2-2005

Specification of a Common Framework for Characterizing Alignment

Paolo Bouquet

Marc Ehrig

Jerome Euzenat

Enrico Franconi

Pascal Hitzler
pascal.hitzler@wright.edu

See next page for additional authors

Repository Citation

Bouquet, P., Ehrig, M., Euzenat, J., Franconi, E., Hitzler, P., Krotzsch, M., Serafini, L., Stamu, G., Sure, Y., & Tessaris, S. (2005). Specification of a Common Framework for Characterizing Alignment. . https://corescholar.libraries.wright.edu/cse/239

Authors
Paolo Bouquet, Marc Ehrig, Jerome Euzenat, Enrico Franconi, Pascal Hitzler, Markus Krotzsch, Luciano Serafini, Giorgos Stamu, York Sure, and Sergio Tessaris

knowledge**web**

realizing the semantic web

D2.2.1 Specification of a common framework for characterizing alignment

Coordinator: Paolo Bouquet (U. Trento)
Marc Ehrig (U. Karlsruhe), Jérôme Euzenat (INRIA), Enrico Franconi (FU Bolzano), Pascal Hitzler (U. Karlsruhe), Markus Krötzsch (TU Dresden), Luciano Serafini (U. Trento), Giorgos Stamou (NTU Athens), York Sure (U. Karlsruhe), Sergio Tessaris (FU Bolzano)

Abstract.
Definition of a common framework for characterizing alignment of heterogeneous information. This report describes various approaches towards this goal and shows the relations between them. It also provides a description of the alignment sructure and process.

Keyword list: ontology alignment, data integration, peer-to-peer

Document Identifier	KWEB/2004/D2.2.1/v2.0
Project	KWEB EU-IST-2004-507482
Version	v2.0
Date	February 2, 2005
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
Contact person: Dieter Fensel
E-mail address: dieter.fensel@uibk.ac.at

École Polytechnique Fédérale de Lausanne (EPFL)
Computer Science Department
Swiss Federal Institute of Technology
IN (Ecublens), CH-1015 Lausanne
Switzerland
Contact person: Boi Faltings
E-mail address: boi.faltings@epfl.ch

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

Freie Universität Berlin (FU Berlin)
Takustrasse 9
14195 Berlin
Germany
Contact person: Robert Tolksdorf
E-mail address: tolk@inf.fu-berlin.de

Free University of Bozen-Bolzano (FUB)
Piazza Domenicani 3
39100 Bolzano
Italy
Contact person: Enrico Franconi
E-mail address: franconi@inf.unibz.it

Institut National de Recherche en
Informatique et en Automatique (INRIA)
ZIRST - 655 avenue de l'Europe -
Montbonnot Saint Martin
38334 Saint-Ismier
France
Contact person: Jérôme Euzenat
E-mail address: Jerome.Euzenat@inrialpes.fr

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

Learning Lab Lower Saxony (L3S)
Expo Plaza 1
30539 Hannover
Germany
Contact person: Wolfgang Nejdl
E-mail address: nejdl@learninglab.de

National University of Ireland Galway (NUIG)
National University of Ireland
Science and Technology Building
University Road
Galway
Ireland
Contact person: Christoph Bussler
E-mail address: chris.bussler@deri.ie

The Open University (OU)
Knowledge Media Institute
The Open University
Milton Keynes, MK7 6AA
United Kingdom
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
Contact person: Asunción Gómez Pérez
E-mail address: asun@fi.upm.es

University of Liverpool (UniLiv)
Chadwick Building, Peach Street
L697ZF Liverpool
United Kingdom
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
Contact person: Hamish Cunningham
E-mail address: hamish@dcs.shef.ac.uk

Vrije Universiteit Amsterdam (VUA)
De Boelelaan 1081a
1081HV. Amsterdam
The Netherlands
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
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
Contact person: Carole Goble
E-mail address: carole@cs.man.ac.uk

University of Trento (UniTn)
Via Sommarive 14
38050 Trento
Italy
Contact person: Fausto Giunchiglia
E-mail address: fausto@dit.unitn.it

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

Work package participants

The following partners have taken an active part in the work leading to the elaboration of this document, even if they might not have directly contributed to writing parts of this document:

Centre for Research and Technology Hellas
École Polytechnique Fédérale de Lausanne
Free University of Bozen-Bolzano
Institut National de Recherche en Informatique et en Automatique
National University of Ireland Galway
Universidad Politécnica de Madrid
University of Innsbruck
University of Karlsruhe
University of Manchester
University of Sheffield
University of Trento
Vrije Universiteit Amsterdam
Vrije Universiteit Brussel

Changes

Version	Date	Author	Changes
0.1	27-04-2004	Enrico Franconi & Sergio Tessaris (FUB)	creation
0.2	11-05-2004	Paolo Bouquet (UniTN)	editing/integration
0.3	20-05-2004	Jérôme Euzenat (INRIA)	general view and dimensions of alignment
0.4	02-06-2004	Giorgos Stamou (NTUA)	semantics of fuzzy mappings
0.5	12-06-2004	Jérôme Euzenat (INRIA)	dressing
0.6	10-05-2004	Paolo Bouquet (UniTN)	general characterization + dressing
1.0	15-06-2004	Jérôme Euzenat (INRIA)	delivered to quality assessor
1.1	10-07-2004	Paolo Bouquet (UniTN)	delivered to quality controler
1.2	26-07-2004	Jérôme Euzenat (INRIA)	added executive summary
1.3	14-12-2004	Paolo Bouquet (UniTN) & Pascal Hitzler (UniKarl)	revised and delivered to quality assessor
1.4	07-01-2005	Jérôme Euzenat (INRIA)	delivered to quality control
1.5	17-01-2005	Jérôme Euzenat (INRIA)	few corrections
2.0	02-02-2005	Jérôme Euzenat (INRIA)	implemented QC remarks

Executive Summary

In a distributed and open system (like the semantic web), heterogeneity cannot be avoided. Some of the heterogeneity problems can be solved by aligning heterogeneous ontologies. This is illustrated through a number of use cases of ontology alignment (in deliverable D2.2.3).

In this document the problem of overcoming heterogeneity is equated to the problem of discovering, expressing and using ontology alignments. The goal of this document is to provide a common framework for future work in this domain. It first provides a definition of many of the terms used in the domain. In particular, aligning ontologies consists of providing the corresponding entities in these ontologies. These correspondences are called mappings.

We identify four levels at which heterogeneity occurs: syntactic, teminological, conceptual and semiotic. We focus on the terminological and conceptual levels and do not consider the other aspects in this document. So the ontologies are considered as expressed in the same (or at least comparable) languages.

Then we provide definitions for the nature of alignments through the approximation relations between the aligned ontologies and the structure and semantics of mappings.

An alignment is a set of mappings expressing the correspondence between two entities of different ontologies through their relation and a trust assessment. The relation can be equivalence as well as specialisation/generalisation or any other kind of relation. The trust assessment can be boolean as well as given by other measures (e.g., probabilistic or symbolic measures) .

A very general categorical framework is provided for situating alignment, merging and their relations. A more precise semantics is then given for these mappings in the context of distributed systems. This framework is instanciated in model theoretic terms for crisp mappings and fuzzy mappings. This semantics allows to fix the goal of the alignment process and to ground the use of the produced alignments in order to merge and transform ontologies or translate data flows. This will be the goal of future work.

We then turn to the characterisation of the alignment process which takes two ontologies and produces such an alignment. It is characterised by a number of dimensions applying to the input ontologies, input alignments (when the task is alignment completion), method parameters, output alignment and the alignment process itself. Specifying each of these dimensions enables to consider particular applications or methods. It thus provide a basis for evaluating alignment methods described in deliverable D2.2.3 by defining what are input and output of the alignment process. Designing benchmarks will be the goal of our future work (deliverable D2.2.2).

Contents

Chapter 1

Introduction

1.1 Objectives of this document

The goal of this document is to provide a common framework for future work in the domain of semantic interoperability. In this document the problem of overcoming heterogeneity is equated to the problem of discovering, expressing and using mappings across ontologies.

The main contribution is a general characterization of what an alignment and an alignment process are in the context of the Semantic Web enterprise, and a specification of a formal semantics. The framework aims at the greatest possible generality, but will not cover approaches and models which fail to fulfill a high level requirement of any semantic web development, namely that mappings – the result of the alignment process – should have an explicit and formal semantics, as this is the minimal conditions for their usability in any semantic-based application.

This common framework is to be used by the partners of the Knowledge Web work package 2.2 and other Knowledge web groups as a reference document for models, languages and techniques related to the problem of aligning heterogeneous information. It is not a goal of this document to provide any detail or model of how a mapping between heterogeneous representations (e.g. ontologies) can be discovered, nor how mappings can be used in any specific application (e.g., data integration, query answering). The actual discovery and usage of mappings is the object of deliverable D2.2.3. Deliverable 2.2.2 will use this framework for characterising benchmark tests.

1.2 Terminology

The framework presented in this document builds on top of a lot of recent work on the problem of semantic interoperability. In this area, different authors use different words to refer to similar concepts, and vice versa sometimes different concepts are referred to by the same name. In this section, we provide a tentative and partial glossary with the definition of terms as they will be used in the rest of the document and should be used within the Knowledge web work package 2.2.

Mapping: a formal expression that states the semantic relation between two entities belonging to different ontologies. When this relation is oriented, this corresponds to a restriction of the usual mathematical meaning of mapping: a function (whose domain is a singleton). Mappings are discussed at length in Chapter 5.

Ontology Alignment: a set of correspondences between two or more (in case of multi-alignment) ontologies (by analogy with DNA sequence alignment). These correspondences are expressed as mappings. Alignments are detailed in Chapter 3.

Ontology Coordination: broadest term that applies whenever knowledge from two or more ontologies must be used at the same time in a meaningful way (e.g. to achieve a single goal).

Ontology Transformation: a general term for referring to any process which leads to a new ontology o' from an ontology o by using a transformation function t. Transformations and the like are the subject of further work in this work package.

Ontology Translation: an ontology transformation function t for translating an ontology o written in some language L into another ontology o' written in a distinct language L'.

Ontology Merging: the creation of a new ontology o_m from two (possibly overlapping) source ontologies o' and o''. This concept is closely related to that of *integration* in the database community.

Ontology Reconciliation: a process that harmonizes the content of two (or more) ontologies, typically requiring changes on one of the two sides or even on both sides [Hameed *et al.*, 2004].

Meaning Negotiation: the protocol through which two agents (either human or artificial) agree on the changes required to reconciliate their ontologies.

1.3 Structure of the document

This deliverable will first consider the types of heterogeneity that may occur in the semantic web and how to overcome them through alignment (Chapter 2). It will then propose a general structure (Chapter 3) for these alignments which is exploited in a categorical framework (Chapter 4). The semantics is then refined in the context of a model theoretic characterisation of mappings in distributed systems (Chapter 5). The framework ends with a characterization of the alignment process (Chapter 6).

Chapter 2

Semantic heterogeneity

In a distributed and open system (like the semantic web), heterogeneity cannot be avoided. Different actors have different interests and habits, use different tools, and use knowledge at different levels of detail. These various reasons for heterogeneity lead to different forms of heterogeneity that are considered below.

An ontology is a set of assertions that are meant to model some particular domain, in a consensual way. However, there is a huge body of evidence (in the literature of artificial intelligence, Cognitive Science, Linguistics, Epistemology, Sociology of Knowledge) that an ontology – as any other explicit representation of knowledge – always depends on a collection of implicit assumptions, no matter how hard its designers work to make it as "objective" as possible. These assumptions (including its designer's goals, background knowledge, biases, etc.) have the effect of creating several forms of heterogeneity between ontologies, even between ontologies on the same domain. We classify below (Section 2.1) the main forms of heterogeneity.

Heterogeneity affects the ontology used as well as the data exchanged. However, the actors of the semantic web have to communicate and to collaborate and thus need to mitigate this kind of heterogeneity. We consider then how alignments can be used for overcoming these problems (Section 2.2).

2.1 Forms of heterogeneity

Heterogeneity may occur at different levels, and a detailed list of all forms of possible mismatches is beyond the scope of this document (see [Giunchiglia and Walsh, 1992; Benerecetti *et al.*, 2000; Klein, 2001; Euzenat, 2001; Corcho, 2004; Hameed *et al.*, 2004; Ghidini and Giunchiglia, 2004]). However, for the sake of the definition of a common framework, we suggest that they can be classified into four main levels: syntactic, terminological, conceptual, semiotic/pragmatic. Each of them is briefly described in the following sections.

2.1.1 The syntactic level

At the syntactic level, we encounter all forms of heterogeneity that depend on the choice of the representation format. Indeed, there are several proposed formats for ontology representation (e.g. OWL, KIF), and each of them is based on a different syntax.

Some of them are syntactic sugar (e.g., n3 and RDFS, DATALOG and a subset of Prolog), some of them are more complicated and involve expressing the same thing (having the same set of models) through totally different syntax.

Example 1 (Translating from DLR to CPDL) *In order to decide query containment in the DLR description logics, [Calvanese* et al.*, 1998a] defines a mapping from the DLR logic (which introduces n-ary relations) to the CPDL logic (Propositional Dynamic Logic with Converse). These relations are represented by concepts with exactly n features to the components of the relation.*

This transformation is a consequence preserving syntactic transformation.

In this document, and more generally in the work package 2.2 of Knowledge web, we are not strongly concerned about this syntactic level, which is well understood in computer science in general. Therefore, in what follows, we will assume that the different formats can be interoperated at a syntactic level. This is typically achieved through a translation function (see Section 1.2).

As a working assumption, from now on we assume that ontologies are represented using a common syntax (e.g., OWL). However, the framework presented in this document does not depend in any essential way on this assumption.

2.1.2 The terminological level

At the terminological level, we encounter all forms of mismatches that are related to the process of naming the entities (e.g. individuals, classes, properties, relations) that occur in an ontology. Naming is the process of associating a linguistic object from a public language (namely a language that is then use to exchange information with other parties) to entities described in an ontology. This level should not be confused with the conceptual level (see below); indeed, tricky terminological mismatches may occur in situations where the involved ontologies are conceptually equivalent.

Typical examples of mismatches at the terminological level are:

- different words are used to name the same entity (*synonymy*);
- the same word is used to name different entities (*polysemy*);
- words from different languages (English, French, Italian, Spanish, German, Greek, etc.) are used to name entities;
- syntactic variations of the same word (different acceptable spellings, abbreviations, use of optional prefixes or suffixes, etc.).

In a sense, mismatches at the terminological level are not as deep as those occurring at the conceptual level (see below). However, we should notice that most real cases have to do with the terminological level (e.g., with the way different people name the same entities), and therefore this level is at least as crucial as the other one.

2.1.3 The conceptual level

At the conceptual level, we encounter mismatches which have to do with the content of an ontology. Discrepancies at this level can be analyzed in two main classes:

- *metaphysical* differences, which have to do with how the world is “broken into pieces” (i.e., what entities, properties and relations are represented in an ontology);
- *epistemic* differences, which have to do with the assertions that are made about the selected entities.

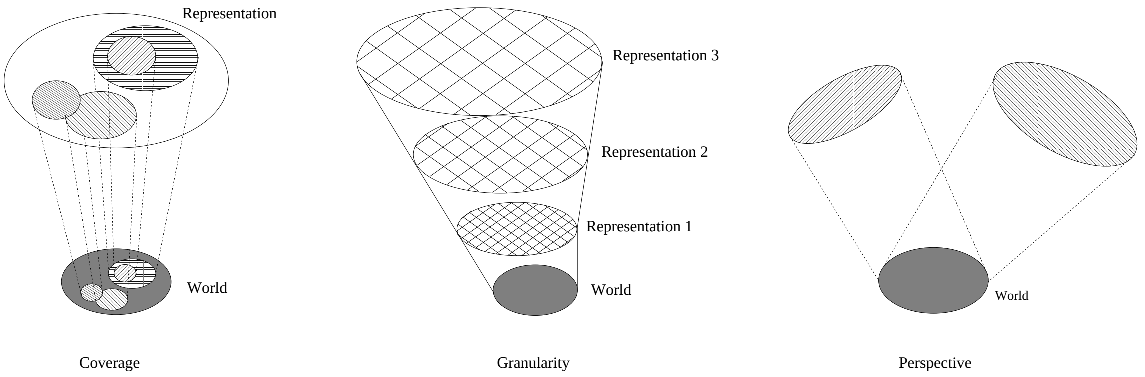


Figure 2.1: The three dimensions of heterogeneity at the conceptual level

These two kinds of differences explain, for example, why different ontologies of the same domain may start from different primitive classes, or why different ontologies may contain different (possibly contradictory) assertions about the same entities. As epistemic differences cannot be dealt with exclusively through mappings, in the rest of the document we will ignore this kind of difference, but we'll comment a bit later on the relation between the two forms of heterogeneity.

The practical forms in which metaphysical differences can arise are countless. However, following the artificial intelligence literature in this topic (in particular [Benerecetti *et al.*, 2000]), we suggest to cluster them into three abstract types:

Coverage: an ontology may differ from another as they cover different portions – possibly overlapping – of the world (or even of a single domain). For example, an ontology on sport may include car racing, whereas another may decide to ignore it as part of the sport domain; an ontology may contains properties of car racing that another disregards; and so on.

Granularity: an ontology may differ from another as the first provides a more (or less) detailed description of the same entities. For example, an ontology concerned with accounting and taxes, or delivery, would only consider the generic concept of document, while an ontology for libraries or scholars would distinguish between types of documents, e.g. books, biographies or autobiographies. Likewise, in the ontology of a Finnish, or a nivologist, there are many concepts of snow depending on how it is, while the ontology of a Tahitian or computer scientist would include significantly fewer snow related concepts.

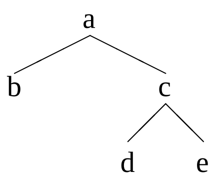
Perspective: an ontology may provide a viewpoint on some domain which is different from the viewpoint adopted in another ontology. For example, two ontologies may represent the same domain at the same level of coverage and granularity, but at different points in time (which means that the same property can hold at the time when the first ontology was designed and do not hold at the time when the other was designed, without a real epistemic disagreement), or from a different spatial perspective (what is on the right hand side from one agent's perspective may be to the left hand side for another agent facing the opposite direction).

Figure 2.1 provides a graphical representations of these three dimensions along which ontology may differ at the conceptual level.

2.1.4 The semiotic/pragmatic level

Finally, at the semiotic/pragmatic level, we encounter all the discrepancies that have to do with the fact that different individuals/communities may interpret the same ontology in different ways in different contexts.

For instance, in a context related to knowledge formalisation, a user can express knowledge under the form of class hierarchies and first order clauses and then communicate it by using an interoperability language. But if this last language expresses all the knowledge with clauses (though preserving the semantics of the assertions), the initial user will hardly recognise (and hardly understand) the semantically equivalent result (see figure 2.2). Hence, when a transformation translates between formal languages, good understanding cannot be ensured by meaning preservation (which can indeed be preserved in this case) [Euzenat, 2000; Bechhofer *et al.*, 2001]. In this case, there is no syntactic heterogeneity (because clauses are allowed in the initial model) and no conceptual nor terminological heterogeneity: only a failure to interpret the (equivalent) representation by its designer.



$$\begin{aligned}\forall x, b(x) &\implies a(x)\\ \forall x, c(x) &\implies a(x)\\ \forall x, d(x) &\implies c(x)\\ \forall x, e(x) &\implies c(x)\end{aligned}$$

Figure 2.2: Do these representations mean the same?

The intended usage has a great impact on alignment, as it can be quite risky to map entities onto each other only because they are semantically related. For example, if the concept `Europe` appears in the classification schema of a multimedia repository along a path such as `Images/B&W/Europe`, we should not conclude that it is equivalent to the concept of Europe in a geographic ontology, as the pragmatically determined meaning of the first is to be a container of black and white images of Europe, whereas the intended meaning of the second is the continent itself (this is not to say that the two things are not connected, but to warn that mappings should take intended use of each structure into account).

2.2 Overcoming heterogeneity

One common approach to the problems of heterogeneity is the definition of relations across the heterogeneous representations, in particular across ontologies. These correspondence can be used for various tasks such as merging ontologies, generating mediators, translating messages, etc. These relations can be used for transforming expression of one ontology into a form compatible with that of the other. This may happen at any level:

syntactic: through semantic-preserving transducers;
terminological: through functions mapping lexical information;
conceptual: through general transformation of the representations (sometimes requiring a complete prover for some languages);

pragmatic: through transformation taking care of the context. This path is hardly explored though ([Bouquet *et al.*, 2003; Goguen, 1999] are exceptions).

This means that any mapping across ontologies has four distinct components:

1. a *syntactic component*: the syntactic transformations needed to transform one representation format into another. As we said, this aspect is not central in this document, and therefore we will ignore it;

2. a *terminological component*: the part of a mapping that expresses terminological relations between the expressions used to name the entities to be mapped. Simple examples are: the name of two entities is the same; the two entities are named with expressions that are one the translation of the other in a different language; the name of an entity is an abbreviation of the name of the other;

3. a *conceptual component*: the part of a mapping that expresses the relation between entities in different ontologies. Simple examples are: concept c_1 in ontology O_1 is equivalent to concept c_2 in ontology O_2; concept c_1 in ontology O_1 is similar to concept c_2 in ontology O_2; individual i_1 in ontology O_1 is the same as individual i_2 in ontology O_2...

4. a *semiotic/pragmatic component*: the part of a mapping that bridges the use of entities in different ontologies. For example, that the concept c_1 used in a schema O_1 to classify a collection of documents is used in a sense defined by c_2 in ontology O_2; that the name used for a concept in a schema is taken from a lexicon L.

In work package 2.2, we restrict our attention to terminological and conceptual heterogeneity. Indeed, syntactic heterogeneity is well understood in computer science and is generally solved by proving the semantic-preserving correspondence between two languages; pragmatic heterogeneity is currently a relatively poorly structured research domain (in which we contribute anyway). The techniques for finding, expressing and using alignments at the terminological and conceptual level are relatively integrated. Finally, the conceptual part is the most studied component of a mapping, and is probably the most important one for its role in the development of the semantic web.

The use of mappings across heterogeneous ontologies requires a clear definition of their meaning. Similarly, for generating useful alignments, it is better to know what is expected from them. This is the reason why the current framework will provide a syntax and semantics for the mappings which make correspondences.

Because we are concerned here with finding alignments, this framework provides two main elements:

- a general characterization of ontology alignments (§ 3) and a formal semantic for mappings (§ 5) that is to be used when using them as well as when finding them;
- a general definition of the alignment process (§ 6) and its potential dimensions (in particular more specific constraints applying to the resulting mapping).

These issues are covered in the next chapters.

Chapter 3

Ontology alignment and mapping

For our purposes, aligning two (or more) ontologies is a process that produces a set of mappings across ontologies which allow ontology coordination (see Glossary); said differently, mappings are tools that may enable the "flow" of information across heterogeneous ontologies by describing the relations existing between entities of different ontologies.

Following the analysis we proposed in Section 2.1, here we present a very abstract characterization of mappings (Section 3.1), and then propose a structure for these mappings (Section 3.2).

3.1 General characterization of alignments

Let o and o' be two ontologies. Then, following the analysis we provided in Section 2.1, the three basic relations between ontologies can be characterized as follows:

Coverage: the two ontologies describe different (possibly overlapping) regions of the world at the same level of detail and from a unique perspective (see left hand side of Figure 2.1);

Granularity: the two ontologies describe the same region of the world from the same perspective but at different levels of detail (see central part of Figure 2.1);

Perspective: the two ontologies describe the same region of the world, at the same level of detail, but from a different perspective (see right hand side of Figure 2.1).

With respect to this description, an alignment can be viewed as an operator $\alpha(o, o')$ that:

- given two ontologies o and o' with different coverage, tells us how the two ontologies can be used together to achieve a (less partial) description of the world;
- given two ontologies o and o' with different granularity, tells us how facts in o can be systematically translated into facts of o' (for example, how a fact f belonging to o can be rewritten as a logically equivalent fact f' in o');
- given two ontologies o and o' with different perspective, tells us how a fact f in o would be seen from the perspective of o'.

Let us briefly discuss the three categories and consider a few simple examples.

3.1.1 Aligning ontologies with different coverage

Two ontologies which differ only for the portion of the world they describe can be disjoint or may overlap. In the first case, as we excluded epistemic discrepancies (in particular, inconsistencies), they can easily be viewed as a partitioned theory, which can be jointly used to provide knowledge about the world. A simple example is an ontology o about soccer and an ontology o' about cricket. If we assume that there is no intersection between the two, an alignment will tell us that the two ontologies have no relation at all, and thus operations like inclusion, merge, and so on can be performed with no harm.

The situation is slightly different when the two theories (partially or completely) overlap. A simple example is an ontology o describing team sports and another describing indoor sports, where some sports (like volleyball) may belong to both ontologies. In this case, we must be able to recognize the common part and solve possible syntactic and terminological problems. Indeed, if we exclude inconsistencies, the only potential heterogeneity between the two ontologies may concern the syntactic format (e.g., RDF Schemas and OWL) and the choice of names used to identify the common entities (e.g., individuals, classes, and so on).

3.1.2 Aligning ontologies with different granularity

Let us now consider the case of two ontologies o and o' that describe the same portion of the world, but at different level of granularity. Simple examples are: when o characterizes the position of physical objects only by two coordinates (latitude and longitude), whereas o' takes into account also a third coordinate (height above the sea level); when o expresses measures in centimeters and o' in millimiters; and so on.

For granularity, the alignment should provide a way to move from the level of representation of an ontology to the level of representation of another ontology. Model-theoretically, this operation is more complex than the operation required in the previous case (coverage), as it requires to put in relation models which are intrinsically heterogeneous (e.g., facts of the form $loc(x, y)$ in o with facts of the form $loc(x, y, t)$ in o', where x and y may be expressed in different units of measure).

3.1.3 Aligning ontologies with different perspective

Finally, let us imagine that two ontologies o and o' describe the same region of the world, at the same level of granularity but from a different perspective. A very intuitive example is a representation using indexical expressions (like “here”, “I”, “now”, “yesterday”), as the content of such an expression essentially depends on where, when, from whom it is uttered, and their correct interpretation often requires the ability of *shifting* one’s perspective. But of course there are less direct examples. For example, the supporters of two different political parties will apply opposed descriptions to the same politicians; “cold” will be applied to different climatic conditions in Finland and in Greece; and so on.

In this case, alignment should provide a way of “rotating” the perspective of an ontology, or – as we said above – to shift its viewpoint. For some forms of heterogeneity, this can be done systematically and in a relatively simple way (e.g. for indexical descriptions); however, in general the change of perspective is a very hard task for any ontology alignment method.

3.2 Structure of a mapping

In this document, we propose to see alignment as a process that starts from two representations o and o' and produces a set of mappings between pairs of (simple or complex) entities $\langle e, e' \rangle$ belonging to O and O' respectively.

Intuitively, we will assume that in general a mapping can be described as a quadruple:

$$\langle e, e', n, R \rangle$$

where:

1. e and e' are the entities between which a relation is asserted by the mapping (e.g., formulas, terms, classes, individuals);

2. n is a degree of trust (confidence) in that mapping (notice, this degree does not refer to the relation R, it is rather a measure of the trust in the fact that the mapping is appropriate ("I trust 70% the fact that the mapping is correct/reliable/…"). The trust degree can be computed in many ways, including users' feedback or log analysis;

3. R is the relation associated to a mapping, where R identifies the relation holding between e and e'. Nothing is said about the relation but that it must apply to the pair of entities. For instance, this relation can be a simple set-theoretic relation (applied to entities seen as sets or their interpretation seen as sets), a fuzzy relation (see Section 5.4), a probabilistic distribution over a complete set of relations, a similarity measure, etc.

The degree of confidence must satisfy some minimum requirements, due to the model-theoretic semantics of the basic representation language like OWL. First of all, we require that the degrees are part of a structure $\langle D, \leq, \bot, \top \rangle$ such that D is the set of degrees, $\leq$ is an order on $D \times D$ such that whatever $d \in D$, $\bot \leq d \leq \top$. This structure is applicable to a wide number of measures (e.g., boolean lattice, fuzzy degrees, probabilities, any lattice). Moreover, whatever method is associated to the computation of the degree of confidence associated to some mapping, whenever such degree is $\bot$, then this must be interpreted as if the relation is logically false, and whenever such degree is $\top$, then this must be interpreted as if the relation be logically true. Last, we require some sort of smoothness, continuity and monotonicity of the degree of confidence function from zero to one; this requires that some argument should be made about the satisfaction of the (crisp) semantic-based meanings of the basic representation even in the cases when the degree of confidence is neither zero nor one. In this deliverable, we will concentrate on the mapping construct of the representation language, which is the crucial construct involved in the alignment procedure.

This will be the role of Deliverable 2.2.5 to elaborate a better and concrete format for these alignments. The current structure will be sufficient for the purposes of the present deliverable.

Chapter 4

Categorical presentation of alignment and merge

In this section we propose a special case of what we said above and an application to the problem of generating an ontology o_m which mediates between two heterogeneous ontologies o and o'.

Ontology merging describes the process of integrating two (or more) ontologies into a single one. How this is done best is a subject of ongoing research in the Semantic Web community. In this chapter, we propose a generic solution to the question, what the result of a merging should be in the ideal case. We do this independent of a specific choice of ontology representation language, and thus provide a sort of *blueprint* for the development of algorithms applicable in practice. Our methods are taken from category theory. More precisely, we argue that ontology merging is best captured by the notion of categorical *pushout*. This presentation is a first step towards the development of practically applicable algorithms.

In this chapter we explain how merging of ontologies is captured by the pushout construction from category theory, and argue that this is a very natural approach to the problem. For this purpose, we view category theory as a universal "meta specification language" that enables us to specify properties of ontological relationships and constructions in a way that does not depend on any particular implementation. This can be achieved since the basic objects of study in category theory are the *relationships* between multiple ontological specifications, not the internal structure of a single knowledge representation.

Categorical pushouts are already considered in some approaches to ontology research [Jannink *et al.*, 1998; Kent, 2000; Schorlemmer *et al.*, 2002] and we do not claim our treatment to be entirely original. Still we have the impression that the potential of category theoretic approaches is by far not exhausted in todays ontology research. Consequently, our goal is not only to demonstrate how a concrete problem can be captured with categorical formalisms, but also to give introduction and motivation for those who did not study the mathematical framework of category theory yet. In this respect our attempts to make categories more accessible follow the spirit of [Goguen, 1991] and the current discussions on the *Information Flow Framework* [Kent, 2000].

In contrast to some of the works mentioned above, we do not try to give a comprehensive overview of even the most important categorical methods. Instead, our treatment will focus on the particular aspect of ontology merging, for which we will give both intuitive explanations and precise definitions. This reflects our belief that, at the current stage of research, it is not desirable to fade out the mathematical details of the categorical approach completely, since the interfaces to current techniques in ontology research are not yet available to their full extent. We will also keep this treatment rather general, not narrowing the discussion to specific formalisms — this added generality is one of the strengths of category theory.

We proceed as follows: In the next section, we will give the intuitions that make a categorical framework fitting the problems of ontology alignment and merging. Then we provide a short introduction to categories, together with some basic examples that we will consider throughout this text. Then we investigate how category theory deals with cartesian products and relations, which we will utilize to model "ontology mapping" and "ontology alignment" in categorical terms thus establishing the framework for the first part of any merging operation. Section 4.3 then forms the core of this note, explaining pushouts and their relevance to ontology merging. In Section 4.4 we will explain how our theoretical considerations can be used to obtain practical methods for ontology merging. The last section includes references to the literature, pointing to sources of further information on categorical and ontological issues touched on herein.

4.1 Intuition

We present the intuition behind alignment and merging in function of some approximation relation between ontologies. We give the classical interpretation of this in both model theoretic terms and categorical terms. This is informally presented in Figure 4.1.

Let us call approximation a relation between ontologies which expresses that one ontology (a) is a representation of at least the same modeled domains as another ($\alpha(o, o')$). In logic, this relation corresponds to entailment. In category theory, the ontology will be called an *object* and the relation a *morphism*. This formulation will be completed below, but one can define other relations between ontologies such as having at least one common approximated ontology. Syntactically, it is possible to provide a set of generators that will complete an ontology (e.g., adding a constraint on a class, classifying an individual), providing an approximated ontology.

Model-theoretic semantics assigns to any ontology the set of its models. If the ontology is correctly designed, the modeled domain is part of these. Model-theoretic semantics provides a formal meaning to the intuitions behind notions such as approximation: an ontology approximates another if its models contains all the models of the other (this is the standard interpretation of entailment). So, the more approximated an ontology, the less models it has[1].

In these very general terms, aligning two ontologies (o and o') consists of finding a most specific ontology ($\alpha(o, o')$) that approximates both ontologies. If one ontology is approximated by another, the result of alignment should be the latter ($\alpha(o, o') = o$). In model-theoretic terms, it amounts to finding an ontology whose set of models is maximal for inclusion and is included in the intersection of the set of models of the two aligned ontologies. In categorical terms, it means that there exists an object ($\alpha(o, o')$) and a pair of morphisms from it to the two ontologies (o and o').

Finding the alignments between two ontologies is very useful. In particular, if one wants to merge two ontologies ($\mu(o, o')$), it is sufficient to stick the non aligned part of one ontology to the aligned subpart with the other ontology. We will see in the remainder that this corresponds, in categorical terms to the push-out construction.

[1] This amounts to consider the image in the domain of interpretation as a model, not the interpretation function itself, of course.

$$\begin{tikzcd}
 & & & & & & \cdots \\
 & & & & & m \arrow[ur] & \\
 & & o' \arrow[rr] & & \mu(o,o') \arrow[ur] & & \\
 & \alpha(o,o') \arrow[ur] \arrow[rr] & & o \arrow[ur] & & & \\
a \arrow[ur] & & & & & &
\end{tikzcd}$$

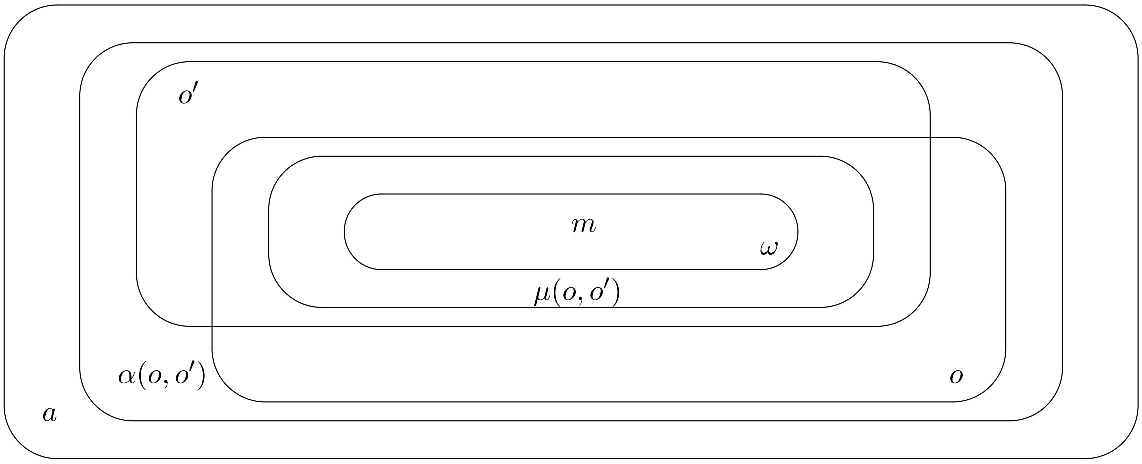


Figure 4.1: Relations between ontologies and alignment ($\alpha(o,o')$) and the corresponding model-theoretic interpretation (each ontology is represented by its set of models).

This general description of alignment, can be compared to the three cases introduced. Indeed, if taken literally

coverage corresponds to the point where $\alpha(o, o') = \emptyset$ ($\emptyset$ being the empty ontology);
granularity corresponds to the case $\alpha(o, o') = o$;
perspective corresponds to the general case presented here.

Of course, the reality is that the most frequent cases do not distinctly belong to one of these.

4.2 Categorical preliminaries

We will now express this intuition more formally in the language of category theory. For that purpose, we introduce briefly its preliminaries. This section be skipped by the knowledgeable reader.

Given two ontologies, it is for example possible that one is a sub-ontology of the other or that both of them represent the same information, i.e. that they are equivalent. Depending on the chosen representation of ontologies this can be recognized by looking at the internal structures of the given ontologies. But looking at the internal structures requires an individual treatment for each new approach to the mathematical modelling of ontologies, while a generic treatment which abstracts from the particular choice of ontology language would certainly be preferable for understanding what the result of a merging shall be.

Fortunately, there is another possibility to compare objects mathematically, which lends itself to such a generic treatment. For example, two partially ordered sets can be considered to be *equivalent*, if there exists a bijective function (i.e. one which is one-to-one and onto) between these sets which does also preserve the order (i.e. which is *monotonic*). In this case, being monotonic means that a function respects the internal structure of partially ordered sets, while bijectivity indicates the equivalence of two ordered sets. Structure-preserving functions are a typical implementation of what is called a *morphism* in category theory, and what we will recognize as a suitable substitute for the consideration of internal structures.

While monotonic functions are reasonable morphisms for comparing partial orders, other mathematical spaces may suggest different kinds of morphisms: Vector spaces are considered with linear functions, groups with group homomorphisms, geometries with movements (e.g. on the plane), topological spaces with continuous functions, etc. Considering plain sets (with no further internal structure), a morphism between two sets could be any function between them. This approach ignores most individual features of the elements of a set: functions do not distinguish whether the elements of some set are labeled a, b, c, or dog, cat, $house$. Labels are only needed to specify the function, but the essential feature of a set turns out to be its cardinality. Sets of the same size would therefore appear equivalent.

The idea that emerges from these observations is that the relationships between objects are basically captured by the morphisms that exist between them. By deciding for a particular type of morphisms, we determine which internal properties of the mathematical objects are considered "essential" (e.g. order structure or cardinality). This is the approach taken in category theory: a class of *objects* (e.g. order structures) is equipped with *morphisms* (e.g. monotonic functions), thus forming a large directed graph with objects as nodes and morphisms as arrows. Depending on the given situation, arrows can be identified with certain functions or relations between the entities that were chosen for objects, but no such concrete meaning is required. In order to constitute a category, a directed graph only has to include a *composition* operation for pairs of compatible arrows, satisfying some straightforward axioms that are typical for the composition of functions and the relational product. Let us now make this informal description precise.

Definition 1 (Category) *A category $\mathscr{C}$ consists of the following:*

- *A class*[2] *of* objects $|\mathscr{C}|$,
- *for any two objects A, $B \in |\mathscr{C}|$, a set $\mathscr{C}(A, B)$ of* morphisms *from A to B,*
- *for any three objects A, B, $C \in |\mathscr{C}|$, a* composition function
  $$\circ : \mathscr{C}(B, C) \times \mathscr{C}(A, B) \to \mathscr{C}(A, C),$$
  that combines a morphism from A to B with one from B to C to obtain a morphism from A to C,
- *for any object $A \in |\mathscr{C}|$, an* identity morphism $\mathrm{id}_A \in \mathscr{C}(A, A)$.

This data is required to satisfy the following additional axioms:

- *For all $f \in \mathscr{C}(A, B)$, $f \circ \mathrm{id}_A = f = \mathrm{id}_B \circ f$, and*
- *for all $f \in \mathscr{C}(A, B)$, $g \in \mathscr{C}(B, C)$, and $h \in \mathscr{C}(C, D)$, $((h \circ g) \circ f) = (h \circ (g \circ f))$.*

We will also write $f : A \to B$ for $f \in \mathscr{C}(A, B)$.

The additional requirements for a directed graph to become a category are few indeed, and in many cases the definition of morphisms already entails an obvious and well-behaved composition operation. Yet the results one can derive from the components of a category are surprisingly rich, and usually all the essential knowledge about a class of objects is captured by some suitable category. The defining axioms of a category provide an abstract interface to all kinds of structures, and category theory allows for a unified treatment of all of them, since internal features of objects are disregarded completely.

As a simple example, consider the category Set of all sets and functions between them, i.e. $|\mathsf{Set}|$ consists of all sets, and given two sets $A, B \in |\mathsf{Set}|$ the collection $\mathsf{Set}(A, B)$ of all morphisms from A to B is just the collection of all functions from A to B. The identity morphisms are given as the identity functions, i.e. those functions which map all elements onto themselves. Composition in Set is given by composition of functions. Another category which we will discuss in more detail later is the category Poset of all partially ordered sets together with monotone functions. We remark that categorical morphisms are often given by functions, but that this is by no means necessary. For example, we can view a single partially ordered set as a category, where we have a single morphism between two elements p and q if and only if $p \leq q$. Composition is provided by transitivity and identity morphisms exist by reflexivity. This last example might appear somewhat peculiar at first, but it shows how general the basic notions in category theory really are.

Above, we gave two examples of specific relationships between objects: being a subobject and being equivalent. In the given examples, these do still refer to the internal structure of the objects, so we have to consider defining both in purely categorical terms. Let us first explore the notion of equivalence (or, speaking categorically, *isomorphism*) for the categories Set and Poset. Restating our earlier insights, we find that two partially ordered sets P and Q are equivalent (isomorphic) whenever there is a monotone function $f : P \to Q$ that has a monotone *inverse*, i.e. for which

[2] *Class* should be understood as a kind of *collection*. Classes of objects are not always *sets* of objects for reasons which have to do with Russell's paradox from set theory, but we shall not inconvenience us with such matters here. The term *class* is certainly not supposed to mean classes as e.g. in Description Logics!

there is a monotone function $g : Q \to P$ with $g \circ f = \mathrm{id}_P$ and $f \circ g = \mathrm{id}_Q$. Generalizing this to arbitrary categories, we call a morphism an *isomorphism* if it has a (necessarily unique) inverse morphism.

Application of this definition to Set reveals bijective functions as the isomorphisms of sets. Likewise, the categorical definition immediately provides us with a suitable notion of equivalence in any category we may wish to study. As mentioned in the introduction, the possible translations of information between ontologies are suggestive morphisms for ontology research. Indeed, no matter how ontologies and translations between them are implemented, composition of translations methods and the existence of identity translations should always be available. In consequence, isomorphic ontologies are intuitively described by the possibility of translating knowledge back and forth between them without loosing information. In a similar fashion, we will gain a general description of *ontology merging* later on, though it will be a bit more involved.

To obtain the notion of a subobject, let us consider the category Set and note that every subset $A \subseteq B$ of a given set B can be obtained as the image (range) of some injective (i.e. one-to-one) function into B. By injectivity, the domain of any such function is in bijective correspondence with the subset A. Thus any subset can be given by some injective function and any such function defines a subset[3]. We remark that the subobjects are really given by the injective function, not by its domain: for instance, a function from the one-element set $\{a\}$ to the natural numbers can map a to 0, to 1, or to any other number. In spite of the constant function domain, this always denotes different subsets ($\{0\}$, $\{1\}$, $\dots$) of the natural numbers. Yet, due to injectivity, the set $\{a\}$ is surely isomorphic (thus essentially equivalent) to the indicated subsets; still the *position* of the isomorphic copy of $\{a\}$ as a subobjects of the codomain does make a difference.

Now in order to obtain a categorical definition of injectivity, we observe that an injective function $f : A \to B$ in Set has the following peculiar property: for every pair of functions $g : C \to A$ and $h : C \to A$, the equality $f \circ g = f \circ h$ implies $g = h$. Morphisms (of arbitrary categories) with this feature are called *monomorphisms* and it turns out to be appropriate to consider a monomorphism as the specification of a subobject of its codomain. Intuitively, the condition describes a monomorphism $f : A \to B$ as an embedding of A into B that does not obliterate any essential features of A. Two morphisms $C \to A$ that are distinct on some part of A must also be distinct when extended via f to B. Thus the monomorphism can be thought of as a pointer to the specified subobject, which in turn is isomorphic to the domain of the monomorphism.

These examples give but a brief glimpse at the expressiveness of category theory. We continue next with discussing some constructions which will help in understanding pushouts.

Products and Relations

In set theory, the cartesian product of two sets is defined as the set of all pairs of elements from two given sets. This is not a suitable description from the viewpoint of category theory, since we want to avoid to mention the internal (element-based) structure of our objects. In order to rephrase this in categorical language, we need to find alternative criteria that rely exclusively on properties of the morphisms. To this end, an important observation is that a product does in general also provide two *projection functions* to the first respectively second component of the product. Furthermore, the product is distinguished by a *universal property* given in the next definition.

[3] Note, however, that there are usually many injective functions with the same image. So for obtaining an exact definition of subobjects, one would still have to identify the equivalent injective functions.

Definition 2 (Product) *Consider a category $\mathscr{C}$ and objects $A, B \in |\mathscr{C}|$. Given an object $C \in |\mathscr{C}|$ and morphisms $p_1 : C \to A$ and $p_2 : C \to B$, we say that (C, p_1, p_2) is the* product *of A and B if the following universal property holds:*

For any object $D \in |\mathscr{C}|$ and morphisms $q_1 : C \to A$ and $q_2 : C \to B$, there is a unique morphism $\langle q_1, q_2 \rangle : D \to C$, such that $q_1 = p_1 \circ \langle q_1, q_2 \rangle$ and $q_2 = p_2 \circ \langle q_1, q_2 \rangle$. The latter situation is depicted in the following diagram:

$$\begin{tikzcd}
D \arrow[dr, dashed, "{\langle q_1,q_2\rangle}"'] \arrow[drr, bend left, "q_2"] \arrow[ddr, bend right, "q_1"'] & & \\
& C \arrow[r, "p_2"'] \arrow[d, "p_1"] & B \\
& A &
\end{tikzcd}$$

For example, when considering the category Set and its usual cartesian product, we can define the function $\langle q_1, q_2 \rangle$ by setting $\langle q_1, q_2 \rangle(d) = (q_1(d), q_2(d))$. In spite of this, the above defines the cartesian product of sets only up to isomorphism (i.e. bijective correspondence) — *any* set with the cardinality of the cartesian product can be equipped with appropriate morphisms. This is a typical feature of category theory: isomorphic objects are not distinguished, since they behave similar in all practical situations. It is the choice of morphisms that determines what distinctions are considered relevant in the first place. Yet we will henceforth assume that we have fixed one representative for the product of any two elements A and B which we label $A \times B$. We also remark that products do not exist in every category, so the previous convention needs to be restricted to existing products.

We remark, nevertheless, that the notion of *product* of two objects depends solely on the chosen category, i.e. on the objects and their morphisms. Fixing, for example, a specific ontology language, and finding an agreement on which features of an ontology should be preserved by a corresponding morphism, we obtain a notion of *product* in a canonical way.

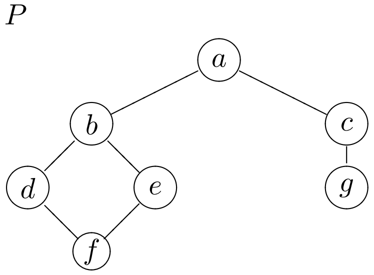
The categorical product definition also turns out to be suitable to model many well-known product constructions. As a product of two partially ordered sets one usually considers the product-order, i.e. the cartesian product of the two sets, ordered such that a pair (a, b) is below a pair (c, d) whenever a is below c and b is below d. The partially ordered set obtained in this way corresponds to the categorical product in Poset, which arguably is the reason for the significance of this particular construction. To give another example: If we consider a single partially ordered set as a category, as discussed earlier, then the product of two of its elements is just the greatest lower bound. This is also an example where a product may fail to exist.

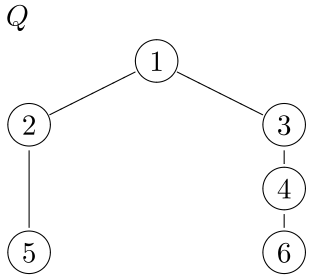
Combining the product construction with our earlier considerations on subobjects into practice, we can also introduce *binary relations* on objects. Indeed an ordinary set-theoretic binary relation is just a subset of the cartesian product of two objects. Hence it makes sense to consider a monomorphism $r : D \to (A \times B)$ from some object D to the product of A and B as a binary relation between A and B. Note that this does also give us two functions $p_1 \circ r : D \to A$ and $p_2 \circ r : D \to B$ to the two components of the product, for which the morphism r is already the unique factorization that exists due to the definition of a product. Much generalized theory can be developed around this, but we shall be content at this point.

4.3 Merging ontologies via pushouts

We will now return to our initial motivation. Our intuition is that the objects of our category represent ontologies and that the morphisms between them serve as meaningful transitions between these specifications. The categorical product construction is not suitable for the purpose of modelling ontology merging, since it does obviously not consider any relationship between two ontologies. Such a relationship — commonly referred to as an *ontology mapping* — however is the base of an ontology merging process, so we have to find a means of modelling it in our categorical setting. We are in fact more interested in a certain kind of *sum* than in a product. Indeed, if two ontologies were entirely unrelated, they could be combined by just taking their disjoint union (provided that this operation makes sense for the chosen ontology representation language). However, we are more interested in merging ontologies that do overlap (via some mapping), where some elements are related while others are not. Merging two such ontologies should lead to a new ontology that identifies equivalent elements but that tries to keep unrelated elements apart, as far as this is possible without violating the requirements that are imposed on the structure of an ontology.

As an example, let us consider the following two partial orders:

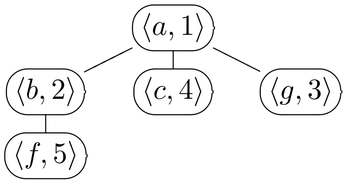




We assume that some elements of these structures are known to be equivalent. This is expressed by a relation $R \subseteq P \times Q$ (called an ontology alignment above) that we define as the set of pairs:

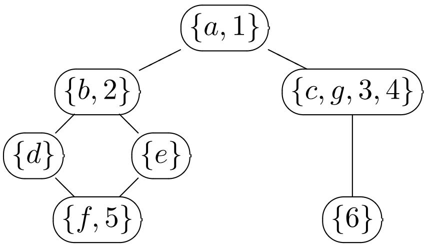
$$R = \{\langle a, 1\rangle, \langle b, 2\rangle, \langle c, 4\rangle, \langle f, 5\rangle, \langle g, 3\rangle\}.$$

This relation is not an object of the category, but it can easily be expressed as the poset $\alpha(P, Q)$:



and a pair of morphisms from $\alpha(P, Q)$ to P and Q mapping each element from $\alpha(P, Q)$ to the corresponding element in P and Q (here, the labels which have been ascribed to the elements are not meaningful, it is possible to erase them, the morphisms play the central role). It is clear that these morphisms preserve the order.

A reasonable result of merging the posets P and Q would then be the following structure:



Observe that all elements related by R are indeed identified, but that some additional identifications are necessary to obtain a partially ordered set. Categorically, we can already specify the data that we have considered for such an operation. The given situation is depicted in the following diagram:

$$\begin{tikzcd}
R \arrow[dr, tail] \arrow[drr, dashed, "r_2"] \arrow[ddr, dashed, "r_1"'] & & \\
& P \times Q \arrow[r] \arrow[d] & Q \\
& P &
\end{tikzcd}$$

The shape of the arrow from R to $P \times Q$ indicates that it defines a subobject (a monomorphism). The dotted arrows r_1 and r_2 are those that are obtained by composing the projections of the product with this monomorphism. They project every pair of elements of R to its first and second component, respectively. Now the result of merging P and Q is not just some poset $\mathsf{merge}_R(P,Q)$, but also the two obvious embeddings of P and Q into $\mathsf{merge}_R(P,Q)$. As a diagram, we obtain:

$$\begin{tikzcd}
& Q \arrow[d, "e_2"] \\
P \arrow[r, "e_1"'] & \mathsf{merge}_R(P,Q)
\end{tikzcd}$$

The property that R-related elements are identified can now be expressed in terms of functions: we find that, for any pair $(p,q) \in R$, $e_1(p) = e_2(q)$. Still a better way to express this for arbitrary morphisms is to say that $e_1 \circ r_1 = e_2 \circ r_2$.

This condition alone, however, does not suffice. Usually, there are many objects for which $e_1 \circ r_1 = e_2 \circ r_2$ holds. Which of these is the one which we want to consider as the *merging* of P and Q? Clearly, the merging shall not identify anything unnecessarily. This can be stated by means of another *universal property*, as follows.

Definition 3 (Pushout) *For a category $\mathscr{C}$, consider objects R, P, Q, and morphisms $p_1 : R \to P$ and $p_2 : R \to Q$. An object S together with two morphisms $e_1 : P \to S$ and $e_2 : Q \to S$ is a* pushout *if it satisfies the following properties:*

(i) $e_1 \circ p_1 = e_2 \circ p_2$*, i.e. the following diagram* commutes

$$\begin{tikzcd}
R \arrow[r, "p_2"] \arrow[d, "p_1"'] & Q \arrow[d, "e_2"] \\
P \arrow[r, "e_1"'] & S
\end{tikzcd}$$

(ii) *For every other object T and morphisms $f_1 : P \to T$ and $f_2 : Q \to T$, with $f_1 \circ p_1 = f_2 \circ p_2$, there is a unique morphism $m : S \to T$ such that $f_1 = e_1 \circ m$ and $f_2 = e_2 \circ m$. This situation is depicted in the following diagram.*

$$\begin{tikzcd}
R \arrow[r, "p_2"] \arrow[d, "p_1"'] & Q \arrow[d, "e_2"] \arrow[ddr, bend left, "f_2"] & \\
P \arrow[r, "e_1"'] \arrow[drr, bend right, "f_1"] & S \arrow[dr, dashed, "m"] & \\
 & & T
\end{tikzcd}$$

Condition (ii) in this definition states the universal property of the pushout, requiring that it is in a sense the most general object that meets all requirements. Let us try to explain this a bit further. We have already understood that in this setting we can encode the ontology mapping (e.g. binary relation) R conveniently, in that the resulting S identifies (at least) all those elements which are related by R. But now we want to *avoid* the identification of other elements as much as possible. Intuitively, this means that a suitable pushout object needs to keep elements from both components as distinct as possible, while still implementing all necessary identifications, and without including irrelevant information. Enforcing the desired identifications was achieved by condition (i) in the above definition. Excessive identifications are prevented by requiring the *existence* of a factorization m: appending m to e_1 and e_2 cannot make prior identifications undone, and hence a pair that was merged in S can never be separated in an alternative solution (T, f_1, f_2) if a suitable m is known to exist. Finally, the possibility of including entirely unrelated information, like adding some elements not present in either P or Q, is ruled out by assuring *uniqueness* of the factorization m: if S would include elements that are neither in the image of e_1 nor in the image of e_2 then a valid factorization can assign these to arbitrary values in T without loosing the factorization property — but this would result in many possible choices in place of m. In other words, having "unnecessary" elements in the S would result in additional degrees of freedom in the choice of m, thus violating the required uniqueness.

Note also that we ignored our earlier restriction of R being a subobject of the product $P \times Q$. However, by the universal property of the product, any object R with functions to P and Q must have a unique factorization through the product, and hence does still capture part of the idea of a relation. Furthermore, such a generalized R can also be viewed as a suitable background knowledge that both P and Q are based on. In spite of this generalization, R is still an object of the considered category, i.e. it is itself an ontology with all necessary structure. We just dropped some side conditions on this object, such that some redundancy can be introduced into the ontology mapping if desired.

4.4 How to put our approach into practice

Let us now see how our approach can be used as a guidance for ontology merging. We noted earlier that the notion of product hinges only on (1) a decision regarding the ontology representation language used, and (2) the structural properties which shall be preserved by a morphism. The situation for pushouts is similar, we just need a third decision, namely (3) the fixing of an ontology mapping. Once these decisions have been made, the notion of pushout, and thus of the merging of two ontologies, is determined canonically, but one still needs to find a convenient concrete representation for the specified object (4). From there, conceptually sound algorithms for calculating both ontology mappings (5) and pushouts (6) can be devised.

Decisions need to be made step by step, and we propose the following workflow. Later steps, however, may indicate that earlier decisions need to be revised, and thus to retrace to earlier points.

1. *Decide on ontology representation language used.* This first step is probably the most unproblematic, since there are standard ontology languages around, and the specific application case will usually dictate the language. Potential candidates are e.g. F-Logic [Kifer *et al.*, 1995; Angele and Lausen, 2004] and different variants of OWL [Antoniou and van Harmelen, 2004].

2. *Determine what suitable morphisms are.* This step consists of describing the conditions which morphisms must satisfy. These conditions will primarily be dictated by the semantic interpretation of the ontology representation language chosen earlier, and by the specific requirements of the application case. Typical conditions could include the following.

   - The preservation of class hierarchies, i.e. functions shall be monotonic with respect to the *general class inclusion* orders on classes and/or roles.
   - The preservation of types (e.g. classes, roles, annotated objects).
   - The taking into account of model-theoretic logical properties, if featured by the underlying ontology representation language, like satisfiability, or the preservation of specific models.
   - The taking into account of proof-theoretic properties, i.e. such relating to particular inference methods chosen for reasoning with ontologies.
   - The preservation of language classes, e.g. by requiring that the merging of two OWL Lite ontologies shall not result in an OWL ontology which is not in OWL Lite.

3. *Determine what the ontology mapping is for this setting.* Usually, ontology mappings will be given by (binary) relations between elements of ontologies, indicating which elements shall be identified in the merging process. However, as the product of two ontologies may not always be described conveniently as a set of pairs of elements — as in the case of Set or Poset —, it needs to be understood at this stage, what the product really is, and thus what ontology mappings are in this setting.

4. *Determine what pushouts are for this setting.* While the characteristics of a pushout are fully determined by the previous steps, it is still necessary to find a particular instance of the pushout (both for the object and the embedding morphisms) in terms of the ontology language. This requires to define a possible result for arbitrary pushout operations and to show that it satisfies the formal requirements of a pushout. Difficulties at this stage arise from

the fact that, like products, pushouts are not guaranteed to exist in general. Negative results may yield effective conditions for the existence of pushouts or even suggest a modification of the considered theory.

5. *Algorithmize how to obtain the mapping.* The issue of how to obtain suitable ontology mappings is a separate issue from the one discussed here, and will usually depend heavily on the application domain and on the ontology representation language chosen. Machine learning techniques may be used here together with linguistics-based approaches (see e.g. [Ehrig and Sure, 2004]). Fuzzy relations usually obtained by such approaches may however have to be defuzzified at some stage, in order to obtain a precise ontology mapping which will be used for the merging.

6. *Algorithmize how to obtain the pushout.* At this stage, it is theoretically clear what the pushout — and thus the merged ontology — will be. Casting this insight into an algorithm may require a considerable amount of work. The practitioner may also choose at this step to forego an exact implementation of the merging, and settle for an approximate or heuristic approach for reasons of efficiency, while at the same time being guided by the exact merging result as the ontology to be approximated.

4.5 Conclusion

We have argued that the problem of merging ontologies based on a given ontology mapping can be formulated conveniently in the language of category theory. This leads to the well-known definition of the categorical pushout construction, which describes ontological merging independently from the concrete implementation that was chosen. Since pushouts do not exist in all categories, this also yields general guidelines for devising systems of interrelated ontologies.

Moreover, the definition helps circumscribing what are exactly the alignments: indeed, an alignment is not exactly any kind of relation as presented above. Since it should join correctly the two ontologies, then, there should be a viable push-out from this relation.

Methods and insights from category theory could be used to assist in the development both of rigorous theoretical settings for ontology merging and of conceptually sound algorithms for practical implementations. In addition it yields a direct definition of the merge of two ontologies once an alignment is provided. Conversely, similar considerations can also be useful to validate alignment and merging constructions that have been conceived exclusively on practical grounds, since one may ask in which sense (in which category) a given merging process produces results of general validity.

More precise definitions of this requires the instanciation of objects and morphisms. This is provided in the framework of first order logic in the next section.

Chapter 5

Semantics for mappings

The previous chapter provided a very general view of what is to be found in mappings. However, it did not gave a precise semantics of the mapping that are expected from alignment and reconciliation processes. This chapter provides the semantics of these mappings.

The chapter begins with an example whose purpose is providing the reader with the intuition behind the introduced formalism. In the following sections syntax and semantics of the mappings are introduced. While in the last part of this chapter we briefly show that well known approaches to information integration fit into the described framework.

5.1 Motivating Example

Let's consider an example emphasising the difference between the rule-based semantics of integration and the classical semantics given to data integration systems, in the case of crisp mappings. Suppose we have three distributed information nodes. The first one (Σ_1) is the municipality's internal database, which has a binary table `Citizen-1` which contains the name of the citizen and the marital status (with values *single* or *married*). The second one (Σ_2) is a public database, obtained from the municipality's database, with two unary tables `Male-2` and `Female-2`. The third information node (Σ_3) is the Pension Agency database, obtained from a public database, with the unary table `Citizen-3` and a binary table `Marriage-3` (stating that two people are married). The three information nodes are interconnected by means of the following mappings:

$1:\texttt{Citizen-1}(x,y) \rightsquigarrow_\alpha 2:(\texttt{Male-2}(x) \vee \texttt{Female-2}(x))$
(this mapping connects Σ_1 with Σ_2)

$2:\texttt{Male-2}(x) \rightsquigarrow_\alpha 3:\texttt{Citizen-3}(x)$
$2:\texttt{Female-2}(x) \rightsquigarrow_\alpha 3:\texttt{Citizen-3}(x)$
(these mappings connect Σ_2 with Σ_3)

In the classical model, the `Citizen-3` table in Σ_3 should be filled with all of the individuals in the `Citizen-1` table in Σ_1, since the following mapping is logically implied:

$1:\texttt{Citizen-1}(x) \rightsquigarrow_\alpha 3:\texttt{Citizen-3}(x)$

However, in a rule-based integrated system – which can be compared to a peer-to-peer system – this is not a desirable conclusion. In fact, mappings should be interpreted only for fetching data,

and not for deduction. In this example, the tables `Female-2` and `Male-2` in Σ_2 will be empty, since the data is fetched from Σ_1, where the gender of any specific entry in `Citizen-1` is not known. From the perspective of Σ_2, the only thing that is known is that each citizen is in the view $(\texttt{Female-2} \vee \texttt{Male-2})$. Therefore, when Σ_3 asks for data from Σ_2, the result will be empty.
In other words, the mappings

$$2 : \texttt{Male-2}(x) \rightsquigarrow_\alpha 3 : \texttt{Citizen-3}(x)$$
$$2 : \texttt{Female-2}(x) \rightsquigarrow_\alpha 3 : \texttt{Citizen-3}(x)$$

will transfer no data from Σ_2 to Σ_3, since no individual is known in Σ_2 to be either definitely a male (in which case the first mapping would apply) or definitely a female (in which case the second mapping would apply). We only know that any citizen in Σ_1 is either male or female in Σ_2, and no reasoning about the mappings should be allowed.

Suppose now to have an additional cyclic pair of mappings connecting Σ_1 and Σ_3 as follows:

$$\begin{aligned} &1 : \texttt{Citizen-1}(x, \text{“married”}) \rightsquigarrow_\alpha 3 : \texttt{Marriage-3}(x, y) \\ &3 : \texttt{Marriage-3}(x, y) \rightsquigarrow_\alpha 1 : (\texttt{Citizen-1}(x, \text{“married”}) \ \wedge \\ &\qquad\qquad\qquad\qquad\qquad\quad \texttt{Citizen-1}(y, \text{“married”})) \end{aligned}$$

These cyclic mappings serve the purpose to *synchronise* the people who are known to be married from within the information node Σ_1 (by means of the `Citizen-1` table) with the people who are known to be married from within the information node Σ_3 (by means of the `Marriage-3` table).

Suppose that it is known in Σ_1 that only John is married, and nothing in known in Σ_3 about marriages. The cyclic mappings will propagate this information to Σ_3. However, there is still a subtle difference between the mappings interpreted in a classical way an the mappings interpreted as rules. In the classical model, a query to Σ_3 asking for the non existence of some married person different from John will get a negative answer. In a rule-based setting, we actually expect a positive answer, since the only information that is fetched is about John.

5.2 Syntax

Mappings are means to align knowledge among entities providing information. For this reason, the first concept to be introduced is a formal representation of these entities. These entities are called *information nodes*, and can be considered as first order theories on (possibly) distinct signatures.

However, the purpose of semantic alignment is to share knowledge among the nodes. Therefore, it is assumed a shared set of constant names which provides a sort of common vocabulary for the objects in the information system. One example of such shared constants are the URN in the world wide web.

Definition 4 (Information node) *Let I be a nonempty finite set of indexes $\{1, 2, \ldots, n\}$, and C be a set of constants. For each pair of distinct i, $j \in I$, let L_i be a first order function-free language with signature disjoint from L_j but for the shared constants C. An* information node Σ_i *is a theory on the first order language L_i.*

Given the starting blocks provided by the information nodes, the *mappings* are defined as relationships connecting formulae from different information nodes. As highlighted by the example, there are two different types of mappings according to the semantics of the integration among the nodes.

The mapping are distinguished into *classical* and *rule-based*. As their names suggest, their semantics differ in order to take into account the main two approaches in information integration. Only the syntax is described in this section, the formal semantics is introduced in the following section.

Definition 5 (Mapping) *A* mapping *is an expression of either of the form:*

$$\begin{array}{ll} i : \phi(\mathbf{x}) \Rightarrow_{\alpha} j : \psi(\mathbf{x}) & \textit{(classical mapping)} \\ i : \phi(\mathbf{x}) \rightsquigarrow_{\alpha} j : \psi(\mathbf{x}) & \textit{(rule-based mapping)} \end{array}$$

where i, j are distinct indices, $\alpha \in [\bot, \top]$ is a degree of confidence and ϕ is an open formula of L_i, and $\psi(\mathbf{x})$ is an open formula of L_j, both with free variables $\mathbf{x} = \{x_1, \ldots, x_\ell\}$. A crisp mapping *is a mapping with a degree of confidence $\alpha = \top$.*

In addition to generic formula mappings, a special kind of mappings is included to allows the alignment of arbitrary constants.

Definition 6 (Constant Mapping) *A* constant mapping *is an expression of the form $c_i \mapsto_{\alpha} c_j$ where i, j are distinct indices, $\alpha \in [\bot, \top]$ is a degree of confidence, c_i is a constant of Σ_i, c_j is a constant of Σ_j.*

To any mapping is associated a degree of confidence to allow the representation of uncertainty into the framework. This aspect is described in Section 5.4.

An integrated system is defined as the information nodes themselves, together with the mappings connecting them.

Definition 7 (Integrated system) *An* integrated system *is composed by a set MDB of information nodes and a set MAP of mappings.*

The purpose of an information system is to provide knowledge; therefore querying is an essential aspect of this framework. A query is always considered w.r.t. a given node, the alignment provides the mechanism in which different nodes can contribute to the answer of a given query. For this reason, queries are defined as formulae written with a language from an (arbitrary) single node.

Queries can have free variables, and in this case they retrieve set of tuples corresponding to the variables. When queries have no free variables, they are called boolean, since they can be either true or false.

Definition 8 (Query) *A* query *is a (possibly open) first order formula in the language of one of the information node Σ_i.*

5.3 Semantics of crisp mappings

In order to simplify the exposition in this section only crisp mappings are considered. The following section will take into account the degree of confidence as well.

To describe the semantics of the integrated system, the semantics of each node must be accounted for. Then the interconnection among the single nodes are considered, together with the restrictions imposed by the mappings.

It has been assumed that each node is a first order theory, therefore interpretations for each node are given in terms of a domain and first order interpretations. An interpretation for the integrated system consists in the set of interpretations for the single nodes. However, this is not sufficient because the node interpretation are not required to share the same domain.

Domains of different node interpretations are related by means of *match* relations, mapping elements of a domain to elements of the domain of a different node.

Definition 9 (Interpretation) *Let $\langle MDB, MAP\rangle$ be an integrated system with crisp mappings only, and for each information node Σ_i let Δ_i be a non empty set of objects. For each pair of distinct indices in MDB, we define a* match *relation $R_{i,j} \subseteq \Delta_i \times \Delta_j$.*

An integrated interpretation *for the integrated system is a collection of information node models $\mathbf{m} = \{m_1, m_2, \dots m_n\}$. For each information node Σ_i in MDB, an information node model m_i is a first order model of Σ_i on the domain Δ_i that interpret constants in C as themselves; i.e.,*

$$m_i \models \Sigma_i.$$

Models of an integration system are the interpretations which satisfy the mappings among the nodes.

Definition 10 (Models) *Let's define an assignment α_i for the information node Σ_i in the usual way as a function from variable symbols in L_i to elements in Δ_i. In addition, we restrict the assignments to satisfy the match relations, i.e., $R_{i,j}(\alpha_i(x), \alpha_j(x))$ for each variable symbol x and each pair of distinct indices i, j in the integrated system.*

A model $\mathbf{M}$ *of an integrated system – written $\mathbf{M} \models \langle MDB, MAP\rangle$ – is a nonempty set of integrated interpretations satisfying every mapping, i.e, for each pair of assignments α_i and α_j the following holds:*

- *if the mapping is classical – $(i : \phi(\mathbf{x}) \Rightarrow_\alpha j : \psi(\mathbf{x}))$ – then*

$$\forall \mathbf{m} \in \mathbf{M}.\, ((\mathbf{m}|_i, \alpha_i \models \phi(\mathbf{x})) \rightarrow (\mathbf{m}|_j, \alpha_j \models \psi(\mathbf{x})))$$

- *if the mapping is rule-based – $(i : \phi(\mathbf{x}) \rightsquigarrow_\alpha j : \psi(\mathbf{x}))$ – then*

$$(\forall \mathbf{m} \in \mathbf{M}.(\mathbf{m}|_i, \alpha_i \models \phi(\mathbf{x}))) \rightarrow (\forall \mathbf{m} \in \mathbf{M}.(\mathbf{m}|_j, \alpha_j \models \psi(\mathbf{x})))$$

- *if the mapping is between constants – $c_i \mapsto_\alpha c_j$ – then*

$$\forall \mathbf{m} \in \mathbf{M}.\, ((\mathbf{m}|_i, \alpha_i \models (x = c_i)) \rightarrow (\mathbf{m}|_j, \alpha_j \models (x = c_j)))$$

where we intend $\mathbf{m}|_i$ to be the element m_i of $\mathbf{m}$.

Although the mappings are restricted to three kinds, the freedom in their combination allows to represent a variety of commonly used mappings. In fact, even the constant mapping can be represented by means of a classical mapping; as shown in the semantics above.

Among the widely used mappings, two common examples are equivalence and disjointness. The first one stating the equivalence between two formulae, and the second their disjointness. Given the nature of these two constraints they are better represented by means of classical mappings.

An equivalence mapping can be represented by means of two symmetric mappings. For example, to say that $Car(\cdot)$ in the node 1 is equivalent to $Voiture(\cdot)$ in node 2, the following two mappings can be used:

$$1 : Car(x) \Rightarrow_\alpha 2 : Voiture(x)$$
$$2 : Voiture(x) \Rightarrow_\alpha 1 : Car(x)$$

Disjointness mappings can be represented using negation in one of the formulae. For example, to say that two nodes, although they use the same predicate $Person(\cdot)$, contain informations about two different group of people the following mapping can be employed:

$$1 : Person(x) \Rightarrow_\alpha 2 : \neg Person(x)$$

Complex mappings can be represented using rule-based mappings as well; but the nature of their semantics makes their combination less intuitive.

Semantics for the queries is provided in the usual way, by means of the models of an integrated system. Note that answers to a query are given in terms of the shared constants.

Definition 11 (Query answer) *Let $Q_i(\mathbf{x})$ be a query with free variables $\mathbf{x}$ (possibly empty). The* answer set *of Q_i is the set of substitutions of $\mathbf{x}$ with constants $\mathbf{c}$, such that any model $\mathbf{M}$ of an integrated system satisfies the query, i.e.,*

$$\{\mathbf{c} \in C \times \cdots \times C \mid \forall \mathbf{M}.\ (\mathbf{M} \models \langle \mathit{MDB}, \mathit{MAP} \rangle) \rightarrow \forall \mathbf{m} \in \mathbf{M}.\ (m_i \models Q_i(\mathbf{c}))\}$$

5.4 Semantics of fuzzy mappings

In real-life applications, the conceptualisation of the specific domain may result to a represented knowledge that has deficiencies. In general, the information represented can be imprecise, incomplete, vague, fragmentary, contradictory, random, etc. Moreover, the mappings between different information nodes are also caused by uncertainty.

Modelling this can take advantage of relationships between formulas which are not crisp (like $\Longrightarrow$), but have an α component which is different from $\top$ and $\bot$. We present here the semantics of the fuzzy one. In the framework presented here, we try to face this uncertainty, by using degrees (between 0 and 1) that represent the confidence of a specific hypothesis. Three types of uncertainty are introduced in the knowledge representation and alignment process:

- Fuzzy interpretations, i.e. a degree of membership to each interpretation (different semantics for the constructors of the representation language).
- Fuzzy mappings, i.e. a degree of confidence associated to each mapping.
- Fuzzy alignment, i.e. a degree of trust associated to the alignment system.

In this section, we concentrate on the first and the second types of uncertainty. We assume that we have mappings (crisp or not) between information nodes constructed with the aid of a fuzzy extension of its representation language. This means that the syntactic constructors of the language have different semantics based on the notion of a fuzzy interpretation. It is important to notice that all the above constructors should satisfy some minimal requirements that ensure the validity of the extension. In Deliverable 2.5.1 (Specification of Coordination of Rule and Ontology Languages), and more specifically in Section 5 (A Fuzzy Extension), a fuzzy DL extension is presented and the above requirements are summarised in the definition of *valid fuzzy assertional extensions*. The use of the extended language L_i (with extended semantics) results to formulas that have truth values between 0 and 1 (and not only 0 or 1), under a fuzzy model m_i of Σ_i (on the domain Δ_i) and an assignment α.

Definition 12 (Semantics of fuzzy mappings) *Let $\langle MDB, MAP \rangle$ be an integrated system with information nodes and mappings that are crisp or not. Let also R be a fuzzy match relation $R_{i,j} : \Delta_i \times \Delta_j \to [0,1]$.*
An integrated interpretation *for the integrated system is a collection of fuzzy models $\mathbf{m} = \{m_1, m_2, \ldots m_n\}$, where m_i is a fuzzy model of Σ_i.*
A model $\mathbf{M}$ *of an integrated system is a nonempty set of integrated interpretations satisfying every mapping, i.e, for each pair of assignments α_i and α_j that satisfy $R_{i,j}(\alpha_i(x), \alpha_j(x))$ (i.e., $R_{i,j}(\alpha_i(x), \alpha_j(x)) > 0$) the following holds:*

- *if the mapping is classical – $(i : \phi(\mathbf{x}) \Rightarrow_\alpha j : \psi(\mathbf{x}))$ – then*

$$\inf_{\mathbf{m} \in \mathbf{M}} \omega_t((\mathbf{m}|_i, \alpha_i \models \phi(\mathbf{x})), (\mathbf{m}|_j, \alpha_j \models \psi(\mathbf{x}))) \geqslant \alpha$$

- *if the mapping is rule-based – $(i : \phi(\mathbf{x}) \leadsto_\alpha j : \psi(\mathbf{x}))$ – then*

$$\omega_t[\inf_{\mathbf{m} \in \mathbf{M}} (\mathbf{m}|_i, \alpha_i \models \phi(\mathbf{x})), \inf_{\mathbf{m} \in \mathbf{M}} (\mathbf{m}|_j, \alpha_j \models \psi(\mathbf{x}))] \geqslant \alpha$$

where ω_t is a fuzzy implication, t is a triangular norm.

5.5 Comparison with other approaches

Classical logic-based Information Integration If we consider an integrated system where there is a unique common domain Δ for each information node, the match relation is the identity relation over Δ, and only classical mappings are present, then the logical framework exactly characterises (and generalises) the classical logic-based information integration approach [Franconi *et al.*, 2001; Catarci and Lenzerini, 1993; Calvanese *et al.*, 1998b; Jarke *et al.*, 1999; 2000; Calvanese *et al.*, 2002; Peim *et al.*, 2004].

Consider, as an example, the case of multiple databases to be integrated. Each database have its own conceptual schema and logical schema, where the logical schema can be seen a set of views over the conceptual schema (local-as-view approach). We assume that each symbol of each schema is identified by a unique global symbol; i.e., the various databases have disjoint signatures. Interdependencies between entities and relationships in different schemas are represented by means of integrity constraints involving symbols of the schemas. Such interdependencies are called *inter-model assertions*. The union of the various schemas with the inter-model assertions and the local views forms the global integrated schema, or the *mediator*. It is worth noting that the integration process is incremental – since the integrated schema can be monotonically refined as soon as there is new understanding of the different component schemas – and that the resulting unified schema is strongly dependent from (actually, it includes) the schemas of the single information sources.

This approach gives both a clear semantics to the integration process of ontologies, and a calculus for deriving inconsistencies and checking the validity of integrity constraints in the integrated schema. Most importantly, in this framework global queries can be defined as views over single ontologies, or they can be generalised to span over multiple ontologies. The view-based query processing mechanism will guarantee the correct answer to the global query from the local sources [Calì *et al.*, 2004].

In [Lenzerini, 2002] a comparison is given between the above local-as-view approach to processing global queries and the global-as-view approach, which is more common in current information integration architectures.

Only recently has knowledge representation research started to have an interest in query processing and information access. Recent work has come up with advanced reasoning techniques for query evaluation and rewriting using views under the constraints given by the ontology – also called view-based query processing [Ullman, 1997; Calvanese *et al.*, 2000b]. This means that the notion of accessing information through the navigation of an Ontology modelling the document's domain – which can be seen as a conceptual schema – has its formal foundations.

Two approaches to view-based query processing exist, namely query rewriting (see, e.g., [Beeri *et al.*, 1997]) and query answering (see, e.g., [Abiteboul and Duschka, 1998; Calvanese *et al.*, 2000a; Peim *et al.*, 2002]). In the former approach, we are given a query Q, a set of view definitions characterising the actual data, and a set of (conceptual) constraints – all over the conceptual vocabulary – and the goal is to reformulate the query into an expression, the rewriting, that refers only to the views, and provides the answer to Q. Typically, the rewriting is formulated in the same language used for the query and the views. In the latter approach, besides Q, the view definitions and the constraints, we are also given the extensions of the (materialised) views. The goal is to compute the set of tuples that are implied by these extensions, i.e., the set of tuples that are in the answer set of Q in all the databases that are consistent with the views and the constraints.

In both cases, view definitions can be characterised in the framework presented in this document. In fact, the mappings are general enough to be used to define queries over the different databases. Analysing the techniques for answering these queries is outside the scope of this document; however, with opportune restrictions the techniques presented in literature can be used in this framework.

Rule-based Information Integration If we consider an integrated system where there is a unique common domain Δ for each information node, the match relation is the identity relation over Δ, and only rule-based mappings are present, then the logical framework exactly characterises (and generalises) the peer-to-peer logic-based information integration approach. If we push further, by allowing arbitrary distinct domains for the information nodes as well as a general match relation, then the logical framework characterises the context based approach. In the following, we will briefly show how the most relevant approaches in the literature are actually within our proposed logical framework.

The autoepistemic approach, which is the basis for the rule-based semantics, was first introduced by [Donini *et al.*, 1998], with the goal of formalising the *constraint rules* implemented in many practical knowledge representation systems. These rules are also the basis of the recent formalisations of peer-to-peer systems [Franconi *et al.*, 2003a]. As shown in [Franconi *et al.*, 2003a], the autoepistemic semantics as defined above is equivalent to the context-based semantics of [Giunchiglia, 1993; Ghidini and Serafini, 1998; Ghidini and Giunchiglia, 2001], and to the use of the autoepistemic operator, as defined, e.g., in [Reiter, 1992].

The framework presented in this document shares the same spirit of the Piazza system [Halevy *et al.*, 2003; Tatarinov and Halevy, 2004]. The vision of the Piazza peer data management system (PDMS) project is to provide semantic mediation between an environment of peers, each with its own schema. Rather than requiring the use of a single, uniform, centralised mediated schema to share data between peers, Piazza allows peers to define semantic mappings between pairs of peers (or among small subsets of peers). In turn, transitive relationships among the schemas of the peers are exploited so the entire resources of the PDMS can be used. The Piazza system is limited in the fact that it does not allow full GLAV mapping rules (i.e., heads must be atomic queries), it does not allow for cyclic mapping rules, and it does not allow for dynamic networks.

In the field of PDMS as defined above – which includes [Bernstein *et al.*, 2002; Serafini *et al.*, 2003; Halevy *et al.*, 2003; Tatarinov and Halevy, 2004; Calvanese *et al.*, 2003; 2004; Fagin *et al.*, 2003] – there are only two other approaches which deal in a well founded way with cycles in the mapping rules [Serafini and Ghidini, 2000; Calvanese *et al.*, 2003]. The acyclic case is relatively simple – a query is propagated through the network until it reaches the leaves of the network. The work in [Calvanese *et al.*, 2003] uses a notion of semantics similar to the semantics introduced in [Franconi *et al.*, 2003b], but it describes a partially distributed algorithm, that assumes that nodes may exchange mappings and data, so that a unique node will eventually evaluate in one shot the query answer – there is no distributed computation and the network may be flooded with data. The paper [Serafini and Ghidini, 2000] describes a local algorithm to compute query answers, but it does not allow real GLAV mapping rules (with existential variables in the head).

Chapter 6

Ontology alignment process

Mappings are the basic building blocks of ontology alignment. The goal of this section is to provide a precise definition of what the alignment process is in general, and what are its main dimensions.

Determining these dimensions is very important for characterizing what known or yet to be invented alignment algorithm does and then in which situation it is adapted. It should also be very useful in designing benchmark tests and comparing similar algorithms.

6.1 Characterization of the alignment process

The alignment process simply consists of generating an alignment (A') from a pair of ontologies (o and o'). However, there are various other parameters which can extend the definition of the alignment process. These are namely, the use of an input alignment (A) which is to be completed by the process, the alignment methods parameters (which can be weigths for instance) and some external resources used by the alignment process (which can be general-purpose resources not made for the case under consideration, e.g., lexicons, databases). This process can be defined as follow:

Definition 13 (Alignment process) *The alignment process can be seen as a function f which, from a pair of ontologies o and o' to align, an input alignment A, a set of parameters p, a set oracles and resources r, returns a new alignment A' between these ontologies:*

$$A' = f(o, o', A, p, r)$$

This can be represented as in Figure 6.1.

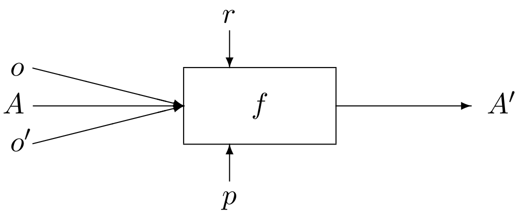


Figure 6.1: The alignment process.

Moreover, it can be useful to specifically consider the alignment of many ontologies within the same process. We call this multi-alignment.

Definition 14 (multi-alignment process) *The multi-alignment process can be seen as a function f which, from a set of ontologies to align $\{o_1, \dots o_n\}$, an input multi-alignment A, a set of parameters p, a set oracles and resources r, returns a new alignment A' between these ontologies:*

$$A' = f(o_1, \dots o_n, A, p, r)$$

6.2 Dimensions of an alignment process

Beside the general scheme presented above, there are many restrictions that can be put on this alignment process. These restrictions are useful for either constraining the alignment algorithm to deliver a particular kind of alignment (e.g., 1-1 preserving consequences) or to choose an algorithm adapted to the required constraints.

These dimensions affect most of the components of the above alignment definition:

Input ontologies (o, o') Input ontologies can be applied various constraints:

heterogeneity of the input languages: are they described in the same knowledge representation languages? This corresponds to asking for the non emptyness of the syntactic component of the resulting alignment.

languages: what are the languages of the ontologies (especially in case of homogeneous languages)? Example of languages are KIF, OWL, RDFS, UML, F-Logic, etc.

number: is this an alignment or a multi-alignment?

Input alignment (A) The input alignment:

complete/update: Is the alignment process required to complete an existing alignment? (i.e., is A non empty).

multiplicity : How many entities of one ontology can correspond to one entity of the others? Usual notations are 1:1, 1:m, n:1 or n:m. We prefer to note if the mapping is injective, surjective and total or partial on both side. We then end up with more alignment arities (noted with, 1 for injective and total, ? for injective, + for total and * for none and each sign concerning one mapping and its converse): ?:?, ?:1, 1:?, 1:1, ?:+, +:?, 1:+, +:1, +:+, ?:*, *:?, 1:*, *:1, +:*, *:+, *:*. These assertions could be provided as input (or constraint) for the alignment algorithm or be provided as a result by the same algorithm.

Parameters (p, r)

oracles/resources Are oracle authorized? If so, which ones (the answer can be any)? Is human input authorized?

training Can training be performed on a sample?

proper parameters Are some parameter necessary? And what are they? This point is quite important when a method is very sensitive the variation of parameters. A good tuning of these must be available.

Output alignment (A')

- **multiplicity** The multiplicity of the output alignment is similar to that of the input alignment (see above).
- **justification** Is a justification of the results provided?
- **relations** Should the relations involved in the correspondences be only equivalence relations or could they be more complex?
- **strictness** Can the result be expressed with trust-degrees different than $\top$ and $\bot$ or should they be strictified before?

Alignment process (f) The alignment process itself can be constrained:

- **resource constraints** Is there a maximal amount of time or space available for computing the alignment?
- **Language restrictions** Is the mapping scope limited to some kind of entities (e.g., only T-box, only classes)?
- **Property** Must some property be true of the alignment? For instance, one might want that the alignment (as defined in the previous chapter be a conseqeunce of the combination of the ontologies (i.e., $o, o' \models A'$) or that alignments preserve consequences (e.g., $\forall \phi, \phi' \in L, \phi \models \phi' \implies A'(\phi) \models A'(\phi')$) or that the initial alignment is preserved (i.e., $o, o', A' \models A$).

The purpose of the dimensions is the definition of the parameters and characteristics of expected behavior in benchmark and the comparison of algorithms and systems in deliverable D2.2.3.

6.3 Data alignment and integration

Data alignment and integration consists in merging data (and sometimes data streams, d and d') expressed in different ontologies (o and o'). For that purpose, the ontologies have to be aligned beforehand and the data integration can use this alignment. This is an example of combined off-line and on-line alignment.

It can be thought of as:

1. a first ontology alignment phase (f), possibly with an instance training set,
2. a data alignment phase (f') using the first alignment (A').

This is presented in Figure 6.2.

In this setting, the second phase benefits from the precompiling of the first alignment. Indeed, the second alignment process f' can be thought of as a compilation of the first alignment. This covers enough applications to deserve a separate threatment (e.g., for benchmarking).

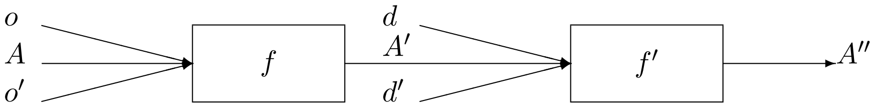


Figure 6.2: Data integration as another alignment process.

Conclusion

This document provided very general characterisation of the nature of alignments. They have first be considered under conceptual facets of the context in which alignments are needed (and thus what they align). Then a categorical framework has characterised alignments as the seed for the merge operation (itself defined as a push-out). Furthermore, global and distributed model-theoretic semantics have been given to the mappings composing alignments. Finally, the alignment process, by which alignments are produced has been proposed a definition covering most of the actual systems.

As could be expected from such a broad topic, the provided definitions are very abstract. We feel that they cover most of what have been implemented as alignment systems. Some of the definitions presented here provide however a very acute and fruitful view of what alignment are.

This work is very useful for designing tools and alignment algorithms and examining their properties. It will be used within the network for proposing a common alignment format that can be exchanged among a variety of tools (aligners, merger, transformers, etc.). It has been used as well for designing the alignment benchmarks of Deliverable 2.2.2.

These characterisations raise very interesting questions such as: is it possible to further characterise alignments or most specific alignments in the categorical framework? Can model theory account for more complex structure in the relations between distributed sites (for instance, by modelling the effort required by a site to acquire knowledge)? These questions are worth investigating for grounding further the work on ontology reconciliation in Knowledge web. We will devote some resources toward this.

Bibliography

[Abiteboul and Duschka, 1998] S. Abiteboul and O. Duschka. Complexity of answering queries using materialised views. In *Proc. of the 17th ACM Symp. on Principles of Database Systems (PODS'98)*, pages 254–265, 1998.

[Angele and Lausen, 2004] J. Angele and G. Lausen. Ontologies in F-logic. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, pages 29–50. Springer, 2004.

[Antoniou and van Harmelen, 2004] G. Antoniou and F. van Harmelen. Web Ontology Language: OWL. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies*, pages 67–92. Springer, 2004.

[Bechhofer *et al.*, 2001] Sean Bechhofer, Carole Goble, and Ian Horrocks. DAML+OIL is not enough. In Isabel Cruz, Stefan Decker, Jérôme Euzenat, and Deborah McGuinness, editors, *Proc. Semantic web working symposium, Stanford (CA US)*, pages 151–159, 2001.

[Beeri *et al.*, 1997] C. Beeri, A. Y. Levy, and M.-C. Rousset. Rewriting queries using views in description logics. In *Proc. of the 16th ACM Symp. on Principles of Database Systems (PODS'97)*, pages 99–108, 1997.

[Benerecetti *et al.*, 2000] M. Benerecetti, Paulo Bouquet, and Chiara Ghidini. Contextual reasoning distilled. *Journal of Theoretical and Experimental Artificial Intelligence*, 12(3):279–305, July 2000.

[Bernstein *et al.*, 2002] P. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. Data management for peer-to-peer computing: A vision. *Workshop on the Web and Databases, WebDB*, 2002.

[Bouquet *et al.*, 2003] Paolo Bouquet, Bernardo Magnini, Luciano Serafini, and Stefano Zanobini. A SAT-based algorithm for context matching. In Patrick Blackburn, C. Ghidini, Raymond Turner, and Fausto Giunchiglia, editors, *Proc. 4th International and Interdisciplinary COnference on Modeling and Using Context (CONTEXT-03)*, volume 2680 of *Lecture Notes in Artificial Intelligence*, pages 66–79. Springer Verlag, 2003.

[Calì *et al.*, 2004] Andrea Calì, Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Data integration under integrity constraints. *Information Systems*, 29(2):147–163, 2004.

[Calvanese *et al.*, 1998a] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. On the decidability of query containment under constraints. In *Proc. of the 17th ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS'98)*, pages 149–158, 1998.

[Calvanese *et al.*, 1998b] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, Daniele Nardi, and Riccardo Rosati. Information integration: Conceptual modeling and reasoning support. In *Proc. of the 6th Int. Conf. on Cooperative Information Systems (CoopIS'98)*, pages 280–291, 1998.

[Calvanese *et al.*, 2000a] D. Calvanese, G. De Giacomo, and M. Lenzerini. Answering queries using views over description logics knowledge bases. In *Proc. of the 16th Nat. Conf. on Artificial Intelligence (AAAI 2000)*, 2000.

[Calvanese *et al.*, 2000b] D. Calvanese, G. De Giacomo, M. Lenzerini, and Moshe Y. Vardi. View-based query processing and constraint satisfaction. In *Proc. of the 15th IEEE Sym. on Logic in Computer Science (LICS 2000)*, 2000.

[Calvanese *et al.*, 2002] Diego Calvanese, Giuseppe De Giacomo, and Maurizio Lenzerini. Description logics for information integration. In A. Kakas and F. Sadri, editors, *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski*, volume 2408 of *Lecture Notes in Computer Science*, pages 41–60. Springer, 2002.

[Calvanese *et al.*, 2003] Diego Calvanese, Elio Damaggio, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Semantic data integration in p2p systems. In *Proc. of the VLDB International Workshop On Databases, Information Systems and Peer-to-Peer Computing (DBISP2P-2003)*, 2003.

[Calvanese *et al.*, 2004] Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Logical foundations of peer-to-peer data integration. In *Proc. of the 23rd ACM SIGACT SIGMOD SIGART Sym. on Principles of Database Systems (PODS-2004)*, 2004. To appear.

[Catarci and Lenzerini, 1993] T. Catarci and M. Lenzerini. Representing and Using Interschema Knowledge in Cooperative Information Systems. *Journal of Intelligent and Cooperative Systems*, 2(4):375–398, 1993.

[Corcho, 2004] Oscar Corcho. *A declarative approach to ontology translation with knowledge preservation*. PhD thesis, Universidad Politécnica de Madrid, 2004.

[Donini *et al.*, 1998] F. M. Donini, M. Lenzerini, D. Nardi, W. Nutt, and A. Schaerf. An epistemic operator for description logics. *Artificial Intelligence*, 100(1-2):225–274, April 1998.

[Ehrig and Sure, 2004] M. Ehrig and Y. Sure. Ontology mapping — an integrated approach. In Christoph Bussler, John Davis, Dieter Fensel, and Rudi Studer, editors, *Proceedings of the First European Semantic Web Symposium*, volume 3053 of *Lecture Notes in Computer Science*, pages 76–91, Heraklion, Greece, 2004. Springer Verlag.

[Euzenat, 2000] Jérôme Euzenat. Towards formal knowledge intelligibility at the semiotic level. In *Proc. ECAI workshop on applied semiotics: control problems, Berlin (DE)*, pages 59–61, 2000.

[Euzenat, 2001] Jérôme Euzenat. Towards a principled approach to semantic interoperability. In Asunción Gómez Pérez, Michael Gruninger, Heiner Stuckenschmidt, and Michael Uschold, editors, *Proc. IJCAI 2001 workshop on ontology and information sharing, Seattle (WA US)*, pages 19–25, 2001.

[Fagin *et al.*, 2003] Ronald Fagin, Phokion G. Kolaitis, R. J. Miller, and Lucian Popa. Data exchange: Semantics and query answering. In *Proceedings of the 9th International Conference on Database Theory*, pages 207–224. Springer-Verlag, 2003.

[Franconi *et al.*, 2001] Enrico Franconi, Ken Barker, and Diego Calvanese, editors. *Proceedings of the International Workshop on Foundations of Models for Information Integration (FMII-2001)*. Springer-Verlag, 2001.

[Franconi *et al.*, 2003a] Enrico Franconi, Gabriel Kuper, A. Lopatenko, and L. Serafini. A robust logical and computational characterisation of peer-to-peer database systems. In *International VLDB Workshop On Databases, Information Systems and Peer-to-Peer Computing (DBISP2P'03)*, 2003.

[Franconi *et al.*, 2003b] Enrico Franconi, Gabriel Kuper, Andrei Lopatenko, and Luciano Serafini. A robust logical and computational characterisation of peer-to-peer database systems. In *Proceedings of the VLDB International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P'03)*, 2003.

[Ghidini and Giunchiglia, 2001] Chiara Ghidini and Fausto Giunchiglia. Local Models Semantics, or Contextual Reasoning = Locality + Compatibility. *Artificial Intelligence*, 127(2):221–259, April 2001.

[Ghidini and Giunchiglia, 2004] Chiara Ghidini and Fausto Giunchiglia. A semantics for abstraction. In *Proc. ECAI*, 2004.

[Ghidini and Serafini, 1998] Chiara Ghidini and Luciano Serafini. Distributed first order logics. In Franz Baader and Klaus Ulrich Schulz, editors, *Frontiers of Combining Systems 2*, Berlin, 1998. Research Studies Press.

[Giunchiglia and Walsh, 1992] Fausto Giunchiglia and Toby Walsh. A Theory of Abstraction. *Artificial Intelligence*, 57(2-3):323–390, 1992. Also IRST-Technical Report 9001-14, IRST, Trento, Italy.

[Giunchiglia, 1993] Fausto Giunchiglia. Contextual reasoning. *Epistemologia, special issue on I Linguaggi e le Macchine*, XVI:345–364, 1993. Short version in Proceedings IJCAI'93 Workshop on Using Knowledge in its Context, Chambery, France, 1993, pp. 39–49. Also IRST-Technical Report 9211-20, IRST, Trento, Italy.

[Goguen, 1991] J. Goguen. A categorical manifesto. *Mathematical Structures in Computer Science*, 1:49–67, 1991.

[Goguen, 1999] Joseph Goguen. An introduction to algebraic semiotics, with applications to user interface design. In Chrystopher Nehaniv, editor, *Computation for Metaphor, Analogy and Agents*, volume 1562 of *Lecture Notes in Artificial Intelligence*, pages 242–291. Springer Verlag, Berlin (DE), 1999.

[Halevy *et al.*, 2003] Alon Halevy, Zachary Ives, Dan Suciu, and Igor Tatarinov. Schema mediation in peer data management systems. In *Proceedings of the 19th International Conference on Data Engineering (ICDE'03)*, 2003.

[Hameed *et al.*, 2004] Adil Hameed, Alun Preece, and Derek Sleeman. Ontology reconciliation. In Steffen Staab and Rudi Studer, editors, *Handbook of ontologies*, International handbooks on information systems, chapter 12, pages 231–250. Springer Verlag, Berlin (DE), 2004.

[Jannink *et al.*, 1998] J. Jannink, S. Pichai, D. Verheijen, and G. Wiederhold. Encapsulation and composition of ontologies. In *Proceedings of the AAAI Workshop on AI & Information Integration*, 1998.

[Jarke *et al.*, 1999] M. Jarke, M. Lenzerini, Y. Vassilious, and P. Vassiliadis, editors. *Fundamentals of Data Warehousing*. Springer-Verlag, 1999.

[Jarke *et al.*, 2000] Mathias Jarke, V. Quix, D. Calvanese, Maurizio Lenzerini, Enrico Franconi, S. Ligoudistiano, P. Vassiliadis, and Yannis Vassiliou. Concept based design of data warehouses: The DWQ demonstrators. In *2000 ACM SIGMOD International Conference on Management of Data*, May 2000.

[Kent, 2000] R. Kent. The information flow foundation for conceptual knowledge organization. In *Proceedings of the Sixth International Conference of the International Society for Knowledge Organization*, 2000.

[Kifer *et al.*, 1995] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 42, 1995.

[Klein, 2001] Michel Klein. Combining and relating ontologies: an analysis of problems and solutions. In Asuncion Gomez-Perez, Michael Gruniger, Heiner Stuckenschmidt, and Mike Uschold, editors, *Proc. IJCAI workshop on ontologies and information sharing, Seattle (WA US)*, 2001.

[Lenzerini, 2002] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proc. of PODS-2002*, pages 233–246, 2002.

[Peim *et al.*, 2002] Martin Peim, Enrico Franconi, Norman Paton, and Carole Goble. Query processing with description logic ontologies over object-wrapped databases. In *Proc. of the 14th International Conference on Scientific and Statistical Database Management (SSDBM'02)*, July 2002.

[Peim *et al.*, 2004] Martin Peim, Enrico Franconi, and Norman Paton. Applying functional languages in knowledge-based information integration systems. In Peter Gray, Larry Kerschberg, Peter King, and Alex Poulovassilis, editors, *The Functional Approach to Data Management*. Springer-Verlag, 2004.

[Reiter, 1992] Raymond Reiter. What should a database know? *Journal of Logic Programming*, 14(2,3), 1992.

[Schorlemmer *et al.*, 2002] M. Schorlemmer, S. Potter, and D. Robertson. Automated support for composition of transformtional components in knowledge engineering. Technical Report EDI-INF-RR-0137, Division of Informatics, University of Edinburgh, 2002.

[Serafini and Ghidini, 2000] Luciano Serafini and Chiara Ghidini. Using wrapper agents to answer queries in distributed information systems. In *Proceedings of the First Biennial Int. Conf. on Advances in Information Systems (ADVIS-2000)*, 2000.

[Serafini *et al.*, 2003] Luciano Serafini, Fausto Giunchiglia, John Mylopoulos, and Philip A. Bernstein. Local relational model: A logical formalization of database coordination. In *CONTEXT 2003*, pages 286–299, 2003.

[Tatarinov and Halevy, 2004] Igor Tatarinov and Alon Halevy. Efficient query reformulation in peer data management systems. In *Proceedings of the SIGMOD International Conference on Management of Data (SIGMOD'04)*, 2004.

[Ullman, 1997] J. D. Ullman. Information integration using logical views. In *Proc. of the 6th Int. Conf on Database Theory (ICDT'97)*, pages 19–40, 1997.

Related deliverables

A number of Knowledge web deliverable are clearly related to this one:

Project	Number	**Title** and relationship
KW	D2.2.2	**Specification of a benchmarking methodology for ontology alignment** takes advantage of the alignment process defined here for designing benchmark tests.
KW	D2.2.3	**State of the art on ontology alignment** presents the various ways to find alignments as they are described here.
SEKT	D4.4.1	**Mediation management** presents an alternative set of definitions that slightly differ from those given here in that they distinguish between alignment (similarity assessment between entities) and mapping (strict relations between entities).