + 1.

Language as public good

Several studies underline the economic dynamics of the language (spread, survival and decline) and are linked to economic literature of network externalities [Dal99], [CK93]. Network externalities are assumed to exist with language, so that, say, an individual that chooses to learn a particular language confers a benefit upon all those currently using the same language. In particular, an additional (n+1)th individual speaking the same language would yield direct benefits to all others by adding 2n potential new interactions. An additional member of a linguistic group will also provide an indirect benefit to other members by increasing the demand for language-sensitive goods and services (schools, libraries, bookstores, broadcasting, theatre productions, etc.) and thereby, if economies of scale are present, improving the supply and the variety of such services within the community. Put differently, when the individual’s choice is between two competing and mutually exclusive communication language, a switch from one to another confers a cost on all those using the discarded language, as the size of their compatible group has been diminished. As a consequence, we can understand that the number of speakers appears one of the most appropriate factor in determining the communication value of a language. In fact, the reward obtained by each individual who joins a language community is due to the new potential for communication thus acquired.

A consideration stems from this line of argument. If an individual, who speaks language A and moves to another group, confers an external benefit on all speakers of A within the host group and a loss on all speakers of A in the source group, then one can see the presence of a positive feedback effect between A language network externality and migration in response to economic opportunities which are specific to one group. Hence, because each member of a language community obtains a reward whenever their community expands, migration plays an important role in the long-term evolution of language use (and also cultural and ideological elements) through its impact on language group size. What is more, there is evidence [HR] that migration tends to induce trade flows, because persons bring both established tastes with them, and tastes are to some extent specific to the group of production and because immigrants have the knowledge and contacts necessary to establish and maintain bilateral trade linkages. In other words, network, enlarging socio-economic development, does not benefit only its members, that is its native speakers, but the economic system as a whole. For the same reason changes in ontology can be seen as network externalities effects influenced by decision to join a group and shaped by migration movement.

Its to note also that network externalities can be explained taking in to account that language can be seen as a public good [Dal99]. This means social increasing returns associated with single language use exist because language quantity is not reduced when someone consumes some of it and at the same time is accessible to all. This language characteristic has a negative implication: neither costs nor benefits related to network externalities are incorporated in individuals' decision criteria. So, when individuals make a choice to use, in a relatively permanent sense, one language or another, a set of costs and benefits arise and thus a discrepancy arises between privately determined language choices and those that would maximise social efficiency [CK93]. This is because the balancing of private costs against private benefits does not take into account consequences of that decision for the size of the network and thus the benefits to others on the network. Then, when network externalities exist, private investment decisions on language adoption will not result in an optimal allocation of resources. This means that a person will not learn the second language as well as she could. In addition, other persons will do like her so that the number of persons learning the second language will be too small. The size of the decentralised under-investment in second languages is a function of the degree of imperfection in the mechanism that ”coordinates” investment decisions. Due to the fact that language use is a public good and cause network externalities that increase the language cost for individuals, it can be read like a natural monopoly. Accordingly, there is the possibility that a language becomes the historical equilibrium lingua franca. That is, the development of a common language accomplishes an improvement in collective welfare by internalising those effects that tend to be left out of an individual's decision process.

Its to recognise that the emergence of a lingua franca is like a new technological innovation which reduces the costs of communicating among individuals and consequently confers benefits to all (though not necessarily in equal amounts) who are communicating [Lam02]. Only those, who have to invest resources in the learning of the lingua franca, pay to reap the benefits, individuals whose mother tongue is a lingua franca receive a benefit from that fact alone. Its what [BM79] called the ”seigniorage of language”. From ontology point of view, changes can be directed toward more efficient communication in terms of collective cost.

Its to note that no a priori reasons are found to explain why a language is the current lingua franca. Students say that small ”historical events” select one language but that one, then, tends to ”take off”, and ”lock-in” communication practice. An attempt to give an analytical explanation by [Dal99] proposes four possible elements associated with language diffusion: Large set-up and fixed costs such as those associated with the investment necessary to the maintenance and development of a modern language give the advantage of falling unit costs as the size of the speech community expands. Learning effects improve the efficiency of investments in language capital, of the provision of language-related services, and more generally of the activities meant to promote a language diffusion and export. Coordination effects refer to the advantages associated with undertaking actions and making choices synergetic to those of the other individuals in the same environment. Adaptive expectations, finally, account for the fact that the increased prevalence of a language enhances beliefs of further prevalence. In this sense Dalmazzone asserts: the linguistic composition of a given society may evolve more as a consequence of some crucial past event than of the current objective situation.

Language as transaction cost

In some investigations [SB93], [OCK98], [Hol87], [Gri94] it is asserted that a common language is a medium that enhances organisation, coordination and management activities, through the sharing of knowledge and the creation and the diffusion of innovations. From this perspective, language differences are considered like any other barrier to trade and require the expenditure of resources if they are to be overcome. Therefore, language is recognised to be a specific transaction cost [Wil75]. This interpretation allows some implications. Firstly, because language differences are barriers that can be overcome only by using scarce resources, one would expect people of the same language to exchange more (as well as to work more) with each other than with people who are of a different language, since that will economise on scarce resources and reduced number of participants. In this regard, some studies show that investments within linguistic families, for example Italian and Spanish, benefit from economies of scale. Concordantly to this, [Hel99] demonstrates that the condition that makes trading possible is the actual possibility to communicate between language groups and it is not groups geographic closeness. [Har95] argued also that the higher is the cost of overcoming other kind of barriers to trade (e.g. the geopolitical location of a country, its economic relations with other countries, the level of industrialisation and institutions), the lower will be the yield on resources invested in language barriers. In particular, there are essentially language brokers (interpreters and translators) and they can mediate ”communication” among a large number of monolingual individuals of different languages, permitting more communication to take place per unit of resources than if the same volume of resources had been used to increase the number of bilingual people. Brokers are an intermediate solution with respect to monolingual strategy and multi-language learning strategy in case of numerous cultural communities and represent a specific investment strategy that maximise earning. However, because their presence serves to lower the yield on second languages and, as such, leads to a smaller number of bilingual persons in any given context, they correspond to a quite high-risk rate strategy in respect to content diversification and innovation opportunities.

Language As Optimal Structure

There are economic explanations of the work language based on the idea that language can be interpreted as a content dimension of consumption and production. Researchers deemed this matter in several ways. Some studies stressed the role of production in the maintenance and development of a language. For example the literary productions in a certain language contribute to the utility value of this language. Accordingly, because the activities can be performed in one or another language, such activities can be seen as the ultimate arguments of utility function of a language. As a consequence, the cost of carrying out an activity in a given language should be considered. In this direction [Hoc97] analysed changes in production costs functions depending on the language characteristics of the outputs. Similar studies are interested in observing negotiations of languages in use among organisational communities. In particular they argue that centralisation of language is sub-optimal solutions for an organisation because many languages necessarily exist that are related to inter-organisational and inter-organisational groups. Different languages have to coexist because differential efficiency is related to their different functions, competencies and identities. Along this line, [Col91] focused his work on the choice of language in conversations among people having different linguistic attributes. There are other researches that, assuming that the language in use in a firm depends on ownerships preferences, empathise the role of constrains in the emergence of a work language. There are constrains posed on marketplace (e.g. purpose of distribution in local markets), technology tools (e.g. diffusion of innovations in information and communications technology), and workforce practice (e.g. availability of linguistically trained labour force). Due to this, the language emerges as optimal solution coming form the interaction of these previous factors [Bre98], [BG96], [Bou82]. Through a more analytical method [SP91], [Pra01] game theory to define language content from a discourse situation that in general is one of the possible meaning of a sentence varying from situation to situation. By means of the ideas of rational agency, strategic interaction and equilibrium, developed by economists like Von Neumann, Arrow, Nesh, Debreu and Aumann, Prashant; content as a function of agent architecture, sentence meaning and situation of utterance. Prashant assumes that language is context-dependent action and agent chooses this action (utterance and interpretation) from a set of actions based on the high expected utility. The author asserts also that agent take into account other agents actions generating a complex interaction considering knowledge and beliefs and especially shared knowledge of the situation as far as agents dont have incentive to change. In this sense, students as [Blu99], [Rub00] say that, because a language is structured by subjects, which have their perception of the world and try to construct language in order to maximise their utility, it tends to evolve in an optimal structure. At the same time this studies advise that a language influences the way humans perceive the world, first because its structure (not the content) can drive the agents choice in a specific direction, second because it can influence the decision through existing ways (content) to formulate alternatives [Rub98], [Lip02]. What is asserted is that a language can survive if it is efficient in transmitting a great amount of information. The intent then is to define one general optimal structure that gives reason to a languages preservation. Following [Lip02], this optimal structure is based on four criteria: the ability to identify objects; the ability to indicate relations between objects; the acquisition facility; the ability to communicate new and unexpected events.

2.0.6 Relationship between socio-economic approaches and ontology versioning and evolution

Starting from the research areas above proposed, it is possible to suggest some general idea of how they can influence ontology versioning and meaning evolution. These ideas will be underlined for each area, and than a general conclusion will be proposed.

Sociology of scientific knowledge (SSK): Turning these considerations in the field of ontology development, it is possible to state that the utilisation of a particular language and perspective in a community underlines some implicit ontology that derives from some core interest of the community. In particular:

1. The communities interests influence the practices of every day work.
2. In turn, the practices influence the way in which knowledge is produced and represented.
3. Ontology construction and evolution depends form the interests that influence representation of knowledge.
4. If the communities interests change, practices and ontologies will change.

Ethnomethodology: The ontology is an expression of the way in which work and knowledge are concretely organised around artifacts used in the work.

1. Stability of ontologies is the result of the productive routines i.e. the practices in the every day work.

2. The artifacts have a key role in understanding such practices (and the knowledge sharing).

Situated Learning: This approach suggests that different ontologies are systems of meaning correlated to different practices and communities of users so that versioning should take into account identity factors in communities of practice to explain changes in ontologies. Furthermore, Situated Learning recommends to put the attention on practice negotiation in inter-community relations that allow changes of meanings. In particular:

1. A community creates and develops a particular representation of the domain (ontology) according to its identity.
2. Modifications of the ontology depends on the:
    (a) Relevant identities that manage the resources.
    (b) The introduction of new individuals in the community (that do not share, at least at an initial time, the identity).
    (c) The interaction with different communities (identities).

Economics of Language: The Economic of Language perspective underlines that the use and evolution of ontologies can be affected by different economic principles. For instance, this perspective suggests that the investment in an ontology represents a sunk cost in human capital due to the fact it is subjected to path dependencies in subsequent investments evaluations.

Additionally, the more an ontology is popular the more there is a reduced risk for its future use. This perspective sets up positive feedback effects on their adoption. Consequently, ontologies that are expected to be popular will be used, even if they are not the fittest. Further, one can view ontology evolution as a specific investment to facilitate inter-group communication needs, that is, changes in representation are lead by specific inter-group exchange. Moreover, mediators among different ontologies correspond to a specific strategy that reduces cognitive costs but increase meaning creation.

An ontology is also influenced by pragmatic factors and can be interpreted as a group negotiation based on difference practices and competencies. Anyway, because the ontology can frame its own changes by means of syntactic and semantic previous productions its endurance depends on its adherence with pragmatic needs. Summarising, the use and evolution of an ontology could be affected by:

1. Externality effect.
2. Adoption costs.
3. Ability to identify relevant objects and relations of the domain.
4. Past ontologies developments.

Each approach underlines the close relation between the domain to be represented and who is the person involved in the construction and evolution of meaning structures. Technology and methods/tools used to acquire information are crucial factors in determining the structure of an ontology. In fact, they influence the practices and the conceptualisation of knowledge.

Practices heavily influence the meaning production and evolution. Practices analysis could allow the understanding of ontologies production and change.

In conclusion, from a research point of view, these approaches underlines that ontology versioning methodologies should consider the importance of everyday practices since practices are one of the key factors that influence meaning evolution. Further, methodologies should consider also the concepts of interest and/or identity.

Chapter 3

Versioning Methodology

3.1 A layered approach for versioning various ontology languages

On one hand our goal is to support as many ontology languages as possible with our versioning system. On the other hand we want to provide this support as specific as possible. That means SemVersion should provide general versioning features independent of the ontology language used. At the same time, it should be easy to integrate functions that are specific for a particular ontology language, like calculating semantic diffs.

To achieve this, we chose an RDF-based, layered approach. In this section we describe how our approach follows the Semantic Web architecture, why RDF is particularly suitable as a basis layer, and explain the roles of the RDF versioning layer and the ontology versioning layer in our approach.

3.1.1 Following the layered architecture of Semantic Web languages

For the Semantic Web, a layered architecture has been suggested [PS02]. These layers define different aspects of the Semantic Web like syntax (Unicode, XML, URI), data model (RDF, RDF-Schema) and semantics (ontology vocabulary, logic, proof).

In the Semantic Web architecture RDF plays a key role as a universal data model that is generic enough to encode all the ontology languages needed for representing richer semantics. [BKD$^+$00] and [D$^+$00] demonstrate how new ontology languages can be created on top of RDF and RDF-Schema and how this was done to create the DAML+OIL ontology language which evolved into the Web Ontology Language (OWL) [SD04]. OWL comes in three different flavours (Lite, DL, and Full) and even more OWL flavours are suggested such as DLP [G$^+$03]. But there are also Horn logic based languages that are encoded in RDF like TRIPLE, a rule-based language proposed in [SD02]. Since there are

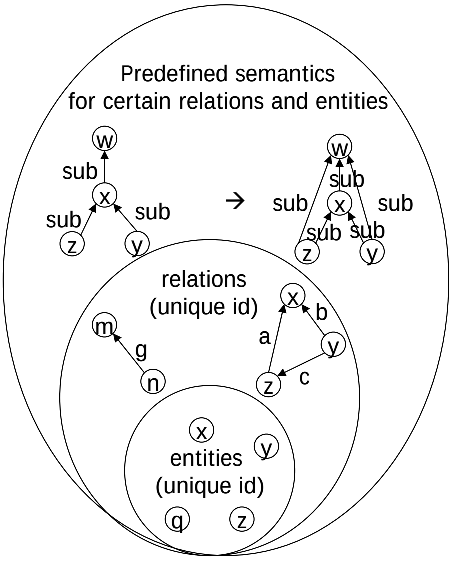

Figure 3.1: Core elements for representing semantics in ontology languages

Semantics	Ontology Languages
Structure	RDF
Syntax	XML etc.

Figure 3.2: Layered view on ontology languages

more and more RDF-based languages, a versioning system that is based on RDF, seems to be a good basis for reaching our goal to support as many languages as possible as specific as possible. The next subsection describes, why RDF is particulary suitable as data model for ontology languages.

3.1.2 RDF as structural core of ontology languages

The most elementary modelling primitive that is needed to model a shared conceptualisation of some domain is a way to denote entities and to unambiguously reference them (Figure 3.1). For this purpose RDF uses URIs, identifiers for resources, that are supposed to be globally unique. Every ontology language needs to provide means to denote entities. For global systems the identifier should be globally unique. Having entities, that can be referenced, the next step is to describe relations between them. As relations are semantic core elements, they should also be unambiguously addressable. Properties in RDF can be seen as binary relations. This is the very basic type of relations between two entities. More complex types of relations can be modelled by defining a special vocabulary for this purpose on top of RDF, like it has been done in OWL. Figure 3.1 illustrates that denoting and stating relations between entities are the second core element in semantic models.

The two core elements for semantic modelling, mechanisms to identify entities and to identify and state relationships between them, are provided by RDF. Ontology languages that build upon RDF use these mechanisms and define the semantics of certain relationships, entities, and combinations of relationships and entities. So RDF provides the structure in which the semantic primitives of the ontology languages are embedded. That means we can distinguish three layers here: syntactic layer (e.g. XML), structural layer (RDF), semantic layer (ontology languages) (see Fig. 3.2).

The various ontology languages differ in their vocabulary, their logical foundations, and epistemological elements, but they have in common that they describe structures of entities and their relations. Therefore RDF is the largest common denominator of all ontology languages. RDF is not only a way to encode the ontology languages or just an arbitrary data model, but it is a *structured* data model that matches exactly the structure of ontology languages.

3.1.3 RDF-layered ontology versioning methodology

Derived from the considerations above we propose a layered RDF-based ontology versioning methodology.

As the base layer, we suggest an **RDF versioning layer** that manages versions of RDF models and their metadata. The metadata itself consist again of one or more RDF models. This leads to a very generic versioning system, where client system can store their version-related metadata. By systematically storing and allowing access to versions of RDF models and their metadata the RDF versioning system provides the complete data management that every versioning system requires. The system can also calculate structural diffs between two RDF models. This is the set-based difference between the two triple sets of the models. The RDF layer is thus responsible for

- versioning data management,
- structural diff, and
- storage layer access.

The RDF versioning layer is a full, usable RDF versioning system on its own. It can also be used to version RDF-encoded ontology languages as well, by creating an **ontology versioning layer** on top of it. This ontology version layer should provide the following additional language specific versioning functions:

- Semantic diffs that take the semantics of the specific language into account.

- Merging with semantic conflict detection as language specific conflicts can not be detected at the RDF level.

3.2 Versioning on the RDF layer

Commit and branch are pure data management operations. Diff, merge and conflict detection depend on the used data model. As merge and conflict detection depend on diff calculation, the diff operation is really at the heart of a versioning system. Versioning on the RDF data layer consists of the core functions download, diff, commit (R2), branch and merge (R3), which will be described first. We then explain additional management functions necessary for a full versioning system (R1, R4).

The most well-known versioning system in the developer community is probably CVS [Ber90]. We explain the SemVersion terminology by referencing the CVS terms.

download via HTTP GET in SemVersion is equivalent to a CVS `checkout`. It is the same as `update` as SemVersion has no locking mechanism.

commit can either be done by committing RDF models or by committing RDF diffs. If the RDF models in question are too large to be send over the network, committing diffs is required. Is this case one needs a way to address blank nodes unambiguously, to be able to update them correctly as well. We present our *blank node enrichment* approach. If a user commits a new version of an ontology, it is considered an update to an existing version.

diff in SemVersion returns a triple-set-based difference between two models. This is a *structural diff* in contrast to a *semantic diff*, which takes also semantically inferred triples into account.

branch and merge work the same as in CVS.

In an ontology life cycle, one can distinguish the engineering phase and the usage phase. The difference between the two phases is: Once an ontology is used, that specific version should not change without notice. In our model, we don't distinguish between these two phases. Instead every ontology version is accessible unchanged via HTTP for an unlimited period of time. To enable automatic updatable versions we introduce „magic URLs" that refer to the most recent version of a given ontology in a given branch.

In an ontology versioning context retrieval will be the most often accessed operation, as e. g. RDFS or OWL ontologies are referred to by their URLs. This implies that released versions of an ontology have to be downloadable via HTTP.

Version A
```
<anon-73> :hasAuthor "Jim Hendler".
<anon-73> :hasTitle "Robots for Kids".
```

Version B
```
<anon-24> :hasAuthor "James Hendler".
<anon-24> :hasAuthor "Allison Druin".
<anon-24> :hasTitle "Robots for Kids".
```

Conservative diff(A,B)
```
+ <anon-24> :hasAuthor "James Hendler".
+ <anon-24> :hasTitle "Robots for Kids".
+ <anon-24> :hasAuthor "Allison Druin".
- <anon-73> :hasAuthor "Jim Hendler".
- <anon-73> :hasTitle "Robots for Kids".
```

Better diff(A,B)
```
+ <anon-65> :hasAuthor "James Hendler".
+ <anon-65> :hasAuthor "Allison Druin".
- <anon-65> :hasAuthor "Jim Hendler".
```

Figure 3.3: Diff Example

3.2.1 RDF specific function

RDF Diff

The most important function of versioning systems is the computation of a ,,diff”, the difference between two arbitrary versions. It allows users to see and discuss about changes between versions and decide about their acceptance. In the next paragraphs we will look at ways to calculate and represent structural diffs.

The **diff function** $d(A,B) \rightarrow \langle a(A,B), r(A,B) \rangle$ is a non commutative function from two triple sets (A, B) to two triple sets of added ($a(A,B)$) and removed ($r(A,B)$) statements, with $a(A,B) = B - A = B \backslash (A \bigcap B)$ and $r(A,B) = A - B = A \backslash (A \bigcap B)$. Such diffs can be computed by simple set arithmetics for triple sets that contain only URIs and literals, as shown in [KSO02]. Unfortunately, blank nodes make the diff more complicated. If a user commits a new model and later requests a diff, the system cannot tell whether two blank nodes are equal or different. They have by definition no globally unique identifier. Literals are no problem, as they may only occur in the object of a statement. As shown in the example in Fig. 3.3, there are always two possible diffs. The conservative diff assumes that blank nodes between two models are never equal, following the RDF semantics for blank nodes. This can be semantically wrong, if a blank node in one model represents the same resource as a blank node in another model. This is

```
@prefix rdf:, rdfs: – as usual
@prefix ts: – the triple set ontology
@prefix u: – namespace for world-wide unique, generated URIs on this SemVersion server
@prefix bne: – bnode-enrichment ontology
@prefix : – a versioned ontology namespace
@prefix ov: – SemVersion data model
ov:added a ts:TripleSet ; #added
    ts:member u:322169000832000001.
u:322169000832000001
    rdf:subject ¡anon-32¿;
    rdf:predicate :hasAuthor;
    rdf:object ”James Hender”.
    rdf:predicate :hasAuhor;
    rdf:object ”Allison Druin”.
ov:removed a ts:TripleSet ; #removed
    rdfs:label ”removed” ;
    ts:member u:322169000832000002;
u:322169000832000002
    rdf:subject ¡anon-32¿;
    rdf:predicate :hasAuthor;
    rdf:object ”Jim Hender”.
# bnode enrichment
¡anon-32¿ bne:id u:322169000832000003 .
```

Figure 3.4: Example for RDF diff encoding as RDF model using blank node enrichment

especially true for two versions of the same model. Thus a better diff is needed, but only possible, if blank nodes can be identified unambiguously.

We propose to overcome this problem by uniquely identifying blank nodes. This can be done by using a technique we call **blank node enrichment**. Blank node enrichment creates an „enriched model" from a normal model by introducing a new property **bne:id**[1]. The value of this *identifier property* plays the role of an inverse functional property like in OWL. We chose to use a globally unique URI which can be created by a generator as described in Section 3.4. Blank nodes should only have one such property value assigned. This unique URI makes blank nodes globally addressable, while they remain formally blank nodes in the RDF model. All existing RDF semantics are still valid.

Most RDF processing tools will leave this information intact. In the MarcOnt scenario, a dedicated ontology builder is used, so this constraint can be enforced. In SemVersion, the content of every version is blank node enriched before it is stored in the RDF storage layer.

If a user edits an ontology locally and deletes all statements involving a particular blank node, SemVersion can deduce that it has been removed from the model. If a user locally creates new blank nodes, the tool she uses might not use blank node enrichment. SemVersion then finds no reference to existing nodes and correctly identifies the added blank nodes. If a user commits a plain model without identifier properties for some blank nodes, SemVersion falls back to a conservative diff for the non-annotated blank nodes and assumes they are new.

If a user commits a diff as an update to a previous version, this diff should also be *blank node enriched*, in order to work properly.

For **representing RDF Diffs** it might be desirable to have a single RDF model containing the whole diff information. There is no standardised way to express sets of triples within a single RDF model. The most common approach – addressing triple sets by the URL of the document containing the RDF triples – has no defined semantics. Delta[BLC01] proposes „quoted graphs", Named Graphs[CBHS04] proposes a new XML-based encoding called „TriX"[CS04]. Both proposals work outside the RDF model. We chose the following RDF-friendly approach: A triple is made addressable by reification, sets of triples are represented as rdfs:Bags. This leads to a trivial triple set ontology[2]. A full RDF diff contains a triple set of added and a triple set of removed statements. Additionally the blank node enrichment statements have to be added, as shown in picture 3.4.

In the SemVersion API we additionally provide a diff represented as multiple, URI-addressable RDF models.

---

[1] with `@prefix bne: http://SemVersion.ontoware.org/bnode-enrichment#`

[2] Available at http://SemVersion.ontoware.org/2004/12/tripleset

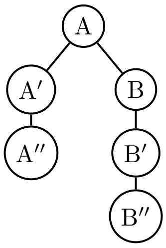

Figure 3.5: A sample version tree

Branch and Merge

Branch and merge operations allow ontology engineers to follow multiple development paths in parallel. A branch operation works like a commit, but the new version is considered to be in a new branch, marked by a different branch label.

For merge we distinguish a merge between two arbitrary versions and the merging of two branches. It is possible to merge arbitrary versions, no only those at the end of a branch. A merge of version A and version B is simply the set union of the triple sets.

Merging two branches is different. First we look at the branch point c, which is defined as the most recent common version of the two branches. Such a version always exists, as branches can only be created by committing a version to an existing version. We also take two versions from the different branches, in most cases the most recent ones, and call them a and b. Consider the example version tree given in Fig. 3.5. Here $c = A, a = A'', b = B''$. In order to merge b back into a we compute the $diff(c, b)$ and apply it to a.

Conflict Detection

RDF models themselves are never in a conflict state. But a diff between two models can indicate a conflict on the ontology layer. SemVersion uses a simple conflict detection heuristic, that detects if a diff adds statements about a resource that was present in c, but has been removed on its way to a. This means, the URI of a resource was used in triples from c, but no triple in a contains this URI.

3.2.2 Management Aspects Of Versioning

SemVersion uses the data model depicted in Fig. 3.6. We first briefly explain the definition and basic operations of each concept:

repository is the root data element for an SemVersion server. Operations: add/remove/list projects, create/delete versioned model;

project is a set of models. Operations: add/remove/list versioned models;

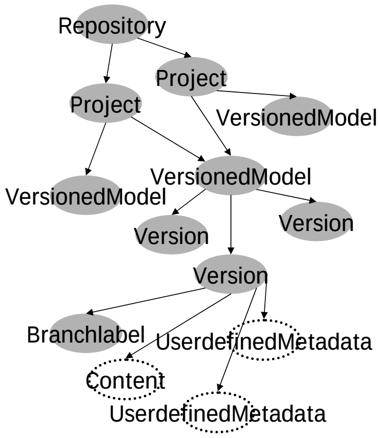

Figure 3.6: Data Model for RDF Versioning

versioned model represents the concept of a thing, that has different versions and branches. Operations: list versions, commit RDF model or RDF diff, commit RDF or RDF diff as branch, list version tree, get current version of branch;

version has predecessors, that are the versions it was created from. It has two predecessors, if it was created as the result of a merge operation. Each version also has a time stamp when it was created and a branch label. Operations: get branch label, add/get user-defined metadata, list history of versions this version was created from, get RDF content (as a set of triples), merge with version, merge with version as branch

branch is an abstract concept. Branches are labelled by URIs.

Typically a new user starts by creating a new project and then adds a RDF model to it. This model is then treated as the first version of a ,,versioned model". The initial RDF model was probably created on the users desktop with third-party ontology engineering tools.

A versioned model consists of different versions that have attributes and relations. Common attributes are *time stamp*, *branch label*, *status of acceptance*. Predecessor relationships indicate the history path. This meta-information about versions can be managed independent of the versioned artefacts themselves. Thus this management layer can be designed very flexible and reusable. As every version can be identified via an URI, one can make arbitrary statements in RDF about them. The concepts of branches, acceptance status and version dependencies can then be represented easily in RDF. SemVersion uses this distinction of stored RDF models and statements about them. Realised as statements about versions is e. g. the concept of ontology engineering projects. Such projects are simple sets of versioned models and give the user a better ability to manage the different ontologies in progress.

Users can store arbitrary RDF encoded metadata objects for each project, versioned model and most important for each version. This data is stored in the RDF storage layer and linked by RDF statements to the versioning artefact it belongs to. Metadata models are also URI-addressable. This metadata strategy enables a good re-use of the SemVersion system, as e. g. the evolution log of an ontology engineering tool could be assigned to a version with this mechanism.

3.3 Using SemVersion for Ontology Versioning

In this section we explain how SemVersion can be used to build an ontology versioning system for a particular RDF-based ontology language. We can reuse the complete version data management infrastructure of SemVersion, that includes managing projects, versioned models, versions and metadata for each of these concepts. Some basic versioning functions can also be used out-of-the box such as retrieve, commit and branch.

The following language specific versioning functions need to be implemented additionally:

- Semantic diffs - the basic structural diff provided by the RDF versioning system is not identical with the semantic diff. To calculate the semantic diff d_l a system has to know the semantics of the specific language l. The semantic closure $s_l(M)$ of a model M is the set of all statements that can be concluded from the statements in M under the semantics of the RDF-based ontology language l. The semantic diff of two models A and B is $d_l(A, B) \rightarrow \langle a_l(A, B), r_l(A, B)\rangle$ with $a_l(A, B) = s_l(B) \backslash (s_l(A) \bigcap s_l(B))$ and $r_l(A, B) = s_l(A) \backslash (s_l(A) \bigcap s_l(B))$. The calculation of a semantic diff can be accomplished by a language specific reasoner or by a language specific set of rules. These rules can be formulated in a language like TRIPLE as demonstrated in [SD02].

- Semantic conflict detection - the ontology language semantics determine what should be considered as a conflict e. g. if the result of a merge is an inconsistent ontology.

Additionally, the specific ontology versioning system might want to have a special diff encoding. Our system can be adopted by providing mapping rules between the RDF diff and the specific language encoded diff.

Further a specific versioning system can use the 'user defined metadata' functionality of SemVersion for storing specific metadata like access rights, degree of agreement, mappings between versions etc.

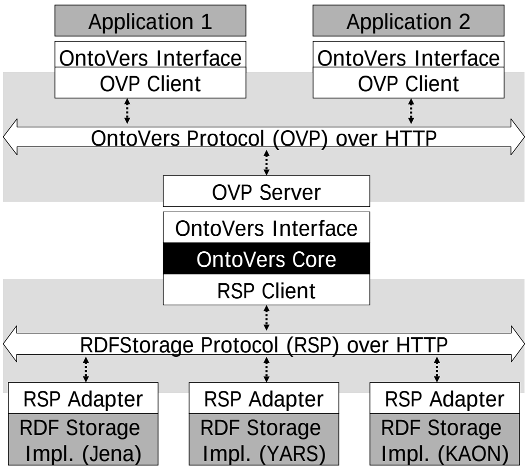

Figure 3.7: System Architecture

3.3.1 Examples

In order to **version RDFS** in a reasonable way, one needs a semantic diff between two RDF Schemas[3]. Besides that, the SemVersion functions should be sufficient.

As a second example, we show briefly how SemVersion can be used to realise the MarcOnt scenario as stated in Section 1.2. SemVersion can manage different branches of versions (M1). Suggestions to the main branch are modelled as different branches, which can evolve separately (M2, M5). Snapshots of the main ontology with suggestions applied are created realised by merging the different branches and showing the user the merged version. As explained on page 30, some conflicts can be detected on the RDF layer (M4). For other conflicts, an ontology language specific semantic diff is required. Mappings between different versions can be stored as metadata of the version for which the backward-mapping is required (M6). As every version can be identified by an URL, it is easy to discuss about them, e. g. link to them in a forum. As URLs are also URIs one can also express arbitrary statements about them in RDF.

3.4 Implementation

3.4.1 Architecture

The overall system architecture is depicted in Figure 3.7. SemVersion is designed as an HTTP-accessible server that handles the versioning business logic. Storage is delegated to an RDF storage layer. In this context, an RDF store does not deal with the semantics for RDF vocabularies like RDFS or OWL.

---

[3] With s_{RDFS} as defined in `http://www.w3.org/TR/rdf-mt/#RDFSRules`

At startup time an SemVersion server loads its root data model from a configured RDF store and caches it in memory. When data is written, the root model is written back to the RDF store. The root model contains information about projects, versioned models, their versions and other metadata. User-defined metadata is stored as separate RDF models in the RDF store. Only time stamps and branch labels are stored directly in the SemVersion root model. This reduces the SemVersion data layer to a clean layer with statements about versioning artefacts. Diffs are calculated on-the-fly in the SemVersion server, but could be cached.

3.4.2 Storage Layer Access

An ontology versioning system should scale in many dimensions. It should allow a large number and size of ontologies. This implies a scalable storage architecture. If the ontologies become large, it is undesirable to download them first and query or manipulate them locally. Thus a remote querying and manipulation facility is needed. There already exist scalable RDF stores with remote query and update functionality. SemVersion adds a small but smart layer of version management on top of such RDF stores. They are connected to SemVersion via the **RDF Storage Protocol** (RSP) over HTTP, which resembles the Java Map interface. The two basic methods are `RDF/XML get(URI u)` and `put( RDF/XML model, URI u)`. The RSP is defined as a simple protocol over HTTP, inspired by the REST [Fie00] architectural style. The method 'get' is e. g. mapped to `HTTP GET http://url-of-rs-store/uri=<u>`. The method 'put' is realised as HTTP POST. The architecture allows to use multiple, distributed RDF stores in parallel which are integrated via tiny RSP adapters.

3.4.3 Handling Commits

The new version will simply be stored – this guarantees that the retrieval will give the user back what she checked in. More sophisticated storage mechanisms could be developed, but the real challenge in ontology versioning is not storage space but the management of the distributed engineering processes within a heterogenous tool environment. The new model is send to the RDF store with a locally generated URI, which is globally unique.

3.4.4 Generating globally unique URIs

The strategy for generating globally unique URIs is as follows: (i) The first part of the URI is the URL the SemVersion server is running at. This reduces the problem of generating globally unique URIs to generating locally unique URIs, assuming that the same SemVersion server URL will not be used for different SemVersion server ever. To soften this constraint, (i)) the current system time for the server, measured in milliseconds is also made a part of the generated URL. Thus the problem is reduced to maintain an accurate server clock and never issue the same URI again in a given period of time (server clock may be off for minutes, but not months). To issue different URIs at all times, (iii) an internal counter is added to the URI string. The URI generator cannot guarantee uniqueness, but the likelihood for the same URI being generated twice is really low.

3.4.5 Application Programming Interface

The general trade-off between the power of a strongly typed, object-oriented API and the flexibility of having direct access to the underlying data exists as well in the RDF and Java world. The open-source project RDFReactor[4], which generates data manipulation classes from an RDF Schema[5], is used to give the user an object-oriented access for many common functions like adding projects, setting the parents of a version or storing the branch label. Alternatively the root RDF model can be used directly.

The OntologyVersionInterface (OVI) has two implementations. One is the real implementation, that calls directly the SemVersion core. The second implementations tunnels all calls over the **Ontology Versioning Protocol** (OVP), which exposes the OVI as an HTTP-based protocol. For convenience SemVersion comes with an OVP Client and OVP Server to bridge between OVP and Java. This architecture allows clients to work remotely over OVP or – via the OVP Client – with the OVI. Local applications can use the OVI directly.

---

[4] `http://RDFReactor.OntoWare.org`

[5] `http://SemVersion.ontoware.org/2004/12/datamodel`

Chapter 4

Neurofuzzy Inferencing and Evolution

In addition to our general versioning approaches we examined new aspects and approaches of ontology versioning that have not been discussed in the research community so far. In order to also consider these aspects in future versioning methodologies and to integrate them into an comprehensive approach, we present our results here.

In this chapter we present an inference engine for propositional rules capable i) to manage imprecise knowledge ii) to evolve the knowledge. The representation and management of uncertainty, imprecision and vague knowledge that exists in real life applications, has received a considerable attention in the Artificial Intelligence(AI) community.

The proposed inference engine operates on a knowledge base that consists of propositional rules. These rules are in the form of ”IF A then B”, where A and B are called logic variables. Propositional rules can be represented using propositional logic. The main concern of Propositional Logics (PLs) is the study of rules by which new logic variables can be produced as functions of some given logic variables. The difference between Description Logics (DL)s and PLs is that PLs are not concerned with the internal structure of the propositions that the logic variable represent. For the interpretation of the propositional rules, there exist two approaches. The first is data-driven and is exemplified by the generalised ’modus ponens’ tautology. In this case available data are supplied to the knowledge based system, which then uses them to evaluate relevant propositional rules and draw all possible conclusions. An alternative method of evaluation is goal driven. It is exemplified by the generalised ’modules tollens’ tautology of logical inference. Here the system searches for data specified in the If-clauses of inference rules that will lead to the objective. These data are found either in the knowledge base, in the Then-clauses of other propositional rules or by querying the user.

The inference engine can refine/adapt the rules, enabling maintenance and evolution of the knowledge base. The adaptivity characteristic of the engine is provided by a neurofuzzy network. Fuzzy systems are numerical model-free estimators. While neural networks encode sampled information in a parallel-distributed framework, fuzzy systems encode structured, empirical (heuristic) or linguistic knowledge in a similar numerical framework [TSK03]. Although they can describe the operation of the system in natural language with the aid of human-like if-then rules, they do not provide the highly desired characteristics of learning and adaptation. The use of neural networks in order to realise the key concepts of a fuzzy logic system enriches the system with the ability of learning and improves the subsymbolic to symbolic mapping [TSK03].

The structure of this chapter is as follows. In the first section we provide the reader with fuzzy set theory preliminaries. In the second section we give a short description of fuzzy propositional logic. In the third section we illustrate the structure of the neurofuzzy network and its adaptation algorithm. Finally, in the last section we demonstrate the performance of the adaptation algorithm and how evolution is performed.

4.0.6 Fuzzy set theory preliminaries

In this section we provide the reader with a brief description of the basics of fuzzy set theory and fuzzy logic.For a more complete and comprehensive presentation the authors suggest [KY95].

Let *X* be a finite crisp set with cardinality *m*, i.e $X = \{x_1, x_2, \ldots, x_m\}$ and let *A* be a fuzzy subset of *X*, with membership function μ_A*(x)*, or simply *A(x)*, $x \in X$.

Height h(A), of *A* is the maximum membership grade of *A*, i.e.

$$h(A) = \sup_{x \in X} A(x)$$

We say that *A* is *normal* if and only (iff) $h(A) = 1$. If $h(A) < 1$, the fuzzy set *A* is said to be *subnormal*.

Support, *Supp(A)*, of *A* is the crisp set which contains all the elements of *X* that have non-zero membership grades in *A*, i.e. $S(A) = \{x \in X \mid A(x) \neq 0\}$.

The *scalar cardinality* $\mid A \mid$, of *A* is defined as

$$\mid A \mid = \sum_{x \in X} A(x)$$

The *certainty subset*, $CS(A)$, of *A* is defined as the crisp set $CS(A) = \{x \in X \mid A(x) = 1\}$ and the *uncertainty subset*, $US(A)$, as the crisp set $US(A) = \{x \in X \mid 0 < A(x) < 1\}$. Obviously, it is $US(A) = S(A) - CS(A)$.

The *fuzzy powerset*, $\mathcal{F}$*(X)*, of the universe of discourse, X, is the set of all the fuzzy subsets of X. Let now $A, B \in \mathcal{F}(X)$. We say that *A* equals *B* iff $A(x) = B(x), \forall x \in X$. We say that *a* is a subset of *B* and denote $A \subseteq B$ iff *A(x)* $\leq$ *B(x)*, $\forall x \in X$. If also $A \subseteq B$ and $A \neq B$, then *A* is a strict subset of *B*.

We will now give the basic theoretic-set operations (complement, intersection and union) defined on fuzzy sets.

The complement $\neg A$ of a fuzzy set *A* is given by $(\neg A)(x) = c(A(x))$ for any $x \in X$. The function *c* satisfies the following conditions in order to normally extend the nature of the standard logic complement:

Boundary conditions: $c(0) = 1$ *and* $c(1) = 0$

Monotonicity: $\forall a, b, \in [0, 1], a \leq b \Rightarrow c(a) \geq c(b)$

Continuity: c *continuous in* $[0, 1]$

Involution: $\forall a \in [0, 1]$ *it is* $c(c(a)) = a$

Several fuzzy complements have been defined in the literature. The standard complement is given by $(\neg A)(x) = 1 - A(x), \forall x \in X$. One example of parametric class of fuzzy complements is the *Sugeno class* defined by $c_\lambda(a) = \frac{1-a}{1+\lambda a}$, where $\lambda \in (-1, \infty)$.

The intersection of two fuzzy sets *A* and *B* is given by $(A \cap B)(x) = t[A(x), B(x)]$ where *t* is a triangular norm (t-norm). A t-norm is a function that satisfies the following conditions:

Boundary condition: $t(a, 1) = a$

Monotonicity: $\forall a, b, d \in [0, 1]$, *with* $b \leq d$ *is* $t(a, b) \leq t(a, d)$

Commutativity: $\forall a, b \in [0, 1]$ *is* $t(a, b) = t(b, a)$

Associativity: $\forall a, b, d \in [0, 1]$ *is* $t(a, t(b, d)) = t(t(a, b), d)$

Moreover, it is called *Archimedean* iff it is continuous and $t(a, a) < a$, $\forall a \in (0, 1)$. Obviously, the *only* continuous t-norm which is not Archimedean is the $min(a, b)$.

Examples of t-norm widely used in the literature are the followings:

Standard intersection: $t(a, b) = min(a, b)$

Algebraic product: $t(a, b) = ab$

Bounded difference: $t(a, b) = max(0, a + b - 1)$

Hamacher's Function: $t(a, b) = \frac{ab}{r+(1-r)(a+b-ab)}$

The t-norm operation is also used for the definition of the cartesian product $A = A_1 \times A_2 \times \ldots \times A_n$ of *n* fuzzy subsets of *X* as a fuzzy subset of the cartesian product $X = X_1 \times X_2 \times \ldots \times X_n$ using:

$$A(x_1, x_2, \ldots, x_n) = t[A_1(x_1), A_2(x_2), \ldots, A_n(x_n)]$$

with $x_i \in X_i$, $i \in N_n$

The fuzzy union of two fuzzy sets is defined analogously to fuzzy intersection using a t-conorm *u*, a function that satisfies the following conditions.

Boundary condition: $u(a, 0) = a$

Monotonicity: $\forall a, b, d \in [0, 1]$, *with* $b \leq d$ *is* $u(a, b) \leq u(a, d)$

Commutativity: $\forall a, b \in [0, 1]$ *is* $u(a, b) = u(b, a)$

Associativity: $\forall a, b, d \in [0, 1]$ *is* $u(a, u(b, d)) = u(u(a, b), d)$

Examples of t-conorms are the following:

Standard union: $u(a, b) = max(a, b)$

Algebraic sum: $u(a, b) = a + b - ab$

Bounded sum: $u(a, b) = min(1, a + b)$

The operations of fuzzy complement, intersection and union extend classical logic into fuzzy logic. Every fuzzy powerset $\mathcal{F}$*(X)* can be considered as a lattice in which the t-norm plays the role of the meet (infimum), while the t-conorm plays the role of the join (supremum). Generally speaking, given a t-norm, there is always a fuzzy complement and a fuzzy union such that the lattice is distributed and complemented under this triple and thus it is a *De Morgan algebra*.

Another important operation, used in fuzzy logic is the *fuzzy implication*, that gives a truth value to the predicate $A \Rightarrow B$ when the truth values of the predicates *A* and *B* are known. Actually, it is an extension of the standard implication since it represents clauses of the form ”if A then B”.A fuzzy implication is a function ω of the form $\omega : [0, 1] \times [0, 1] \rightarrow [0, 1]$. In standard logic it is implemented using the formula $\omega(a, b) = \bar{a} \vee b$ or $\omega(a, b) = max\{x \in 0, 1 \mid a \wedge x \leq b\}$. Extending this definition in fuzzy logic, operation $\omega_{u,c}(a, b) = u(c(a), b)$ is defined, where *u* and *c* is a fuzzy union and a fuzzy complement, respectively. Alternatively, fuzzy implication can be defined using

$$\omega_t(a, b) = \sup\{x \in [0, 1] : t(a, x) \leq b\} \tag{4.1}$$

where *t* is a t-norm.

Let $X_1, X_2, \ldots, X_d$ be crisp sets. A *fuzzy relation*, $R : X_1 \times X_2 \times \ldots \times X_d \rightarrow [0, 1]$ is defined as a fuzzy subset of the cartesian product $X_1 \times X_2 \times \ldots \times X_d$. The membership degree of each element vector $(x_1, x_2, \ldots, x_d) \in X_1 \times X_2 \times, \ldots, X_d$ in the fuzzy relation *R* is the degree in which $x_1, x_2, \ldots, x_d$ are related in terms of *R*. The representation of fuzzy relations by matrices $R = [r_{ij}]$ is used in the case that the universe of discourses are finite.

The basic operations defined on fuzzy relations are the *inverse* and the *composition*. The inverse relation of $R(X, Y)$ is the fuzzy relation $R^{-1}(Y, X)$ with $R^{-1}(y, x) = R(x, y)$ for every $x \in X$ and $y \in Y$. The membership matrix that represents R^{-1} is the inverse matrix of *R*. The $\sup -t$ composition of two fuzzy relations $R_1 : X \times Y \ \rightarrow [0, 1]$ and $R_2 : Y \times Z \ \rightarrow [0, 1]$ is defined by

$$[R_1 o^t R_2](x, z) = \sup_{y \in Y} t[R_1(x, y), R_2(y, z)], \tag{4.2}$$

while the $\inf -\omega_t$ composition is defined by

$$[R_1 o^{\omega_t} R_2](x, z) = \inf_{y \in Y} \omega_t[R_1(x, y), R_2(y, z)] \tag{4.3}$$

where *t* is a t-norm.

All the properties of crisp relations are extended for fuzzy relations. We will now give the extensions for reflexive, symmetric and transitive relations. A fuzzy relation *R* is *reflexive* iff $R(x,x) = 1$ for all $x \in X$. Moreover, it is *symmetric* iff $R(x,y) = R(y,x)$ for all $x, y \in X$. It is also *sup-t transitive* iff

$$R(x,z) \geq \sup_{y \in Y} t[R(x,y), R(y,z)].$$

It has been proved (Klir 1995) that if a fuzzy relation *R* defined on X^2 with $|X| = n \geq 2$, is reflexive, then $R_T = R^{(n-1)}$, where R_T is the transitive closure of *R*.

4.0.7 Fuzzy Propositional Logic

The main difference between classical propositions and fuzzy propositions is in the range of their truth values. While each classical proposition is required to be either true or false, the truth or falsity of fuzzy propositions is a matter of degree. Assuming that truth and falsity is expressed by values 0 or 1, respectively, the degree of truth of each fuzzy proposition is expressed by a number in the unit interval [0,1]. There are various fuzzy propositions, which are classified into the following four types.

1. Unconditional and unqualified propositions
2. Unconditional and qualified propositions
3. Conditional and unqualified propositions
4. Conditional and qualified propositions

In this chapter we will discuss the interpretation of the third type, conditional and unqualified propositions.

Propositions *p* of the conditional and unqualified type are expressed by the canonical form

p: If *X* is *A*, $\rightarrow$ *Y* is *B*,

where *X*, *Y* are variables whose values are in the sets *X*, *Y*, respectively, and *A*, *B*, are fuzzy sets on *X*, *Y*, respectively. These propositions may also be viewed as propositions of the form

$\langle X, Y \rangle \in R$

where *R* is a fuzzy set on $X \times Y$ that is determined for each $x \in X$ and each $y \in Y$ by the formula

$$R(x, y) = \omega[A(x), B(y)], \tag{4.4}$$

where *R* expresses the relationship between the variables *X* and *Y* involved in the given fuzzy proposition. For each $x \in X$ and each $y \in Y$, the membership grade $R(x, y)$ represents the truth value of the proposition

p_{xy}: If $X = x$, Then $Y = y$

Now, the truth values of the propositions "$X = x$" and "$Y = y$" are expressed by the membership grades $A(x)$ and $B(y)$, respectively. Consequently, the truth value of the proposition p_{xy} , given by $R(x, y)$, involves a fuzzy implication in which $A(x)$ is the truth value of the antecedent and $B(y)$ is the truth value of the consequent.

Assume that *R* is a fuzzy relation on $X \times Y$ and A', B' are fuzzy sets on *X* and *Y*, respectively. Then if R and A' are given we can obtain B' by the equation

$$B'(y) = \sup_{x \in X} t[A'(x), R(x, y)] \tag{4.5}$$

for all $y \in Y$. This equation, which can also be written in the matrix from as

$$B' = A' \circ R, \tag{4.6}$$

is called the compositional rule of inference. This procedure is called the generalised fuzzy modus ponens.

The fuzzy relation employed in Eq.(4.5) is usually not given directly, but in some form. In the case that the relation is embedded in a single conditional fuzzy proposition, then is determined using the fuzzy implication operator, Eq.(4.4). A more general case, in which the relation emerges from several conditional fuzzy propositions, is as follows:

Rule 1: If X is A_1, Then Y is B_1

Rule 2: If X is A_2, Then Y is B_2

Rule 3: If X is A_3, Then Y is B_3

....

Rule n: If X is A_n, Then Y is B_n

As previously described, any conditional (If-Then) fuzzy proposition can be expressed in terms of a fuzzy relation R between the two variables involved. One way to determine R is using the fuzzy implication, which operates on fuzzy sets involved in the fuzzy proposition. However, the problem of determining R for a given conditional fuzzy proposition can be detached from fuzzy implications and determine R using fuzzy relational equations.

As described, the equation to be solved for fuzzy modus ponens has the form

$$B = A \circ^t R, \tag{4.7}$$

where A and B are given fuzzy sets that represent, respectively, the IF-part and the THEN-part in the conditional fuzzy proposition involved and *t* is a *t*-norm. It will be proved in the following section that Eq.(4.7) is solvable for the interpretation of inference rules in the respective fuzzy extension of propositional logics.

4.0.8 Fuzzy Reasoning

In the following section we present a complete algorithm for solving fuzzy relational equations for the interpretation of inference rules in the respective fuzzy extension of propositional logics. The proposed interpretation algorithm is realised using a hybrid neurofuzzy architecture Fig. 4.1.

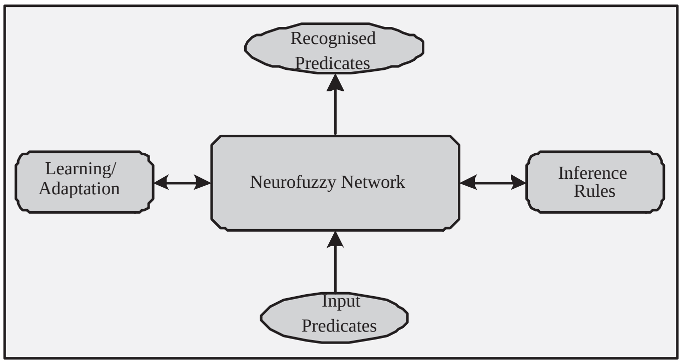

Figure 4.1: The Neurofuzzy Architecture.

Fuzzy systems are numerical model-free estimators. While neural networks encode sampled information in a parallel-distributed framework, fuzzy systems encode structured, empirical (heuristic) or linguistic knowledge in a similar numerical framework [KY95]. Although they can describe the operation of the system in natural language with the aid of human-like if-then rules, they do not provide the highly desired characteristics of learning and adaptation. The use of neural networks in order to realise the key concepts of a fuzzy logic system enriches the system with the ability of learning and improves the susymbolic to symbolic mapping [LL95].

The proposed neurofuzzy network is supported by an adaptation algorithm. This algorithm uses predefined input-output data to i) initiate and ii) adapt the weights of the fuzzy propositional rules. It is important to state that using the adaptation algorithm we are not altering the knowledge or generating new knowledge, but we refine the existed knowledge to achieve the optimal behaviour. Finally, using the adaptation algorithm we incorporate uncertainty by using degrees of confidence. The degree of confidence measures the belief of the existence of the specific concept or relation, since real-life applications involve uncertainty and fuzzy hypothesis.

Neurofuzzy Network Let $y = [y_1, y_2, ..., y_m]$ denote a fuzzy set defined on the set of output predicates, the truth of which will be examined. Actually, each y_i represents the degree in which the *i*-th output fuzzy predicate is satisfied. The input of the proposed neurofuzzy network is a fuzzy set $x = [x_1, x_2, ..., x_n]$ defined on the set of the input predicates, with each x_i representing the degree in which the *i*-th input predicate is detected. The proposed network represents the association $f : X \to Y$ which is the knowledge of the system, in a neurofuzzy structure. After the evaluation of the input predicates, some output predicates represented in the knowledge of the system can be recognized with the aid of fuzzy systems' reasoning [KY95]. One of the widely used ways of constructing fuzzy inference systems is the method of approximate reasoning which can be implemented on the basis of compositional rule of inference [KY95]. The need for results with theoretical soundness lead to the representation of fuzzy inference systems on the basis of generalised sup-*t* norm compositions [ST99], [HP96].

The class of *t*-norms has been studied by many researchers [HP96], [LL95]. Using the definition ω_t in Eq.(4.1) two additional operators $\hat{\omega}_t, \check{\omega}_t : [0,1] \times [0,1] \to [0,1]$, are defined by the following relations:

$$\hat{\omega}_t(a,b) = \begin{cases} 1 & a < b \\ a\hat{\otimes}^t b & a \geq b \end{cases} \tag{4.8}$$

$$\check{\omega}_t(a,b) = \begin{cases} a\check{\otimes}^t b & a \geq b \\ 1 & a < b \end{cases} \tag{4.9}$$

where $a\hat{\otimes}^t b = \sup\{x \in [0,1] : \ t(a,x) = b\}, \ a\check{\otimes}^t b = \inf\{x \in [0,1] : \ t(a,x) = b\}$.

With the aid of the above operators, compositions of fuzzy relations can be defined. These compositions are used in order to construct fuzzy relational equations and represent the rule-based symbolic knowledge with the aid of fuzzy inference [TSK03]. Let *X, Z, Y* be three discrete crisp sets with cardinalities *n*, *l* and *m* respectively, and $A(X,Z)$, $B(Z,Y)$, be two binary fuzzy relations. The definitions of sup-*t* and inf-$\check{\omega}_t$ compositions are given in Eq.(4.2,4.3)

Let us now proceed to a more detailed description of the proposed neurofuzzy architecture Fig. 4.2. It consists of two layers of compositional neurons which are extensions of the conventional neurons [ST99]. While the operation of the conventional neuron is described by the equation:

$$y = a\left(\sum_{i=1}^{n} w_i x_i + \vartheta\right) \tag{4.10}$$

where is non-linearity, ϑ is threshold and w_i are the weights, the operation of the sup-*t* compositional neuron is described by the equation:

$$y = a' \left\{ \sup_{j \in N_n} t(x_i, w_i) \right\} \tag{4.11}$$

where *t* is a *t*-norm and is the following activation function

$$a'(z) = \begin{cases} 0 & x \in (-\infty, 0) \\ x & x \in [0, 1] \\ 1 & x \in (0, +\infty) \end{cases} \tag{4.12}$$

A second type of compositional neuron is constructed using the $\hat{\omega}_t$ operation. The neuron equation is given by:

$$y = a' \left\{ \inf_{j \in N_n} \hat{\omega}_t(x_i, w_i) \right\} \tag{4.13}$$

The proposed architecture is a two-layer neural network of compositional neurons Fig. 4.2. The first layer consists of the inf-$\hat{\omega}_t$ neurons and the second layer consists of the sup-*t* neurons. The system takes as input, predicates, and gives to the output the recognized output predicates. The first layer computes the antecedents of the mapping rules, while the second implements the fuzzy reasoning using the fuzzy modus ponens schema.

The rules are used to initialise the neurofuzzy network (giving its initial structure and weights). During the learning process the number of neurons in the hidden layer and the weights of the two layers may change with the aid of a learning with the objective of the error minimisation. The learning algorithm that supports the above network is applied in each layer independently. During the learning process, the weight matrices are adapted in order to approximate the solution of the fuzzy relational equation describing the association of the input with the output. Using a traditional minimisation algorithm (for example the steepest descent), we cannot take advantage of the specific character of the problem. The algorithm that we use is based on a more sophisticated credit assignment that ”blames” the neurons of the network using the knowledge about the topographic structure of the solution of the fuzzy relation equation [ST99]. After the learning process, the network keeps its transparent structure and the new knowledge represented in it can be extracted in the form of mapping If-Then rules.

Evolution In the process of knowledge adaptation, the If-Then rules are inserted into the proposed neurofuzzy system. This refers to automatically transforming the structured knowledge provided by the knowledge base in order to perform the followings:

1. Define the required input predicates as ”*input predicate(1), input predicate(2),..., input predicate(n)*”. The input predicates will define the set $X = \{x_1, x_2, ..., x_n\}$

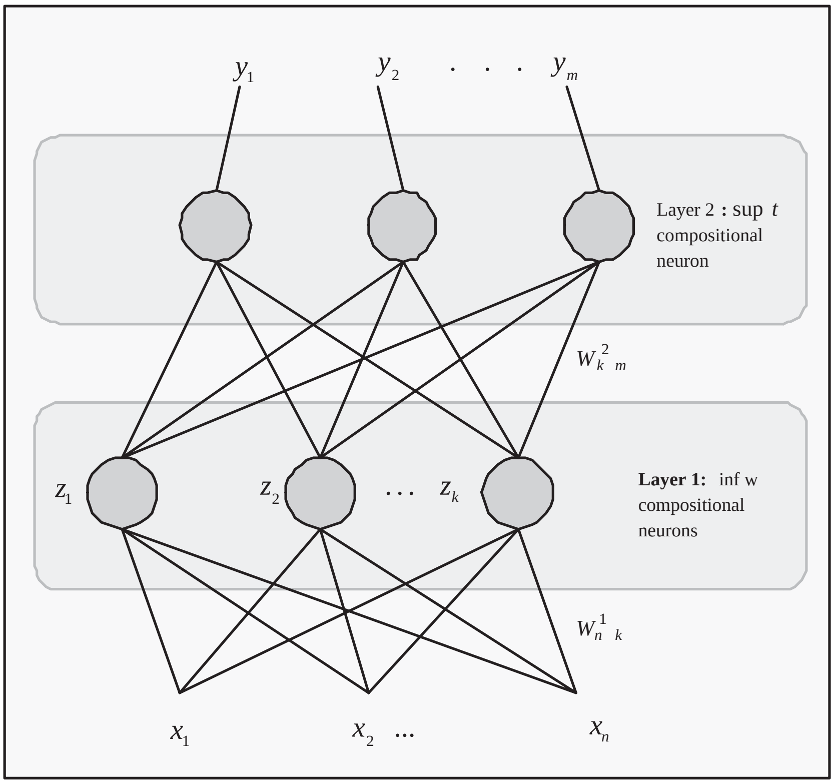

Figure 4.2: The Neurofuzzy Layers

2. Define the required output predicates as *”output predicate(1), output predicate(2),..., output predicate(n)”*. The output predicates will define the set $Y = \{y_1, y_2, ..., y_n\}$.

3. Insert the a priori knowledge given in If-Then rules of the form *”if input predicate(1) and input predicate(2) and then output predicate(5)”* into the neurofuzzy structural elements (the weights of the neurofuzzy system). The number of different antecedents (If parts of the rules) defines the set $Z = \{z_1, z_2, ..., z_n\}$. The predicates could be associated with confidence levels in order to produce the antecedents; this means that the antecedents could have the form (*input predicate(1), input predicate(2)*, 0.7, 0.9), with the 0.7 and 0.9 values corresponding to confidence levels. The above degrees are used in order to define the weights $W^1_{ij},\ i \in N_n,\ j \in N_l$ of the first layer. Furthermore, the consequences could also be associated with confidence levels, i.e. *”if input predicate(1) and input predicate(2) and then output predicate(5)”* with confidence 0.7”. These values are used in order to define the weights $W^2_{ij}\ i \in N_l,\ j \in N_m$ of the second layer.

The knowledge refinement provided by the proposed neurofuzzy system will be now described. Let $X = \{x_1, x_2, ..., x_n\}$ and $Y = \{y_1, y_2, ..., y_n\}$ be the input and output, respectively, predicate sets and let also $R = \{r_1, r_2, ..., r_n\}$ be the set of rules describing the knowledge of the system. The set of antecedents of the rules is denoted by $Z = \{z_1, z_2, , z_n\}$ (see the structure of the neurofuzzy system given in Fig. 4.2). Suppose now that a set of input-output data $D = \{A_1, B_1),\ i \in N_q\}$, where $A_i \in F(X)$ and $B_i \in F(Y)$ ($F(\star)$ is the set of fuzzy sets defined on $\star$), is given sequentially and randomly to the system (some of them are allowed to reiterate before the first appearance of some others). The data sequence is described as $(A^{(q)}, B^{(q)})$, $q \in N$, where $(A^{(i)}, B^{(i)})\ \in D$. The problem that arises is finding of the new weight matrices $W^1_{ij},\ i \in N_n,\ j \in N_l$ and $W^2_{ij},\ i \in N_l,\ j \in N_m$ for which the following error is minimised:

$$\varepsilon = \sum_{i \in N_q} \parallel B_i - y^i \parallel \tag{4.14}$$

where $y^I, i \in N_q$ is the output of the network when the input A_1 is given. The process of the minimisation of the above error is based on the resolution of the following fuzzy relational equations:

$$W^1 \circ^{\hat{\omega}_t} A = Z \tag{4.15}$$

$$Z \circ^t W^2 = B \tag{4.16}$$

where *t* is a continuous *t*-norm and Z is the set of antecedents fired when the input A is given to the network.

For the resolution of the above problem the adaptation process changes the weight matrices W^1 and W^2 in order to approximate a solution of the above fuzzy relational equations. During its operation the proposed network can generalise in a way that is inspired from the theory of fuzzy systems and the generalised modus ponens. Let us here describe the adaptation of the weights of the second layer (the adaptation of the first layer is similar). The proposed algorithm converges independently for each neuron. For simplicity and without loss of generality, let us consider only the single neuron case. The response of the neuron $f^{(k)}$ at time *k* is given by:

$$f^{(k)} = \sup_{i \in N_l} t\left(Z_i^{(k)}, w_i^{(k)}\right) \tag{4.17}$$

where $w_i^{(k)}$ are the weights of the neuron and $z_i^{(k)}$ the input, at time *k*. The desired output at time *k* is $B_i^{(k)}$. The algorithm has as following:

Initialise the weights as $w_i^{(0)}$, $i \in N_l$.

Process the input $z^{(k)}$ and the desired output $B^{(k)}$, compute the response of the network $f^{(k)}$ and update the weight accordingly (on-line variant of learning):

$$w_I^{(k+1)} = W_i^{(k)} + \Delta w_i^{(k)}$$

$$\Delta W_i^{(k)} = \eta l_s$$

$$l_s = \begin{cases} \eta_1 \left(\check{\omega}_t \left(z_i^{(k)}, B^{(k)}\right) - w_i^{(k)}\right), & if\ w_i^{(k)} < \check{\omega}_t \left(z_i^{(k)}\right) \\ \eta_2 \left(w_2^{(k)} - \hat{\omega}_t \left(z_i^{(k)}, b^{(k)}\right)\right), & if\ w_i^{(k)} > \hat{\omega}_t \left(z_i^{(k)}, B^{(k)}\right) \end{cases}$$

where η, η_1, η_2 are the learning rates. The adaptation is activated only if $|\varepsilon(B^{(k)}, y^{(k)}| > \varepsilon_c$, where ε is an error constant.

If the *t*-norm is Archimedean, then the learning signal is computed as:

$$l_s = \left(\hat{\omega}_t \left(z_i^{(k)}, B^{(k)}\right) - w_i^{(k)}\right),\ if\ z_i^{(k)} \geq b_i^{(k)}\ and\ z_i^{(k)} \neq 0,\ else\ l_s = 0$$

With the aid of the above learning process (and similar for the first layer, since the operator $\hat{\omega}_t$ is also used in order to solve the fuzzy relational equation of the first layer [3]), the network approximates the solutions of the fuzzy relational equations given above and thus minimise the error.

Example of Learning, Adaptation and Evolution

In this section we present some simulation results illustrating the operation of the proposed adaptation algorithm. Also, we demonstrate the performance of the proposed neurofuzzy network before and after the adaptation procedure. It operates on the meta-knowledge of the rules, the weights. It does not alter the meaning of the knowledge but adapts the existed knowledge according to predefined input-output data, which are provided by the experts. The results of the algorithm are new adapted weights that operate better in the working context. The new weights are replacing the old weights in the knowledge base. Every time the context is changed the weights must be adapted since the proposed propositional rules are context sensitive. For the system's adaptation demonstration, we consider a small set of the described inputs, outputs and propositional rules. We have 10 input predicates, $X = \{x_1, x_2, ..., x_{14}\}$, 8 antecedents, $Z = \{z_1, z_2, ..., z_8\}$, and 3 output predicates, $Y = \{y_1, y_2, y_3\}$. The rules of the network $R = \{R_1, R_2, ..., R_{10}\}$ are shown in the Table 4.1.

The weights of the two layers are initialised using Table 4.1. The antecedent part is used for $\mathbf{W}^1$, and the output (consequence) is used for $\mathbf{W}^2$. The two matrices are shown below:

$$\mathbf{W}^1 = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}, \mathbf{W}^2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$$

We have used the Yager t-norm with parameter value *p*=2, and learning rate $\eta = 1$. The Yager norm is defined by:

$$Y_{yager}(z, w^2) = 1 - min(1, [(1-z)^p + (1-w^2)^p]1/p,\ p > 0, \tag{4.18}$$

Table 4.1: Example rules

R	*Antecedent*	*Output*
r_1	$x_1 + x_2 + x_4 + x_7 + x_9 + x_{10}$	1
r_2	$x_2 + x_4 + x_6 + x_7 + x_{14}$	1
r_3	$x_1 + x_6 + x_7 + x_{11} + x_{13} + x_{14}$	1
r_4	$x_3 + x_6 + x_8 + x_9 + x_{11}$	2
r_5	$x_3 + x_5 + x_7 + x_9 + x_{11}$	2
r_6	$x_4 + x_5 + x_7 + x_9 + x_{10} + x_{11}$	2
r_7	$x_6 + x_7 + x_{10} + x_{12} + x_{14}$	2
r_8	$x_1 + x_9 + x_{10} + x_{12} + x_{13}$	3

where p is the parameter of the norm. The yager implication is defined by the following equation:

$$Z_{yager}(w^1, x) = \begin{cases} 1 - [(1 - w^1)^p - (1 - x)^p]^1/p, & w^1 > x \\ 1, & \text{else} \end{cases} \tag{4.19}$$

The neurons are adapted independently, and after 20 iterations the weights of the two layers are given by:

$$\mathbf{W}^1 = \begin{bmatrix} .8 & .7 & 0 & .4 & 0 & 0 & 1 & 0 & .8 & 1 & 0 & 0 & 0 & 0 \\ 0 & .7 & 0 & 1 & 0 & .8 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & .7 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & .7 & 0 & 0 & .4 & 0 & 1 & 1 & 0 & .8 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & .7 & 0 & 0 & 0 \\ 0 & 0 & 0 & .5 & 1 & 0 & .7 & 0 & .8 & 1 & .7 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & .9 & 1 & 0 \end{bmatrix}, \mathbf{W}^2 = \begin{bmatrix} 0 & 0 & .8 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & .8 & 0 \\ 0 & 1 & 0 \\ 0 & .7 & 0 \\ 1 & 0 & 0 \\ .6 & 0 & 0 \end{bmatrix}$$

We observe that the adaptation procedure has refined the weights of the rules and consequently the semantics of the knowledge base. The error of the system, as shown in Fig. 4.3 became zero. In Fig. 4.3 the error performance is illustrated, using learning rates, 0.5 and 1. It can be seen that the structure of the knowledge is not changed. The algorithm only adapts the existed knowledge and is not generating new knowledge. The new weights are replacing the old weights in the knowledge base.

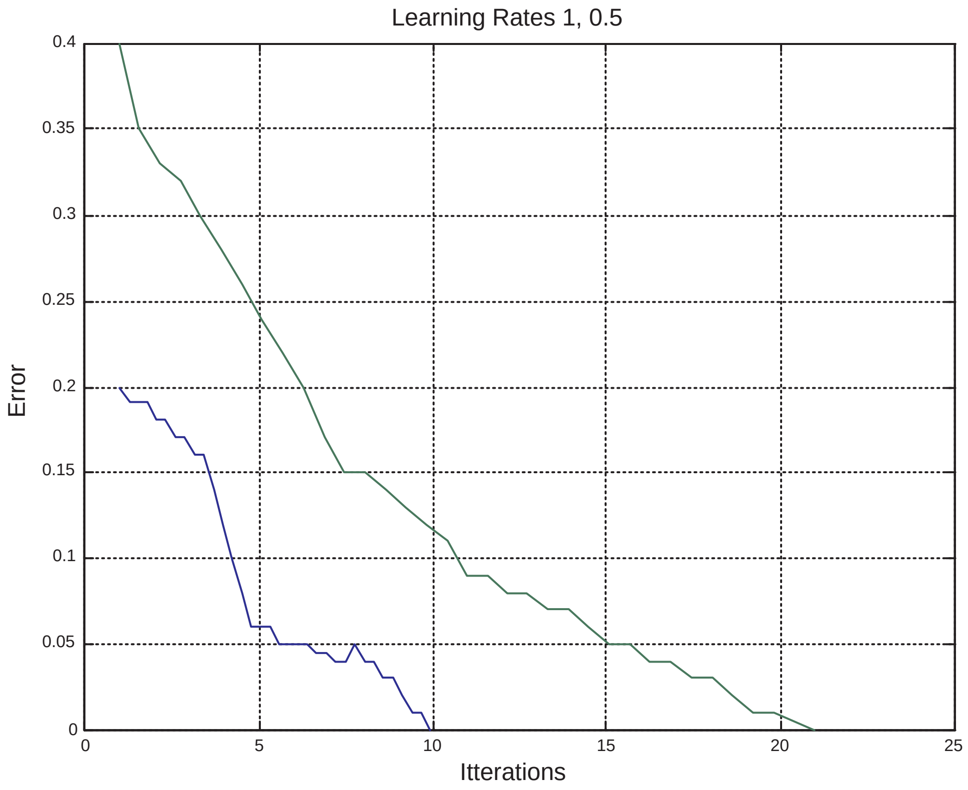

Figure 4.3: The error of the neural network during the adaptation process.

Chapter 5

Conclusions and outlook

In this first deliverable of the ”Dynamics” work package of KnowledgeWeb we elaborated the foundations of a new RDF and ontology versioning system.

We presented a methodology for RDF-based versioning that separates the management aspects from the versioning core functions. We presented blank node enrichment as a technique to identify blank nodes in models and diffs. We also explained how generic RDF versioning is beneficial for ontology language specific versioning. We sketched our implementation of the system. Finally we discussed new aspects of versioning like dealing with uncertainty and economic aspects, which might well be an integrated part of a comprehensive versioning methodology in the future.

There is still a lot of work to do to reach our goal of a broadly accepted general versioning system for semantic models.

Therefore we will execute the following research tasks in the near future:

1. Compiling a report that describes in more detail how a versioning systems for different ontology languages can be build on top of an RDF versioning system.
2. Evolving the developed RDF prototype into a full RDF versioning system.
3. Implementing a full RDF-Schema versioning system on top of this RDF versioning system. Including semantic diffs (transitive closure etc.).
4. Implementing a first prototype of a WSMO-Core versioning system on top of the RDF versioning system to meet the requirements from the Semantic Web Services scenario.
5. Integrate the Consensus Making Portal with the versioning prototype.

This way we will build a general, extendable multi-language ontology versioning system, that will help research and industry to employ ontology based technologies in dynamic settings.

Bibliography

[BD91] J.S. Brown and P. Duguid. Organizational learning and communities of practice: Towards a unified view of working, learning, and innovation. *Organization Science*, 2(1), 1991.

[Ber90] B. Berliner. CVS II: Parallelizing software development. In *Proceedings of the USENIX Winter 1990 Technical Conference*, pages 341–352, Berkeley, CA, 1990. USENIX Association.

[BG96] D. Bloom and G. Grenier. Language, employment and earnings in the united states: Spanish-english differentials from 1970 to 1990. *International Journal of the Sociology of Language*, 121:45–68, 1996.

[BKD$^+$00] J. Broekstra, M. Klein, S. Decker, D. Fensel, and I. Horrocks. Adding formal semantics to the web: building on top of rdf schema, 2000.

[BKKK87] Jay Banerjee, Won Kim, Hyoung-Joo Kim, and Henry F. Korth. Semantics and implementation of schema evolution in object-oriented databases. In *SIGMOD '87: Proceedings of the 1987 ACM SIGMOD international conference on Management of data*, pages 311–322. ACM Press, 1987.

[BLC01] Tim Berners-Lee and Dan Connolly. Delta: an ontology for the distribution of differences between rdf graphs. Technical report, W3C, 2001.

[Blo76] D. Bloor. *Knowledge and Social Imagery*. Routledge and Kegan Paul, 1976.

[Blu99] A. Blume. A learningefficiency explanation of structure in communication, 1999. Working paper.

[BM79] A. Breton and P. Mieszkowski. The economics of bilingualism. In Wallace E. Oates, editor, *The Political Economy of Fiscal Federalism*. D.C. Heath, Lexington, MA., 1979.

[Bou82] P. Bourdieu. *Ce que parler veut dire: l'économie des échanges linguistiques*. Fayard, Paris, 1982.

[Bre64] A. Breton. The economics of nationalism. *Journal of Political Economy*, 62, 1964.

[Bre97] A. Breton. *The Economics of Language*. Department of Economics, University of Toronto, Toronto, 1997.

[Bre98] A. Breton. An economic analysis of language. In *Economic Approaches to Language and Bilingualism*. The New Canadian Perspective, 1998.

[BS99] G. Bowker and S.L. Star. *Sorting Things Out: Classification and its consequences*. MIT press, Cambridge, 1999.

[CBHS04] Jeremy J. Carroll, Christian Bizer, Patrick Hayes, and Patrick Stickler. Named graphs, provenance and trust. Technical report, HP, 05 2004.

[Chi91] B. Chiswick. Speaking, reading and earnings among low-skilled immigrants. *Journal of Labor Economics*, 9, 1991.

[CK93] J. Church and I. King. Bilingualism and network externailities. *Canadian Journal of Economics*, 26(2), 1993.

[Col85] H. M. Collins. *Changing Order. Replication and Induction in Scientific Practice*. Sage, 1985.

[Col91] J. Colomer. The utility of bilingualism. *Rationality and Society*, 2:310–334, 1991.

[CS04] J. Carroll and P. Stickler. Trix: Rdf triples in xml. Technical report, HP, Nokia, 02 2004.

[D^+00] S. Decker et al. The semantic web: The roles of XML and RDF. *IEEE Internet Computing*, 4(5):63–74, 2000.

[Dal99] S. Dalmazzone. Economics of language: a network externalities approach. In A. Breton, editor, *Exploring the Economics Language*. The New Canadian Perspective, 1999.

[Fen00] Dieter Fensel. The semantic web and its languages. *IEEE Computer Society*, 2000.

[Fen01] Dieter Fensel. *Ontologies: A Silver Bullet for Knowledge Management and Electronic Commerce*. Springer, 2001.

[FGM00] Enrico Franconi, Fabio Grandi, and Federica Mandreoli. A semantic approach for schema evolution and versioning in object-oriented databases. *Lecture Notes in Computer Science*, 1861:1048–??, 2000.

[Fie00] Roy T. Fielding. *Architectural Styles and the Design of NetworkBased Software Architectures*. PhD thesis, 2000.

[G$^+$03] B. Grosof et al. Description logic programs: Combining logic programs with description logics. In *Proc of WWW2003*, pages 48–57, Budapest, Hungary, 2003. ACM.

[Gar67] H. Garfinkel. *Studies in Ethnomethdology*. Prentice-Hall, Englewood Cliffs, NJ, 1967.

[Gar88] H. Garfinkel. Respecipication: Evidence for locally produced, naturally accountable phenomena of order, logic, reason, meaning, method, etc. in and as of the essential quiddity of immortal ordinary scociety (i) - an anouncement of studies. *Sociological Theory*, 1988.

[Gar96] H. Garfinkel. Ethnomethodologys program. *Social Psychology Quaterly*, 59(1), 1996.

[Gri90] F. Grin. The economic approach to minority languages. *Journal of Multilingual and Multicultural Development*, 11:153–173, 1990.

[Gri92] F. Grin. Towards a threshold theory of minority language survival. *Kyklos*, 45:69–97, 1992.

[Gri94] F. Grin. The economics of language: Match or mismatch? *International Political Science Review*, 15, 1994.

[Gri99] F. Grin. Economics. In J. Fishman, editor, *Handbook of Language and Ethnic Identity*, pages 9–24. Oxford University Press, Oxford, 1999.

[Gri02] F. Grin. Using language economics and education economics in language education policy. Council of Europe, Strasburg, 2002.

[Gru93] T.R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5:199–220, 1993.

[GV83] G. Grenier and F. Vaillancourt. An economic perspective on learning a second language. *Journal of Multilingual and Multicultural Development*, 4:471–483, 1983.

[GV97] F. Grin and F. Vaillancourt. The economics of multilingualism: Overview of the literature and analytical framework. In W. Grabe, editor, *Multilingualism and Multilingual Communities (ARAL XVII)*, pages 43–65. Cambridge University Press,, Cambridge, 1997.

[Har95] R.G. Harris. Communications costs and trade. *Canadian Journal of Economics*, (special issue), 1995.

[Har04] R. Harry. The economics of language in a virtual integrated global economy. In *Economic Approaches to Language and Bilingualism*. The New Canadian Perspective, 2004.

[Hel99] J.F. Helliwell. Language and trade. In *Exploring the Economics of Language*. The New Canadian Perspective, 1999.

[Her84] J.C. Heritage. *Garfinkel and Ethnomethodology*. Polity Press, Cambridge, 1984.

[Her92] J.C. Heritage. Ethnomethodology. In E.F. Borgatta and M.L. Borgatta, editors, *Encyclopedia of Sociology*, volume 2. MaCMillan, 1992.

[Hoc97] T. Hocevar. Equilibria on linguistic minority markets. *Kyklos*, 28:337–357, 1997.

[Hol87] N. Holden. Language barriers as differential constraints on the international behavior of firms. In H. Tonkin and K. Johnson-Weiner, editors, *The Economics of Language Use*. Center for Research and Documentation on World Language Problems, New York, 1987.

[HP96] K. Hirota and W. Pedrycz. Solving fuzzy relational equations through logical filtering. *Fuzzy Sets and Systems*, 99:179–186, 1996.

[HR] K. Head and J. Ries. Market-access effects of trade liberalization: Evidence from the canada-u.s. free trade agreement. In R.C. Feenstra, editor, *The Effects of U.S. Trade Protection and Promotion Policies*, pages 323–342. University of Chicago Press, Chicago.

[Hue97] W. Huersch. Maintaining consistency and behaviour of object-oriented systems during evolution. In *Proc of the ACM Conf on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA97)*, pages 1–21, 1997.

[KF01] M. Klein and D. Fensel. Ontology versioning for the semantic web. In *1st Int Semantic Web Working Symp. (SWWS)*, pages 75–91, Stanford, CA, USA, 2001.

[Kle04] M. Klein. *Change Management for Distributed Ontologies*. Phd thesis, Vrije Univ. Amsterdam, 2004.

[KSO02] Atanas Kiryakov, Kiril Simov, and Damyan Ognyanov. Ontology middleware: Analysis and design. Technical report, IST Project IST-1999-10132 On-To-Knowledge, 2002.

[KY95] G. J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice-Hall, 1995.

[Lam02] D.M. Lamberton. Lenguage: A social technology? *The Economics of Language*, 2002.

[Lin93] M. Linch. *Scientific Practice and Ordinary Action*. Cambridge University Press, Cambridge, 1993.

[Lin99] M. Linch. Silence in context: Ethnomethodology and social theory. *Human studies*, 22, 1999.

[Lip02] B.L. Lipman. Language and economics. In M. Basili, N. Dimitri, and I. Gilboa, editors, *Cognitive Processes and Rationality in Economics*. 2002.

[LL95] C.-T. Lin and C.S. Lee. *Neural fuzzy Systems: A neuro-fuzzy synergism to intelligent systems*. Prentice-Hall, 1995.

[LW91] J. Lave and E. Wenger. *Situated Learning: Legitimate Peripheral Participation*. Cambridge University Press, 1991.

[MRA04] A; Decker S; Zhdanova A; Ding Y; Stollberg M Martin-Recuerda, F; Harth and S. Arroyo. Deliverable d2.1 report on requirements analysis and state-of-the-art within wp2 ontology management of the dip project. Technical report, 2004.

[NK03] N. Noy and M. Klein. Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems*, (5), 2003.

[OCK98] M. Ó Cinnéide and M. Keane. *Local Socio-Economic Impacts Associated with the Galway Gaeltacht*. Coláiste na hOllscoile Gaillimh, Gaillimh (Galway), 1998.

[Pic92] A. Pickering. *Science as Practice and Culture*. University of Chicago, 1992.

[Pra01] P. Prashant. The use of language. CSLI Pubblication, http://cslipublications.stanford.edu/pdf/1575863545.pdf, 2001.

[PS02] P. Patel-Schneider. Layering the semantic web: Problems and directions, 2002.

[RN95] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, Englewood Cliffs, NJ, 1995.

[Rod96] J.F. Roddick. A survey of schema versioning issues for database systems. *Information and Software Technology*, 37(7):383–393, 1996.

[Rub98] A. Rubinstein. Economics and language. http://arielrubinstein.tau.ac.il/papers/schwartz.pdf, 1998.

[Rub00] A. Rubinstein. *Economics and Language*. Cambridge University Press, 2000.

[SAMZ01] Tran Cao Son Sheila A. McIlraith and Honglei Zeng. Semantic web services, 2001.

[SB93] A. Sproull and Ashcroft B. *The Economics of Gaelic Language Development*. Glasgow Caledonian University, Glasgow, 1993.

[SC03] L. Schneider and J. Cunningham. Ontological foundations of natural language communication in multi-agent systems. In Vasile Palade, Robert J. Howlett, and Lakhmi Jain, editors, *Knowledge Based Intelligent Information and Engineering Systems*, LNAI-2773, pages 1403–1410. Springer, 2003.

[SD02] M. Sintek and S. Decker. TRIPLE – a query, inference, and transformation language for the semantic web. In *Proc. of ISWC2002*, Sardinia, IT, 2002.

[SD04] G. Schreiber and M. Dean. OWL web ontology language reference. `http://www.w3.org/TR/2004/REC-owl-ref-20040210/`, February 2004.

[SP91] R. Selten and J. Pool. The distribution of foreign language skills as a game equilibrium. In R. Selten, editor, *Game Equilibrium Models*. Springer, Berlin, 1991.

[SPKR96] B. Swartout, R. Patil, K. Knight, and T. Russ. Toward distributed use of large-scale ontologies. In *Proceedings of the Tenth Knowledge Acquisition for Knowledge-Based Systems Workshop, KAW '96*, Banff, Alberta, Canada, November 1996.

[SS04] S. Staab and R. Studer, editors. *Handbook on Ontologies in Information Systems*. Int Handbooks on Information Systems. Springer, 2004.

[ST99] G.B. Stamou and S. G. Tzafestas. Fuzzy relation equations and fuzzy inference systems: an inside approach. *IEEE Trans. on Systems, Man and Cybernetics*, 99:694–702, 1999.

[Sta03] Starlab. *Systems Technology and Applications Research Laboratory home page*. Faculty of Sciences, Department of Computer Science, Vrije Universiteit Brussel, http://www.starlab.vub.ac.be/default.htm, 2003.

[Sto04] L. Stojanovic. *Methods and Tools for Ontology Evolution*. PhD thesis, Univ Karlsruhe (TH), 2004.

[sub04] Subversion (http://subversion.tigris.org/), 2004.

[Tic85] W. Tichy. RCS – a system for version control. *Software Practice & Experience*, 15(7):637–654, July 1985.

[TSK03] V. Tzouvaras, G. Stamou, and S. Kollias. Knowledge refinement using fuzzy compositional neural networks. 2003.

[Wil75] O.E. Williamson. *Markets and Hierarchies*. Free Press, New York, 1975.

[Y^+03] K-P. Yee et al. Faceted metadata for image search and browsing. In *Proc of ACM SIGHCI Conf on Human Factors in Comp. Sys. Archive*, Florida, USA, 2003.