CROSI
Capturing Representing and Operationalising Semantic Integration

A joint research project between

The University of Southampton
School of Electronics and Computer Science
Advanced Knowledge Technologies group

and

Hewlett Packard Laboratories@Bristol, UK

26th April 2005

6th month deliverable: ***Semantic Integration technologies survey***

Executive Summary

The University of Southampton and Hewlett Packard Laboratories at Bristol are collaborating in a joint project, CROSI, to investigate semantic integration. **CROSI**, which stands for Capturing, Representing, and Operationalising Semantic Integration, aims to advance the state-of-the-art for semantic integration technologies. Semantic integration has become a much debated topic in today's research agenda, especially with the advent of the Semantic Web. Its roots, however, go long time back in history of computer science with early attempts to resolve the problem found in the database literature of the eighties. It is concerned with the use of explicit semantic descriptions to facilitate information and systems integration. Due to the widespread importance of integration, many disparate communities have tackled this problem. They have developed a wide variety of overlapping but complementary technologies and approaches.

In this deliverable we present a comprehensive survey of the technological landscape in this area. As it is broadly defined and practiced by a number of diverse communities we aim to highlight this diversity. We complement and enhance previously published surveys in this area by focussing on convergence issues and techniques that can be carried over to similar problems. We also aim to identify concrete semantic integration cases which help us to inform a practical set of semantic integration criteria. These could be used to define desiderata for future semantic integration systems.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Semantic integration has become a much-debated topic and it is viewed as a solution provider in both industrial and academic settings. As systems become more distributed and disparate within and across organisational boundaries and market segments, there is a need to preserve the meaning of concepts used in everyday transactions of information sharing. The emergence of the Semantic Web, and its anticipated industrial uptake in the years to come, has made these transactions, arguably, easier to implement and deploy on a large scale in a distributed environment like the Internet. However, at the same time it poses some interesting challenges. For instance, we observe that the demand for knowledge sharing has outstripped the current supply. Moreover, even when knowledge sharing is feasible, this is only within the boundaries of a specific system, when certain assumptions hold, and within a specific domain. The reason for this shortcoming is, probably, the very environment and technologies that created a high demand for sharing: the more ontologies are being deployed on the Semantic Web, the higher the demand to share them for the benefits of knowledge sharing to achieve semantic integration.

One aspect of ontology sharing is to perform some sort of mapping between ontology constructs. That is, given two ontologies, one should be able to map concepts in one ontology onto those in the other. Further, research suggests that we should also be able to combine ontologies where the product of this combination will be, at the very least, the intersection of the two given ontologies. These are the dominant approaches that have been studied and applied in a variety of systems.

Stepping back in time, and reviewing what has been done in this front in the database world, we see a lot of similarities: schema integration is viewed as similar to ontology integration: in schema integration, once given a set of independently developed schemata, we construct a global view; this is viewed as similar to the problem of integrating two independently developed ontologies.

In the database world, schema integration is based upon schema matching: since the schemata are independently developed, they often have different structure and terminology, even if they model the same real world domain. The first step then is to identify and characterise interschema relationships, which is also known as schema matching. This step is then followed by a unification step in which the matching elements are put under a coherent integrated view or schema.

1.1 Semantic interoperability and integration

Before we reflect on similarities and technologies developed in these two domains, we elaborate on various perceptions of what is semantic interoperability and integration. These notions are much contested and fuzzy and have been used over the past decade in a variety of contexts and works. As reported in [72], in addition, both terms are often used indistinctly, and some view these as the same thing.

The ISO/IEC 2382 Information Technology Vocabulary defines interoperability as "the capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units.". In a debate on the mailing list of the IEEE Standard Upper Ontology working group, a more formal approach to semantic interoperability was advocated: Use logic in order to guarantee that after data were transmitted from a sender system to a receiver, all implications made by one system had to hold and be provable by the other, and there should be a logical equivalence between those implications[1].

With respect to integration, Uschold and Gruninger argue that "two agents are semantically integrated if they can successfully communicate with each other" and that "successful exchange of information means that the agents understand each other and there is guaranteed accuracy" [85]. According to Sowa, to integrate two ontologies means to derive a new ontology that facilitates interoperability between systems based on the original ontologies, and he distinguishes three levels of integration [82]: *Alignment*—a mapping of concepts and relations to indicate equivalence—, *partial compatibility*—an alignment that supports equivalent inferences and computations on equivalent concepts and relations—, and *unification*—a one-to-one alignment of all concepts and relations that allows any inference or computation expressed in one ontology to be mapped to an equivalent inference or computation in the other ontology.

Although these definitions of semantic interoperability and integration are by no means exhaustive, and despite the blurred distinction between these two concepts, they are indicative of two trends: on one hand, we have deliberately abstract and rather ambiguous definitions of what semantic interoperability and integration could potentially achieve, but not how to achieve it; and on the other hand, we have formal and mathematically rigorous approaches, which allow for the automatisation of the process of establishing semantic interoperability and integration.

1.2 Ontology mapping and schema matching

Semantic interoperability and integration are all-encompassing terms that refer to many different technologies. For the sake of simplicity, in this report we use *ontology mapping* and *schema matching* as the main strands of work in these broadly defined areas. We will also refer to peripheral areas such as *ontology merging and integration* and *schema integration* as these are often variations or enhancements of mapping.

There are many ways to describe what ontology mapping is. A generic definition based on an algebraic definition of an ontology introduced in [44]: "An ontology is then a pair, $O = (S, A)$ where S is the *(ontological) signature*—describing the vocabulary—and A is a set of *(ontological) axioms*—specifying the intended interpretation of the vocabulary in some domain of discourse." Typically, an ontological signature will be modelled by some mathematical structure. For instance, it could consist of a hierarchy of concept or class symbols modelled as a partial ordered set (*poset*), together with a set of relation symbols whose arguments are defined over the concepts of the concept hierarchy. The relations themselves might also be structured into a poset. Other definitions which resemble the notion of ontological signature are the "ontology" [42], "core ontology" [84] and "ontology

[1]Message thread on the SUO mailing list initiated at http://suo.ieee.org/email/msg07542.html

signature" [8]. In addition to the signature specification, ontological axioms are usually restricted to a particular sort or class of axioms, depending on the kind of ontology.

In [42] the authors argue that ontology mapping is the task of relating the vocabulary of two ontologies that share the same domain of discourse in such a way that the mathematical structure of ontological signatures and their intended interpretations, as specified by the ontological axioms, are respected. Structure-preserving mappings between mathematical structures are called morphisms; for instance a function f between two posets that preserves the partial order $(a \leq b) \rightarrow f(a) \leq f(b))$ is a morphism of posets. Hence they characterise ontology mappings as morphisms of ontological signatures as follows:

A total ontology mapping from $O_1 = (S_1, A_1)$ to $O_2 = (S_2, A_2)$ is a morphism $f : S_1 \rightarrow S_2$ of ontological signatures, such that, $A_2 \models f(A_1)$, i.e., all interpretations that satisfy O2 axioms also satisfy O1 translated axioms. This makes ontology mapping a theory morphism as it is usually defined in the field of algebraic specification [42].

In order to accommodate a weaker notion of ontology mapping they also provide a definition for partial ontology mapping form $O_1 = (S_1, A_1)$ to $O_2 = (S_2, A_2)$ if there exists a sub-ontology $O'_1 = (S'_1, A'_1)$ $(S'_1 \leq S_1$ and $A'_1 \leq A_1)$ such that there is a total mapping from O'_1 to O_2.

Ontology mapping only constitutes a fragment of a more ambitious task concerning the alignment, articulation and merging of ontologies. These terms can be understood in the context of the above theoretical picture. An ontology mapping is a morphism, which usually will consist of a collection of functions assigning the symbols used in one vocabulary to the symbols of the other. But two ontologies may be related in a more general fashion, namely by means of relations instead of functions. Hence, we will call ontology alignment the task of establishing a collection of binary relations between the vocabularies of two ontologies. Since a binary relation can itself be decomposed into a pair of total functions from a common intermediate source, we may describe the alignment of two ontologies O_1 and O_2 by means of a pair of ontology mappings from an intermediate source ontology O_0 (depicted in the figure below). We shall call the intermediate ontology O_0, together with its mappings, the *articulation* of two ontologies.

$$\begin{tikzcd}
 & O_0 \arrow[dl] \arrow[dr] & \\
O_1 & & O_2
\end{tikzcd}$$

Articulation

$$\begin{tikzcd}
 & O_0 \arrow[dl] \arrow[dr] & \\
O_1 \arrow[dr] & & O_2 \arrow[dl] \\
 & O &
\end{tikzcd}$$

Merging

Figure 1.1: Ontology Articulation and Merging

Finally, an articulation allows for defining a way in which the *fusion* or *merging* of ontologies has to be carried out. The intuitive idea is to construct the minimal *union* of vocabularies S_1 and S_2 and axioms A_1 and A_2 that respects the articulation, i.e., that is defined modulo the articulation. This corresponds to the mathematical *pushout* construct. Again, this strong notion of merging can be relaxed by taking the articulation of two sub-ontologies of O_1 and O_2 respectively, and defining the merged ontology O according to their articulation.

Schema matching on the other hand is the operation which takes as input two schemata and produces a mapping between elements of these schemata that correspond semantically to each other. This correspondence has been a focal point of debate in the database literature for years, as there are many understandings and uses of the term. The formal definitions we used for ontology mapping, could be applied to schema matching. Practically speaking though, in the database literature, the notion of a Match operator is dominant and algebraic approaches are only found in the early database theory manuscripts. Another observation is that schema matching is typically performed manually, perhaps supported by a graphical user interface [76].

In [81] the author argues for a distinction between syntactic and semantic matching, with the former being the dominant approach in most database schema matching works. In syntactic matching, semantic correspondences are determined using syntactic similarity measures (normally in the range 0..1) or confidence measures whereas semantic matching [35] should calculate mappings based on *semantic relationships* of the form *equal*, *subsumes*, *subsumedBy* etc.

1.3 Application areas

Semantic interoperability, integration and the technologies that represent them in the artificial intelligence (AI, henceforth) and database (DB, henceforth) worlds, ontologies and schemata, respectively, are solutions to the problem of semantic heterogeneity. This problem arises from the different uses of meaning that is attached to similar or even identical constructs (let them be ontology concepts, or schema elements).

To motivate the importance of semantic integration, we briefly present some key application areas where semantic heterogeneity occurs and there is a need for resolving it. This is a non-exhaustive list but merely an indication of the diversity for the application domain of semantic integration.

1. ***Database schema integration***: "Given a set of independently developed schemas, construct a global view." [81]. The schemata often have different structure and the process of integration aims to unify matching elements. Matching is a whole field in its own right and is the core operation of schema integration.

2. ***Data warehouses***: This is a variation of the schema integration where the data sources are integrated into a data warehouse: "A data warehouse is a decision support database that is extracted from a set of data sources. The extraction process requires transforming data from the source format into the warehouse format." [42]. These transformations could be assisted by database schema matching operations.

3. ***E-Commerce***: Trading partners frequently exchange messages that describe business transactions. As each trading partner uses its own message format, this creates the problem of heterogeneity. That is, message formats may differ in their syntax (EDI structured, XML formatted, etc.) or use different message schemata. To enable systems to exchange messages, application developers need to convert messages between the formats required by different trading partners.

4. ***Semantic query processing***: "A user specifies the output of a query (e.g., the SELECT clause in SQL), and the system figures out how to produce that output (e.g., by determining the FROM and WHERE clause on SQL)." [42] The heterogeneity arises when the user specifies the query output in terms which are different from those used in the schema.

5. ***Ontology integration (or merging)***: Given two distinct, and independently developed ontologies, produce a fragment which captures the intersection of the original ontologies. This area is similar to that of schema integration but more difficult in nature due to the rich and complex knowledge representation structures found in ontologies.

6. ***Ontology mapping***: This is a subset of the previous area, mapping ontologies is a step towards integration and it is often the case that mapping ontologies is adequate for most interoperability scenarios on the Semantic Web.

7. ***Semantic Web agents' interoperability***: A pre-requisite for Semantic Web agents to collaborate is their ability to understand and communicate their mental models. These are often model in the form of an ontology and it is likely to be distinct albeit modelling the same universe of discourse. Mapping their ontologies is a major area of interest where automated and scalable solutions are also sought due to the vast number of agents involved in these scenarios.

8. ***Web-based systems interoperability***: interoperability is also a pre-requisite for a number of Web-based systems who serve distinct applications [41]. In particular, in areas where there is diverse and heterogeneous Web-based data acquisition, there is also a need for interoperability.

1.4 Technological landscape

There are a number of solutions proposed and developed over a number of years, in both the database and artificial intelligence world. These are reviewed extensively in Section 4.4 but here we summarise them briefly. Early works to tackle the semantic heterogeneity problem emerged in the eighties from the database community. In particular, the notion of federated databases, where schemata are treated globally and are exchanged between designers and among disparate systems, informed the requirements for techniques which assist database designers to do schema matching.

Most of these techniques were based on syntactic features of the schemata used, and employed a variety of heuristics to kick off similarity measure algorithms, well known in the information retrieval community. A dominant technique has been the use of correspondence values, typically in the range 0..1, which supposedly captures the intended overlap of two mapped elements. These approaches had their deficiencies though, as it was often observed that schemata with virtually similar syntactic elements were describing different real world concepts. The crux of the problem was the lack of semantic information carried by the designated database schema elements. This information is important for the validity of the proposed mapping which made verification check of the proposed mappings by a human user, a necessity. This is one of the reasons why these approaches could not scale up, neither trusted for employing the idea of federated databases in a large scale.

In the mid to late nineties, more complex and richer knowledge representation models, namely ontologies, became popular and with the advent of global infrastructures for sharing semantics, like the Semantic Web, the need to share ontologies fuel the interest for ontology mapping. Unlike the attempts for schema matching in the database world, ontology mapping efforts are using a wider variety of techniques for finding mappings. There are many similar techniques, like the use of syntactic similarity, certain approaches to correspondence values (though sometimes more elaborated than a numerical range 0..1), but there are also more advanced and semantics-enabled techniques, mainly due to the richer nature of ontologies. For example, there are ontology mapping systems that exploit the hierarchical lattice inherited in ontology structure (partially ordered lattices), take advantage of ontology formalisms which allow certain semantic information to be attached to a particular element (from formal axioms to informal textual descriptions), and some also use the underlying deductive mechanism to infer mappings (for example, use of Description Logics reasoning).

Another common denominator is the use of instance information. As both ontologies and database schemata are expected to be instantiated (either directly with declared ontology instances or indirectly with an instantiated knowledge base that adheres to that ontology or a database that adheres to that schema), some works from both communities use this information to compute mappings.

Recently, the use of machine learning for computing mappings, has found many supporters and uses. The main reason lies behind the observation that ontology or database schema mapping nowadays, is an expensive endeavour to be carried out only by human users. The proliferation of ontologies on the (Semantic) Web and the sheer number of database schemata to be matched, call for automated support to compute mappings in acceptable time. Machine learning is seen as a solution provider as it does most of the hard job in an automated fashion, however, there open issues to resolve, especially with the training data for learning algorithms.

Finally, the use of heuristics was always the easy and preferable choice of engineers. This is not a surprise to everyone who has attempted to do mapping: heuristics are cheap to develop, easy to deploy, and support automation. However, the main problem with heuristics is that they are easily defeasible. Even well-crafted heuristics for a particular case can fail in similar situations. Attempts to solve this problem go beyond the use of syntactic features, linguistic clues, and structural similarities when building heuristics, and use as much semantic information as possible. That is, use the intended meaning of the concepts to be mapped. However, this is not always feasible as semantics are often not captured in the underlying formalism, and a human expert is needed to give their precise meaning.

1.5 Communities involved

There two, broadly speaking, stakeholders in semantic integration technology: producers of such solutions and users. Their roles are sometimes swappable in the sense that semantic integrators are also users of this technology as they could use semantically integrated ontologies in their solutions. The two main communities of interest in this field are the old database community with their data integration and schema matching expertise, and the newest ontology engineering community with ontology mapping and merging technology. Users and beneficiaries of this technology are mostly Semantic Web practitioners and developers, as semantic integration is seen as a prerequisite to accomplish many of the Semantic Web envisaged tasks.

Large corporations' mergers are also calling for (semantic) integration technology. This could be (Semantic) Web based. The variety of technologies involved in a semantic integration solution: machine learning, linguistics, heuristics, graph structures, formal logic, etc., brings a lot more peripheral communities onboard. Communities like the wider (Semantic) Web and agents' are contributing with their test cases and scenarios whereas application areas as those described earlier inform the requirements for building and advancing semantic integration systems.

1.6 Organisation of this report

The remaining of this report is organised as follows:

Chapter 2 describes some of the mapping cases commonly found in the literature. These range from real-world mapping scenarios and challenges set for semantic integrators to identification of exemplar domains where mapping is mostly practiced.

Chapter 3 contains the bulk of this report. It is a comprehensive list of mapping systems originating from a variety of communities to highlight the diversity of works and emphasize the breadth of solutions used by practitioners.

Chapter 4 attempts to classify all this information. We refer to existing typologies and classifications and we introduce further ones.

Chapter 5 describes mapping criteria which could be met by semantic integration systems. We also elaborate on the desiderata of developing semantic integration systems. Finally, we wrap up this survey by highlighting a number of issues for consideration from a semantic integration practitioner.

Chapter 2

Mapping Scenarios

In this section we present use cases from existing approaches in the hope that a common grounding of mapping cases can be identified and possible scenarios can be rooted. Two categories of mapping scenarios are discussed: explicit and implicit.

Explicit here means that the mapping requirements and results are clearly specified, e.g. the e-Commerce Product Classification Challenge, the EON and I^{3}CON ontology alignment repositories and the Life Science ontologies from MyGrid project. On the contrary, implicit mapping cases are either purposely built to or adopted from real-life cases and trimmed to demonstrate the applicability a particular approach. The implicit cases normally do not have mapping specifications, leaving the users plenty of room to manoeuvre and define ad hoc mapping criteria.

2.1 Explicit Mapping Scenarios

Due to the lack of universally accepted test cases and benchmarks in evaluating not only schema matching and ontology mapping but also information integration in general, we resort to publicly accessible on-line repositories or well-documented projects within the domain. This is under both theoretical and practical considerations. EON , I^{3}CON and the e-Commerce Product Classification Challenge provide benchmark-like cases with expected mapping requirements and a reasonable diversity with regard to the modelling domains.

2.1.1 Catalogue Matching

The OntoWeb-1, SIG-1, e-Commerce, Product Classification Challenge [79][1]

The OntoWeb EU NoE SIG-1 devised a scenario for mapping product classification catalogues. The resources were obtained from the United Nations Standards Products and Services System (UNSPSC - http://www.unspsc.org/). An initiative to enhance this classification is the Universal Content Extended Classification (UCEC - www.ucec.org) but the Website is no longer live, as the experiment was set back in 2001. Similar initiatives are the Electronic Commerce Content Standards Association (ECCMA - www.eccma.org) which provides the ECCMA Global Attribute Schema (http://eccma.org/ega/). From the ECCMA Website, we can download the main UNSPSC files, they are all in MS Excel format with some hierarchical information made implicit, e.g., for the “Paper Products

[1]http://www.computer.org/intelligent/ex2001/pdf/x4086.pdf

XX Segment	The logical aggregation of families for analytical purposes
XX Family	A commonly recognised group of inter-related commodity categories
XX Class	A group of commodities sharing a common use or function
XX Commodity	A group of substitutable products or services
XX Business Function	The function performed by an organization in support of the commodity

Table 2.1: Specification of UNSPSC product classfication

and Materials" category, the UNSPSC audit, v7.0901 gives us key, and code information alongside a textual description:

100748 14000000 Paper Materials and Products

Which branches into other codes for sub-categories of "paper", i.e.,

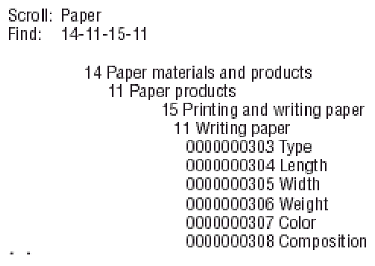
108283 73111600 Pulp and paper processing

Further details are given in the EGCI version:

EGCI	Segment	Family	Class	Commodity	Title	Add_Version	Add_Date
009074	14	11	00	00	Paper products	2	2/22/1999 0:00

According to the early challenge call, there are five levels in the UNSPSC classification, each of which has a textual description as illustrated in Table 2.1.

An example of the product hierarchy is illustrated in the following figure where Segment 14 is the "Paper materials and products", Family 11 the "Paper products", Class 15 the "Printing and writing paper" and Commodity 11 the "Writing paper". A series of attributes of writing paper is further defined, e.g. "0000000303 Type", etc.



```
Scroll: Paper
Find:   14-11-15-11

           14 Paper materials and products
              11 Paper products
                    15 Printing and writing paper
                      11 Writing paper
                        0000000303 Type
                        0000000304 Length
                        0000000305 Width
                        0000000306 Weight
                        0000000307 Color
                        0000000308 Composition
```

Figure 2.1: An example of the product hierarchy

In a similar domain, the German initiative, Ecl@ss (www.eclass.de and www.eclass-online.com) has developed a similar hierarchy with different numbering schema. Some intuitions with respect to the desired mapping were given:

- "Some counter-intuitive categories appear, such as office supplies (other) as a subclass of office supplies in Ecl@ss, and paper pulp as a raw, rather than semi-finished material in UNSPSC/UCEC."

- In UNSPSC/UCEC all paper products and materials are organised in a single tree, whereas in the Ecl@ss paper products are more functionally grouped. For example, paper pulp needs a connection (probably) under semi-finished materials in Ecl@ss, paper towels would (probably) fall under operation/cleaning equipment in Ecl@ss.

- Some categories have either or no more than one, "equivalent" class in different product classification schemes. For example, printing and writing paper constitutes a single category in UNSPSC/UCEC, but two separate categories in Ecl@ss.

One of the expected mapping solutions between UNSPSC (UCEC) "Paper materials & products" and the Ecl@ss hierarchies is illustrated in figure 2.2

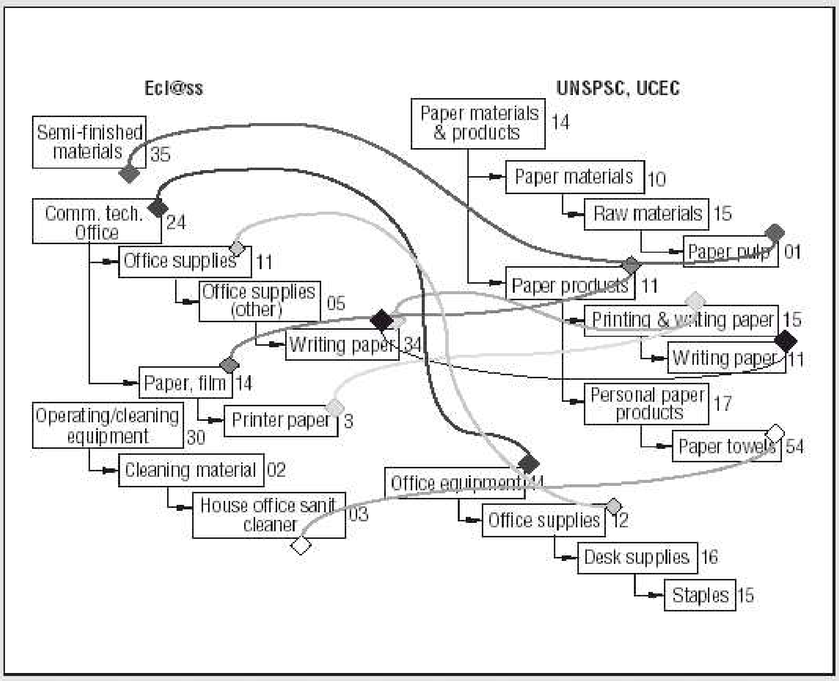


Figure 2: The mapping problem between two different product classification schemes for writing paper.

Figure 2.2: Mapping between UNSPSC/UCEC and Ecl@ss

2.1.2 EON ontology alignment and I^3CON contests

The increasing number of systems and methods for schema matching and/or ontology mapping (integration) fueled the need for a consensus. Two separate events, the EON ontology alignment contest and the I^3CON ontology alignment demonstration competition, are now a joint force in prompting such a consensus.

The **EON alignment contest**[2] is an initiative partly funded by the EU NoE OntoWeb and KnowledgeWeb. The domain of this first test is Bibliographic references. It is, of course, based on a subjective view of a bibliographic ontology. There can be many different classifications of publications (based on area, quality, etc.). We chose the one that is common among scholars based on means of publications.

[2]Described on line at: http://co4.inrialpes.fr/align/Contest/

This reference ontology contains 33 named classes, 39 object properties, 20 data properties, 56 named individuals and 20 anonymous individuals. The reference ontology is put in the context of the Semantic Web by using other external resources for expressing non-bibliographic information. It takes advantage of FOAF (http://xmlns.com/foaf/0.1/) and iCalendar (http://www.w3.org/2002/12/cal/) for expressing the *People*, *Organization* and *Event* concepts. This reference ontology is a bit limited in the sense that it does not contain circular definitions nor attachment to several classes. Similarly, the kind of proposed alignments is also limited: they only match named classes and properties, and mostly use the "=" relation with confidence of 1. The ontologies are described in OWL-DL and serialised in RDF/XML format.

The contest organisers provide the participants with a complete test base, including couples of ontologies to align as well as expected results. The test is based on one particular ontology dedicated to a narrow domain and a number of alternative ontologies of the same domain for which alignments are provided. The ontologies are provided in OWL. The expected alignments are provided in a standard format expressed in RDF/XML and described in http://co4.inrialpes.fr/align/. From the ontology to compare, the competitors are able to produce their alignments in the same format. They can also compute a number of measures of their results. They can use the ontology provided in http://www.atl.external.lmco.com/projects/ontology/ for describing their results.

The **I^{3}CON** (Information Interpretation and Integration Conference)[3] emerged from an ontology alignment information dissemination effort organised by Lockheed Martin, the Ontology Alignment Source (OAS)[4]. The Information Interpretation and Integration Conference (I^{3}CON, pronounced "icon") supports the ontology and schema interpretation and integration research communities. The goal of I^{3}CON is to provide an open forum and a software framework for the systematic and comprehensive evaluation of ontology/schema interpretation and integration tools. I^{3}CON is modelled after the NIST Text Retrieval Conference (TREC), which has succeeded in driving progress in information retrieval research. Like TREC, I^{3}CON provides test data and a software platform on which participants run their own algorithms.

The following mapping cases are described briefly in the I^{3}CON site:

- Animals' Ontologies (http://users.ebiquity.org/ hchen4/ont/animals in OWL format) and a modified version of it.
- Soccer ontology (http://www.lgi2p.ema.fr/ ranwezs/ontologies in DAML format) describes the basic concepts of soccer. Basketball ontology (also in DAML) is a modified version of soccer.daml which uses basketball concepts.
- Computer Science ontology (available in RDF format from http://www.cs.umd.edu/projects/plus/DAML/onts/) and a similar one from http://cicho0.tripod.com/: they both represent the CS departments at the University of Malta and the University of Maryland, respectively.
- Hotel ontologies which describe the characteristics of hotel rooms. The two ontologies are equivalent, but they describe the same concepts in different ways.
- Network Ontologies which describe the nodes and connections in a local area network. One focuses more on the nodes themselves, while the other one is more encompassing of the connections.

[3]Described online at: http://www.atl.external.lmco.com/projects/ontology/i3con.html.
[4]Described online at: http://www.atl.external.lmco.com/projects/ontology/index.html.

- An ontology about Russia which describes an overview of Russia including locations, objects, and cultural elements
- Simple wine ontologies derived from Wine.com taxonomy. WineA is in English and WineB is in “babelized” Spanish.
- Ontologies of military weapons derived from the CIA World Fact Book ontology and the Cyc ontology.

2.1.3 Life Science

myGrid is a UK e-Science project aiming to provide grid-enabled services to meet the immediate needs of bioinformatics. More specifically, myGrid “is building high level services for data and application resource integration” (see also figure 2.3). As far as the integration is concerned, as put forward by the myGrid group [53]:

> The biggest limitation of service descriptions at the moment is their assumption of a single class of user. For example, the myGrid ontology is aimed at bioinformaticians, as this seemed the main user base. Previous systems, such as TAMBIS [4], have been more aimed at biologists; hence biological concepts such as “protein” are modelled and presented instead of bioinformatics concepts such as “Swissprot ID”.

It is reasonable to assume that there is desire and a common ground for sharing TAMBIS specific services with myGrid specific ones, both of which are based on underlying ontological concepts. Therefore, myGrid platform offers not only different versions of the real-life domain ontology that provide the concrete ground for applying various mapping techniques, but also a series of services relying on the ontology that present themselves as the means for evaluating mapping where the interoperability of services can be grounded by the mapping of underlying ontologies.

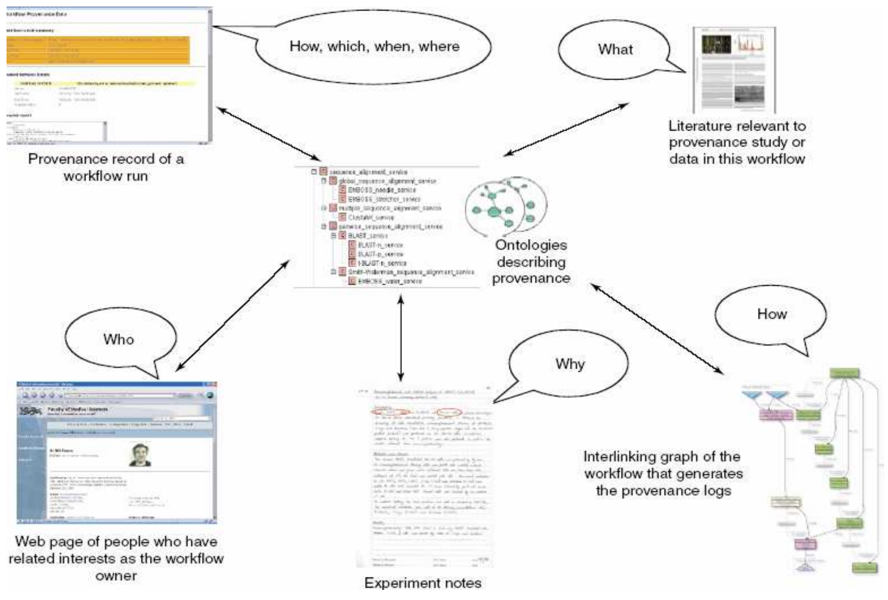

How, which, when, where
What
Provenance record of a workflow run
Literature relevant to provenance study or data in this workflow
Ontologies describing provenance
Who
Why
How
Interlinking graph of the workflow that generates the provenance logs
Web page of people who have related interests as the workflow owner
Experiment notes

Figure 2.3: myGrid architecture

A possible scenario taking advantage of the myGrid services could be the drug discovery. When considering the adverse effect of a new drug, it might be necessary to communicate with different sites and pull out information of the *in vivo* and *in vitro* tests done on the drug. Different sites of the community might have their own terminology. Combining their services means to establish a mutual understanding among different sites and thus a mapping between their terminologies.

Source ontologies available for experimentation are at

http://www.cs.man.ac.uk/∼wroec/mygrid/ontology/mygrid.owl

http://www.cs.man.ac.uk/∼wroec/mygrid/ontology/cutdown-demo.owl (cut-down version)

http://www.cs.man.ac.uk/∼wroec/mygrid/ontology/mygrid-reasoned.owl (DL-processed version)

Mappings between the raw OWL ontologies and the DL-processed ones may present another interesting user case of mapping.

2.2 Implicit Mapping Scenarios

These are purposely built test cases which although might perform well, do not have explicit descriptions of mapping cases nor define the problem under investigation.

2.2.1 e-Commerce related

e-Commerce applications rely heavily on electronic product and service descriptions. The lack of suitable and universal standards makes it particularly difficult to exclude human factors from any solutions. However, manual alignment approaches are time consuming and error prone. A few approaches aim at overcoming such a barrier. For instance, in the QOM system, the authors used an example case from the domain of car retailing. Both ontologies are of limited size and populated with a few instances.

CUPID [56] uses a purpose-made example with two schemata for purchase orders. As illustrated in figure 2.4, purchase order entity might be described in totally different ways (different labels, structures and attributes). Matching among schemata provides a common ground on which the software agents and services processing the orders can be built. A similar example is used in the COMA system.

ConcepTool [60] tries to articulate two different vehicle ontologies, namely the CARRIER's view of transportations containing type of vehicles, ownership, etc. and the FACTORY's view containing a different categorisation of vehicle types, buyers etc.

2.2.2 Database Schema Integration and Ontology Mapping

The integration and/or mapping between database schemata and ontologies are probably the most applied domains. It is evident that most of the systems and approaches face the problem of verifying their mapping results. The lack of a widely accepted benchmark for integration forces researchers to build ad hoc test cases. Many approaches chose to work around the verification barrier by mapping ontologies of familiar domains.

University and Education Breis and Bejar's system [15] presented two different ontologies of SCIENCE_FACULTY of which different attributes are used in describing subparts (e.g. BUILDING, PERSON) of the faculty. Mappings are performed based on

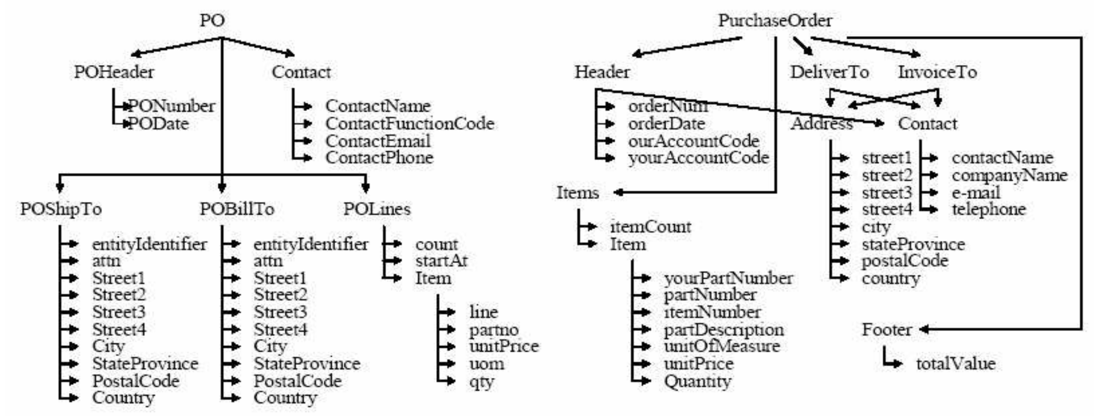


Figure 2.4: Cupid Example

synonyms and attributes. The OIS framework [17] works on global ontology G_u containing two binary relations, WorksFor and Area, local ontology S_1 containing a binary relation InterestedIn and local ontology S_2 containing a binary relation GetGrant and GrantFor. The mapping M_u is formed by the following correspondences:

$$\langle V1 : \mathsf{InterestedIn}; complete\rangle \text{ with } V1(r;f) \leftarrow \mathsf{worksFor}(r;p) \wedge \mathsf{Area}(p;f)$$
$$\langle \mathsf{worksFor}; V2; sound\rangle \text{with} V2(r;p) \leftarrow \mathsf{GetGrant}(r;g) \wedge \mathsf{GrantFor}(g;p)$$

In the correspondences given above, V1 and V2 are views which represent the best way to characterise the objects that satisfy these views in terms of the concepts in the local ontologies S_1 and S_2. The characterisations of these correspondences are sound and complete.

Madhavan and colleagues' framework [54] model students (address, major, GPA, gender, s-number) and the courses (c-number, description) they are enrolled in. Knowledge could be represented in different models, e.g. DAML+OIL and relational schema. A fragment of the mapping results is as follows:

$$\begin{aligned}
\mathsf{Univ.Student(std, ad, gpa)} &\subseteq \mathsf{MyUniv.STUDENT(std)} \wedge \mathsf{MyUniv.Lives\text{-}In(std, ad)} \\
&\wedge \mathsf{MyUniv.Grade(std, gpa)} \\
\mathsf{Univ.Student(std, ad, gpa)} &\subseteq \mathsf{YouUniv.student(std, ad, x, gpa, y)} \\
\mathsf{Univ.Arts\text{-}Std(std)} &\subseteq \mathsf{MyUniv.ARTS\text{-}STD(std)} \\
\mathsf{Univ.Arts\text{-}Std(std)} &\subseteq \mathsf{YouUniv.student(std, x, ``arts", y, z)}
\end{aligned}$$

IF-Map [43] was applied to map AKT's project ontologies, namely *AKTReference* to *Southampton*'s and *Edinburgh*'s local ontologies. Apart from mapping concepts, like *AKTReference*'s document to *Southampton*'s publication, IF-Map also maps relations: *AKTReference*'s hasappellation to *Southampton*'s title. The arities of these relations and the way local communities are classifying their instances allow this sort of mapping, whereas in other situations this would have been inappropriate, when for example title refers to title of a paper. These mappings were generated automatically.

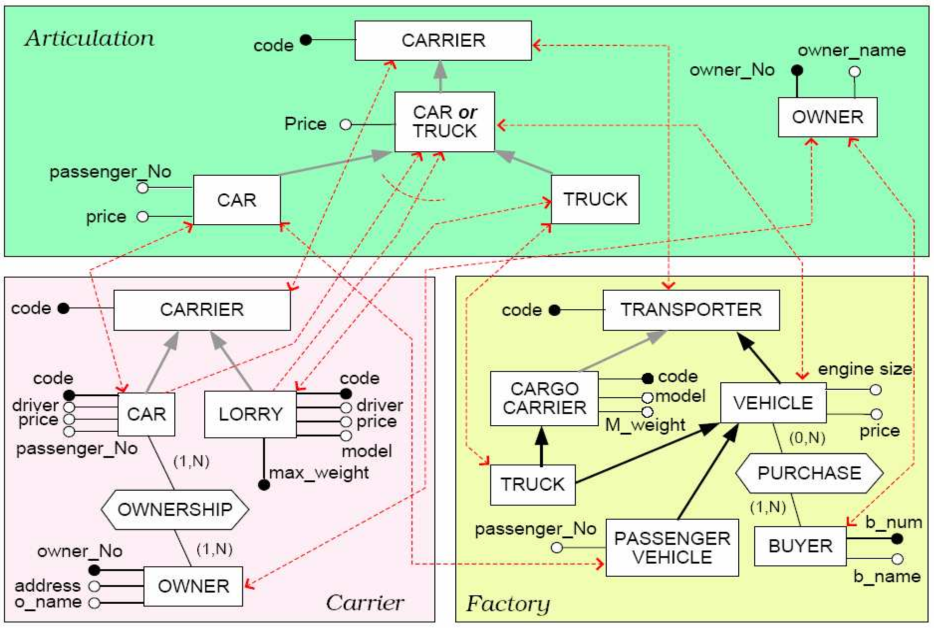


Figure 2.5: Conceptool Example

Marital relationships: Maedche and Staab [57] developed MAFRA of which the mapping capabilities are demonstrated through two small ontologies depicted in UML. A simple ontology defining humans as Man and Woman is mapped against a more complicated one of events related to marital status, e.g. Marriage, Divorce, Birth, Death, etc.

2.2.3 e-Government: organisational and governmental structures

A few systems have used real-life data, either accessible via Internet or provided by data vendors. For instance, ONION [31] has been experimenting with airline and governmental ontologies. In both cases, the ontologies are manually crafted, but are grounded to concrete use cases. More specifically, the airline ontologies are based on terminology used on United Airlines website and US Air Force (TRANSCOM) while the ontologies of governmental structures are constructed from NATO government Webs-sites containing variants such as DepartmentofDefence and DefenseMinistry.

DIKE uses the database schemata of the Italian Central Government Offices (ICGOs) which contains "300 databases and large amounts of data"[78]. Exemplar concepts are, among others, Judicial_Person, Ministry, Payment, etc. (see figure 2.6).

Clifton et. al. [19] reported their experiences with Delta and SEMINT systems using data authorised by US Air Force which include databases such as Advanced Planning System, Computer Aided Force Management System, Wing Command and control System and Air Force Air Operations Data Model.

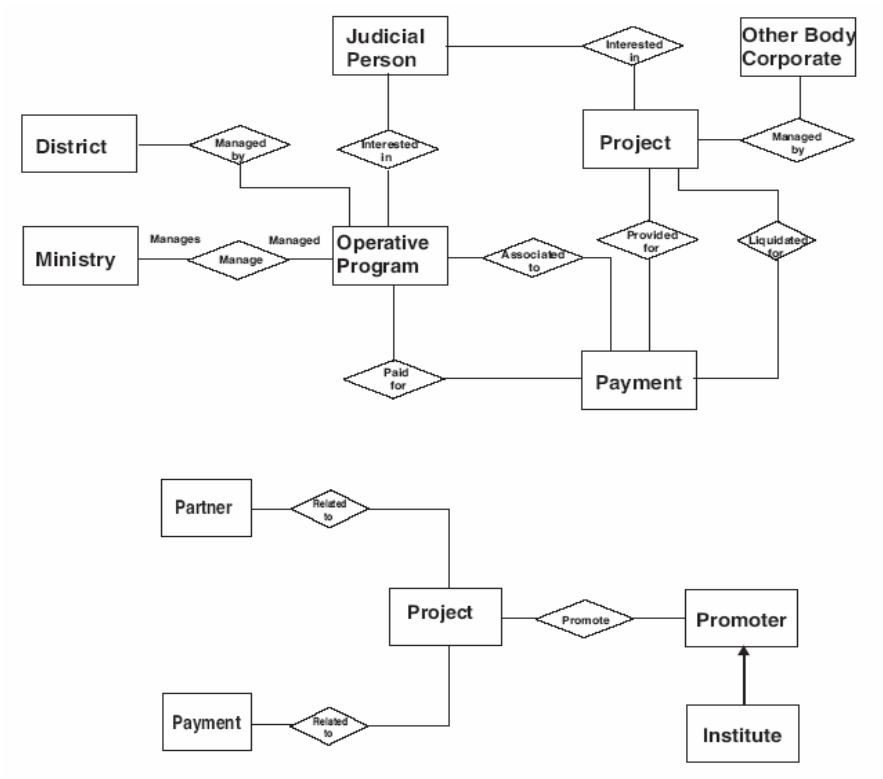


Figure 2.6: DIKE Example

Chapter 3

Semantic Integration Systems

Different communities adopt the notion of semantic integration in different ways. In this section, we aim to unify them into three categories: information integration, database schema matching, and ontology mapping. These notions are broadly defined and as there is a lack of concrete and distinct definitions, we expect overlaps among them, to a certain extent.

Substantial research has been carried out in the semantic integration area, in general. Some of it, is focusing on finding the formal treatment for a variety of issues concerning integration; some concentrate on developing a particular method or technique for mapping (or matching) more efficiently; and some try to develop a comprehensive environment into which integration is only one of the core tasks that need to be addressed. To reflect the diversity of the works we reviewed, the approaches included in this section are also divided into formal frameworks, methods, and platforms.

3.1 Formal Frameworks

A few researchers concentrate on exploring the theoretical foundation of mapping. They envisioned what a mapping framework should look like, how the mapping should proceed, and how the mapping should be represented to facilitate post-mapping processes.

3.1.1 Information Integration

Madhavan et.al system

Madhavan and colleagues [54] developed a theoretical framework and propose a language for mapping among domain models. Their framework enables mapping between models in different representation languages without first translating the models into a common language, the authors claim. The framework uses a helper model when it is not possible to map directly between a pair of models, and it also enables representing mappings that are either incomplete or involve loose information. The models represented in their framework, are representations of a domain in a formal language, and the mapping between models consists of a set of relationships between expressions over the given models. The expression language used in a mapping varies depending on the languages of the models being mapped. The authors claim that mapping formulae in their language can be fairly expressive, which makes it possible to represent complex relationships between models. They applied their framework in an example case with relational database models. They also define a typology of mapping properties: query answerability, mapping inference, and mapping composition. The authors argue: “A mapping between two models rarely maps all the concepts in one model to all concepts in the other. Instead, mappings typically loose some information and can be partial or incomplete."

Question answerability is a proposed formalisation of this property. Mapping inference provides a tool for determining types of mappings, namely equivalent mappings and minimal mappings; and mapping composition enables to map between models that are related by intermediate models.

3.1.2 Database Schemata Integration

Alagić and Bernstein's Generic Schema Model

Alagić and Bernstein proposed a generic schema management model that is applicable to variety of data models with only "some customization required for the data model and problem at hand"[2]. Their framework is built on category theory addressing morphisms at two different levels: the meta level of database schemata and their transformation as schema morphisms; and the instance level of databases conforming to schemata at the meta level. The major contribution, the authors argue, is the preservation of integrity constraints during the schema transformations.

More specifically, a database schema is defined as a tuple (Sig, E) where Sig is the signature of schema and E is the set of integrity constraints. Based on such a formal definition, schema morphism is formalised as morphism between the signatures and a mapping of constraints that ensures "the integrity constraints of the source schema are transformed into constraints that are consistent with those of the target". The authors further elaborated schema equivalence as structural and semantic equivalence of which the difference is when integrity constraints are taken into account.

Alagić and Bernstein continued with a formal specification of schema integration:

$$\begin{tikzcd}
Sch_m \arrow[r, "\phi_1"] \arrow[d, "\phi_2"'] & Sch_1 \arrow[d, "p"'] \\
Sch_2 \arrow[r, "q"] & Sch_{12}
\end{tikzcd}$$

Figure 3.1: Schema Integration in Alagić and Bernstein's Model.

Sch$_m$ is the matching part of Sch$_1$ and Sch$_2$. Arrows in the above diagram give morphisms between different schemata.

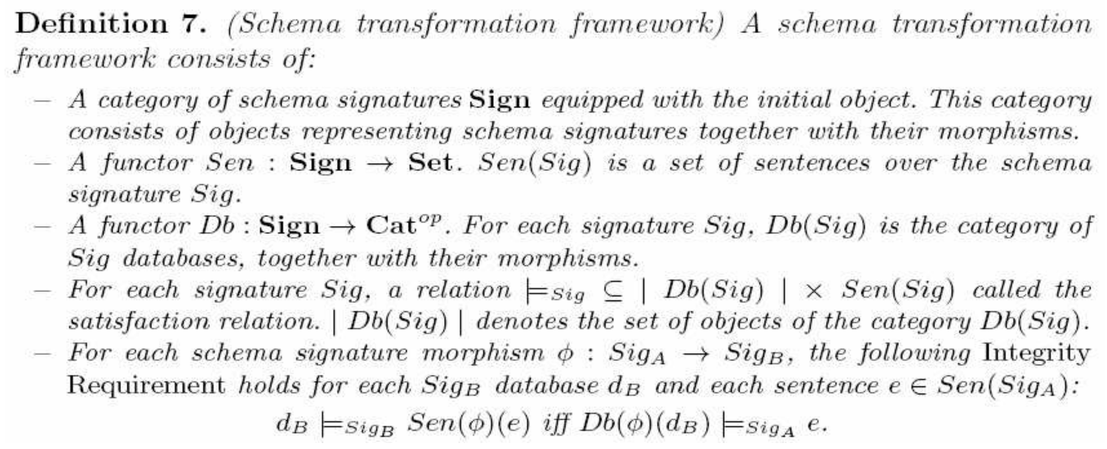
The most important part of their work, as emphasised by the authors, is the proposal of a generic schema transformation framework that can be instantiated using any data models, e.g. relational, object-oriented, and XML. The definition of such a framework is quoted in Figure 3.2 which gives the diagram shown in Figure 3.3.

They concluded by constructing the object oriented schema from the generic framework integration and transformation of a particular category of data models.

3.1.3 Ontology Mapping

Franconi and colleagues ontology mapping framework

Franconi and colleagues [29] elaborated on a category theoretical view of ontology mapping and merging. A mapping is defined as a four-tuple $\langle e, e', n, R \rangle$ where e and e' are the



Definition 7. *(Schema transformation framework) A schema transformation framework consists of:*

- *A category of schema signatures* **Sign** *equipped with the initial object. This category consists of objects representing schema signatures together with their morphisms.*
- *A functor Sen :* **Sign** $\rightarrow$ **Set**. *$Sen(Sig)$ is a set of sentences over the schema signature Sig.*
- *A functor Db :* **Sign** $\rightarrow$ $\mathbf{Cat}^{op}$. *For each signature Sig, $Db(Sig)$ is the category of Sig databases, together with their morphisms.*
- *For each signature Sig, a relation $\models_{Sig} \subseteq \mid Db(Sig) \mid \times Sen(Sig)$ called the satisfaction relation. $\mid Db(Sig) \mid$ denotes the set of objects of the category $Db(Sig)$.*
- *For each schema signature morphism $\phi : Sig_A \rightarrow Sig_B$, the following* Integrity Requirement *holds for each Sig_B database d_B and each sentence $e \in Sen(Sig_A)$:*

$$d_B \models_{Sig_B} Sen(\phi)(e) \text{ iff } Db(\phi)(d_B) \models_{Sig_A} e.$$

Figure 3.2: Definition of Schema Transformation

$$\begin{tikzcd} Db(Sig_A) \arrow[r, "\models_{Sig_A}"] & Sen(Sig_A) \arrow[d, "Sen(\phi)"] \\ Db(Sig_B) \arrow[u, "Db(\phi)"] \arrow[r, "\models_{Sig_B}"] & Sen(Sig_B) \end{tikzcd}$$

Figure 3.3: Illustration of schema transformation model.

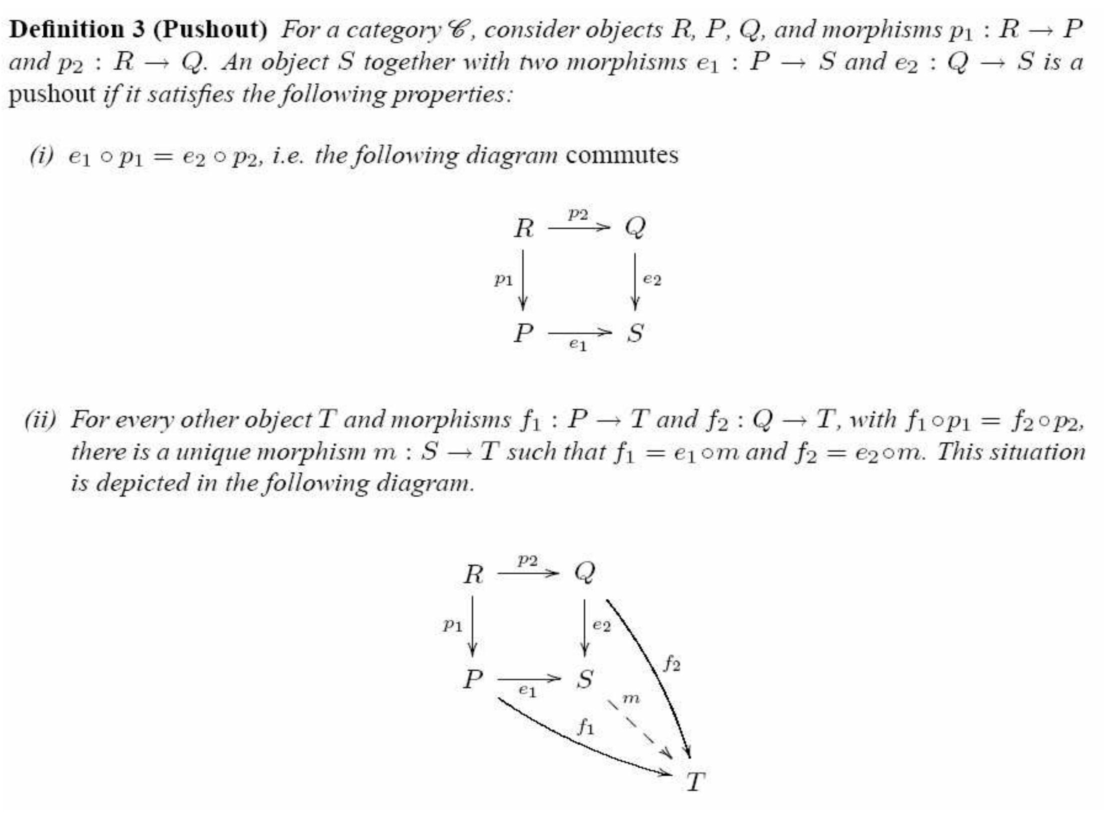
entities between which a relation is asserted by the mapping, n a degree of trust in that mapping, and R is the relation associated to a mapping.

Grounded on category theory, Franconi, *et.al* continued with a formal treatment of ontology merging (and alignment). The authors defined a category, in which objects are ontologies and morphisms between objects serve as "meaningful transitions between ontologies". The authors also provide formal definitions of ontology merging that is depicted as a *pushout* between objects from the category of ontologies.

The above definition, the authors argued, enforces the elements in the resultant ontology S that are related by R while prevents undesired and irrelevant elements by emphasising the existence and uniqueness of "*a* factorization m".

The authors also give a step-by-step guidance of how to put the ontology merging formal framework into practice:

- Decide on ontology representation language used;
- Determine what are the suitable morphisms;
- Determine what ontology mapping is;
- Determine what *pushout* is (see also figure 3.4);
- Algorithmize how to obtain the mapping;
- Algorithmize how to obtain the *pushout*.



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

$$\begin{tikzcd}
R \arrow[r, "p2"] \arrow[d, "p1"'] & Q \arrow[d, "e2"] \\
P \arrow[r, "e_1"'] & S
\end{tikzcd}$$

(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.

$$\begin{tikzcd}
R \arrow[r, "p2"] \arrow[d, "p1"'] & Q \arrow[d, "e2"] \arrow[dr, bend left, "f_2"] & \\
P \arrow[r, "e_1"'] \arrow[rr, bend right, "f_1"] & S \arrow[r, dashed, "m"] & T
\end{tikzcd}$$

Figure 3.4: Definition of *pushout*.

Information Flow Framework (IFF)

Kent [45] proposed a theoretical framework for ontological structures to support ontology sharing. It is based on the Barwise-Seligman theory of information flow [6]. Kent argues that IFF represents the dynamism and stability of knowledge. The former refers to instance collections, their classification relations, and links between ontologies specified by ontological extension and synonymy (type equivalence); it is formalised with Barwise-Seligman's local logics and their structure-preserving transformations—logic infomorphisms. Stability refers to concept/relation symbols and to constraints specified within ontologies; it is formalised with Barwise-Seligman's regular theories and their structure-preserving transformations—theory interpretations. IFF represents ontologies as logics; and ontology sharing as a specifiable ontology extension hierarchy. An ontology, Kent continues, has a classification relation between instances and concept/relation symbols, and also has a set of constraints modelling the ontology's semantics. In Kent's proposed framework, a community ontology is the basic unit of ontology sharing; community ontologies share terminology and constraints through a common generic ontology that each extends, and these constraints are consensual agreements within those communities. Constraints in generic ontologies are also consensual agreements but across communities.

Kent exploits the central distinction made in channel theory between types—the syntactic elements, like concept and relation names, or logical sentences—and tokens—the semantic elements, like particular instances, or logical models—and its organisation by means of classification tables, in order to formally describe the stability and dynamism of conceptual knowledge organisation. He assumes two basic principles, (1) that a community with a well-defined ontology owns its collection of instances (it controls updates to the collection; it can enforce soundness; it controls access rights to the collection), and (2) that instances of separate communities are linked through the concepts of a (common generic ontology), and then goes on to describe a two-step process that determines the core ontology of community connections capturing the organization of conceptual knowledge across communities (as we depict in the figure below).

The process starts from the assumption that the common generic ontology is specified as a logical theory and that the several participating community ontologies extend the common generic ontology according to theory interpretations (in its traditional sense as consequence-preserving mappings [27], and consists of the following steps: (1) A lifting step from theories to logics that incorporates instances into the picture (proper instances for the community ontologies, and so called formal instances for the generic ontology). (2) A fusion step where the logics (theories + instances) of community ontologies are linked through a core ontology of community connections, which depends on how instances are linked through the concepts of the common generic ontology (see second principle above). Kent's framework is purely theoretical, and no method for implementing his two-step process is given. Kent's main objective with IFF is to provide a meta-level foundation for the development of upper ontologies.

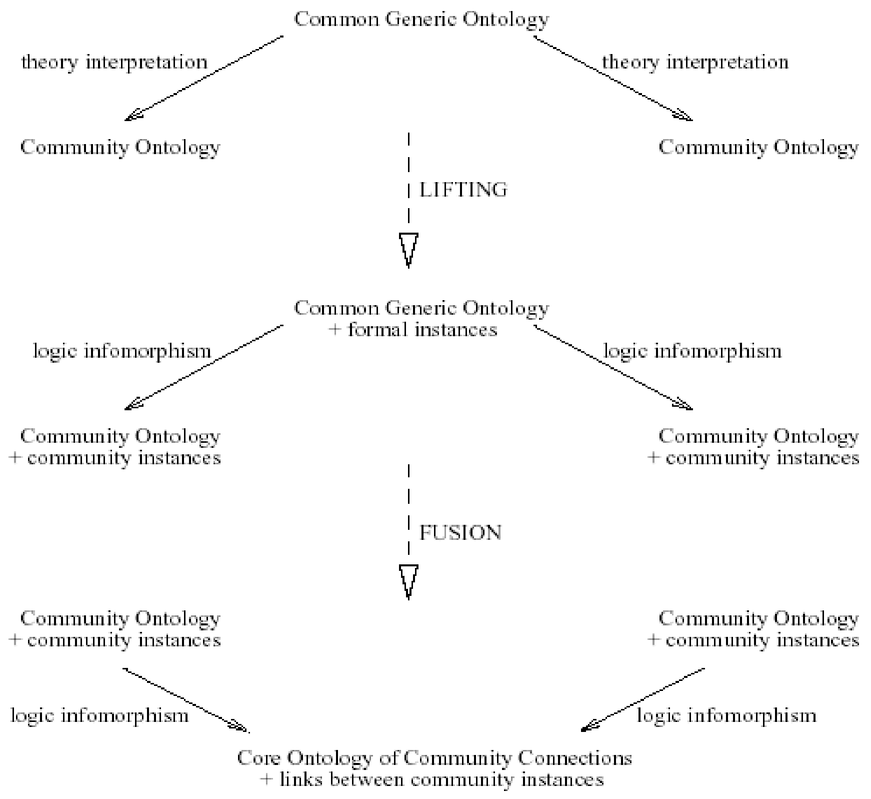

Common Generic Ontology
theory interpretation
theory interpretation
Community Ontology
Community Ontology
LIFTING
Common Generic Ontology + formal instances
logic infomorphism
logic infomorphism
Community Ontology + community instances
Community Ontology + community instances
FUSION
Community Ontology + community instances
Community Ontology + community instances
logic infomorphism
logic infomorphism
Core Ontology of Community Connections + links between community instances

Figure 3.5: Information Flow Framework.

OIS framework

Calvanese and colleagues proposed a formal framework for Ontology Integration Systems—OIS [16]. The framework provides the theory foundation for ontology integration, which is the main focus of their work. Their view of a formal framework is close to that of Kent, and it: "[...] deals with a situation where we have various local ontologies, developed independently from each other, and we are required to build an integrated, global ontology as a mean for extracting information from the local ones."

Ontologies in their framework are expressed as Description Logic (DL) knowledge bases, and mappings between ontologies are expressed through suitable mechanisms based on queries. Although the framework does not make explicit any of the mechanisms proposed, they are employing the notion of queries, which: "[...] allow for mapping a concept in one ontology into a view, i.e., a query, over the other ontologies, which acquires the relevant information by navigating and aggregating several concepts."

They propose two approaches to realize this query/view based mapping: global-centric and local-centric. The global-centric approach is an adaptation of most data integration systems. In such systems, the authors continue, sources are databases, the global ontology is actually a database schema, and the mapping is specified by associating to each relation in the global schema one relational query over the source relations. In contrast, the local-centric approach requires reformulation of the query in terms of the queries to the local sources.

3.2 Methods

Methods aim at applying a particular technique on solving semantic integration problems. Frequently referenced methods include *similarity flooding*, *coefficient computation*, *graph matching*, *formal concept analysis*, *information flow*, *logic satisfiability (SAT)*, *Description Logic (DL)* based inference, etc.

3.2.1 Linguistic Similarity

Linguistic Similarity is based on the assumption that concepts and properties names representing semantic similarity will have similar syntactic features. A refinement of such techniques enhances the result by also taking into account the lengthy textual descriptions (comments) associated with concepts and properties and/or consulting an external thesaurus, e.g. WordNet. A linguistic matcher usually first normalise the input string of names and/or descriptions via stemming and tokenisation. In the simplest form, the equality of tokens will be obtained and combined to give a score of the equality of the whole string. In a more complicated form, synonyms and hypernyms will be considered based on generic and/or domain-specific thesauri and ontologies. For lengthy descriptions, Information Retrieval (IR) techniques can be applied to compare and score the similarity.

As a basic group of matching techniques, linguistic techniques normally act as the first and initial step to suggest a set of raw mappings that other matchers can work with. Quite a few systems reviewed in this survey invoke linguistic matchers at some stage. For example, PROMPT relies on a linguistic matcher to give the initial suggestions of potential mappings which are refined and updated in later steps. Linguistic-based techniques are also employed in Cupid as the first phase of its matching process. They match individual schema elements based on their names, data types, domains, etc., with the help of a thesaurus for short-forms, acronyms and synonyms.

3.2.2 Similarity flooding

A graph matching algorithm which tries to leverage the influence among matching elements from two schemata is presented in [61]. An arbitrary schema S_n is first transformed into a directed labelled graph. The initial mapping between two schemata is obtained using certain mapping techniques, e.g. a simple string matcher comparing common prefixes and suffixes of literals, which are captured in a Pairwise Connectivity Graph (PCG). Nodes of PCG are elements from $S_1 \times S_2$ denoting a "map pair" from S_1 and S_2 respectively. A PCG edge labelled L means that elements of the "map pairs" are connected with an L edge in the original schema.

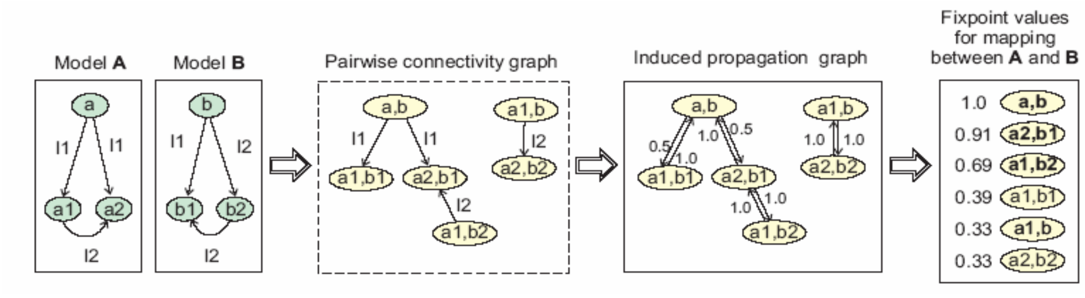


Figure 3.6: Similarity Flooding.

From PCG, the similarity propagation graph is induced that assigns each edge in PCG a propagation coefficient to indicate the influence between nodes in PCG in the given direction. In other words, the weighted edges between "map pairs" indicate how well the similarity of a given map pair propagates to its neighbour and back. The similarity of a "map pair" is then calculated by iteratively accumulating the similarity values of its neighbour "map pair" by taking into account the weight of the edges connecting them. The accumulation of similarity is performed until the "Euclidean length of the residual vector $\Delta(\sigma^n, \sigma^{\text{n-1}})$ is less than for some $n > 0$" or terminated by the user after some maximal number of iterations. The resultant mapping candidates of such a process is referred to as *multimapping*. A series of filter methods are then adopted to reduce the size of *multimapping* and select the most plausible ones. Human verification of the result is sometimes necessary.

3.2.3 Structural similarity

Another technique which is encountered in most of the mapping works, is that of comparing two structures (ontologies or schemata) by taking into account their structural layout, so to speak. That is, when comparing ontologies there is an endemic hierarchical, partially ordered lattice where ontology classes are laid out. Similarly, database schemata use a lattice of connections between tables and classes, not necessarily in a hierarchical fashion though. We deliberately use the notion of structural similarity in a broad sense in order to accommodate many relevant methods that relate to each other, and which could, and sometimes are used in a combined fashion.

The field of similarity metrics is massive and originates, mostly, from the information modelling and retrieval community. A comprehensive, executive summary of those metrics can be found in the Knowledge Web deliverable D.2.2.3 on ontology alignment [58]. Here we recapitulate on some of these methods with emphasis on applications that cross the borders of ontology and database schema based systems.

String-based methods are common to both ontology and database schemata mapping systems. There are variations of string matching algorithms but the idea is to use them in order to compute the similarity of labels for nodes in the structure (of an ontology or a database schema). Some string-based methods go beyond the label to label comparison at particular nodes and take into account an entire path underneath or above the node at question. For example, path comparison used in the COMA system [23].

Moving beyond strings, we see approaches that use more human language oriented technologies, like syntactic and morphological analysis of the nodes' labels. Algorithms which are common in this line of work are stemming algorithms which strip words to their base form by removing suffixes such as plural forms and affixes denoting declension or conjugation. Other systems use external clues, like lexica and dictionaries, to-do analysis of more intrinsic characteristics of the labels, like synonymy, hyponymy and polysemy detection.

These are all used to compare labels at the node level (with some exceptions that use path information) but for structural similarity one has to consider the structure as whole. Typically, algorithms that do structure to structure comparison use the properties found in these structures (transitivity, cardinality, symmetry, etc.) as well as their tree form similarity (for example, similar branches). Other algorithms use information at the nodes other than label, for example, attributes such as datatype, range and domain, etc. These are used then as if they were labels (strings) with the whole range of methods discussed above available for comparing them. In addition to these, most algorithms that compare structures as a whole, use their taxonomic relationships, like part-of, subclass-of, etc. to infer further mappings between nodes in the structure.

3.2.4 Formal Concept Analysis

Formal Concept Analysis (FCA) [32] is a field of mathematics emerged in the nineties that builds upon lattice theory and the work of Ganter and Wille on the mathematization of concept in the eighties. It is mostly suited for analysing objects and attributes of concepts in a domain of interest. Its applications vary across many disciplines but are mostly concentrated in the knowledge acquisition and modelling area. Priss [75] summarizes neatly the basic notions of FCA: "FCA provides a formal framework and vocabulary suited for classification systems, taxonomies, ontologies and hierarchies. Its main unit of study are concepts within a conceptual hierarchy. In the case of type hierarchies, concepts are called "*types*"; in classification systems are called "*classes*"; in ontologies they are called either "*concepts*" or "*types*" or "*classes*"; and in taxonomies are called "*taxa*". Concepts are to be distinguished from objects and attributes, which are usually formal representations of elements of some domain. Objects can also be called *elements, individuals, tokens, instances* or *specimens*. They describe the extensional aspects of concepts. Attributes can also be called *features, characteristics, characters* or *defining elements*. They describe the intentional aspects of concepts. Both intentional and extensional elements can be used separately or together to describe or define concepts. Alternatively, concepts can be defined by establishing relations with other concepts. Thus there are multiple ways of defining, characterizing or comparing concepts. FCA provides a formal framework in which all of these can be combined and applied."

FCA consists of formal contexts and concept lattices. A formal definition of a formal context is as follows [32]: A formal context is a triple $K = (O, P, S)$, where O is a set of objects, P is a set of attributes (or properties), and $S \subseteq O \times P$ is a relation that connects each object o with the attributes satisfied by o. The intent (set of attributes belonging to an object) and the extent (set of objects having these attributes) are also given formal definitions in [32]. A formal concept is then a pair (A, B) consisting of an extent $A \subseteq O$ and an intent $B \subseteq P$, and these concepts are hierarchically ordered by inclusion of their extents. This partial order induces a complete lattice, the concept lattice of the context.

FCA has been used in some of the systems we reviewed in this survey (for example, the FCA-Merge system for ontology merging) and can be applied to semi-structured domains to assist in modelling with objects and attributes in hierarchical, partially ordered lattice. This is then the main structure that most the mapping systems work with. Thus, FCA might not be directly related to mapping, but it's a versatile technology which could be used at the early stages of mapping for structuring a loosely defined domain.

3.2.5 Information Flow

Barwise and Seligman [6] propose a mathematical model that aims at establishing the laws that govern the flow of information. It is a general model that attempts to describe the information flow in any kind of distributed system, ranging form actual physical systems like a flashlight connecting a bulb to a switch to abstract systems such as a mathematical proof connecting premises and hypothesis with inference steps and conclusions.

It is based on the understanding that information flow results from regularities in a distributed system, and that it is by virtue of regularities among the connections that information of some components of a system carries information of other components. The more regularities the system has, the more information flows; the more random the system is constituted the less information will be able to flow among its components. As a notion of a component carrying information about another component, Barwise and Seligman follow the analogy of types and tokens where tokens and its connections carry information. These are classified against types and the theory of information flow captures this aspect of information flow which involves both types and tokens.

With respect to the problem that concerns us here, that of mapping ontologies, the same pattern arises: two communities with different ontologies will be able to share information when they are capable of establishing connections among their tokens in order to infer the relationship among their types. In [43], Kalfoglou and Schorlemmer argued for the relation of information flow to a distributed system like the (Semantic) Web, where the regularities of information flowing between its parts can be captured and used to do ontology mapping. They presented a formalisation of ontology structures where it is possible to express common ontology constructs like classes, relationships, instances as types/tokens pairs to enable information flow based reasoning. The core of their work, the IF-Map system, is presented elsewhere in this section and it aims at capturing the flow of information between ontology constructs that correspond (a.k.a., map into) to each other. The mathematical background of information flow theory ensures that the corresponding types (classes) respect token (instance) membership to each of the mapped types. Their approach is community-oriented, in the sense that communities on the (Semantic) Web own and control their data (instances) and they use them (i.e., classify them) against ontologies for the purpose of knowledge sharing and reuse. It is precisely this information of classifying your own instances against ontologies that is used as evidence for computing the mapping relation between communities' heterogeneous ontologies.

3.2.6 SAT

The idea behind methods in this category is to reduce the matching problem to one that can be solved by resorting to logic satisfiability techniques. Concepts in a hierarchical structure are transformed into well-formed logic formulae (*wffs*). To compute the relationships between two set of *wffs* amounts to examine whether $(\phi, \mathit{wffs}_1, \mathit{wffs}_2)$ is satisfiable. ϕ is the set of relationships normally containing not only equivalence but also complex ones such as "more general than" denoted as $\supseteq$, "less general than" denoted as $\subseteq$, "disjoint with" denoted as $\perp$, etc.

The major difference among these approaches is on how the wffs are computed with respect to each concept (and/or label of concept). Bouquet and colleagues [14] introduce an algorithm with the notions of *label interpretation* and *contextualization*, called Ctx-Match. Each concept in a concept hierarchy is associated with a formula based on the WordNet senses of each word in the label of the concept. The senses associated with each label are refined according to the information provided by its ancestors and direct descendants. Matching of two concepts, C_1 and C_2, is then transformed into checking the satisfiability of a formula composed by contextualised senses associated with their labels and the known WordNet relations among senses expressed in logic formulae, e.g. `art#1` $\subseteq_{\text{WordNet}}$ `humanities#1` denotes that, according to WordNet, the first sense of the word "art" is less general than the first sense of the word "humanities" where "art" and "humanities" are words from the labels of C_1 and C_2, respectively.

S-match [35] goes one step further by distinguishing two different notions of concept, namely the *concept of label* and *the concept of node*. Concept of a label is context insensitive concerning only the WordNet senses of the labels of a concept. On the other hand, concept of a node is context-sensitive, its logic formula is computed as the "intersection of the concepts at labels of all the nodes from the root to the node itself." [35]. The concept of label matrix is constructed containing the relations exist between any two concepts of labels in the two hierarchies of which the matching is to be obtained. Based on such a matrix the concept of node matrix is calculated. We will come back to S-match later in the systems' section.

3.2.7 Machine Learning and Statistic techniques

In recent years, a range of research work proposing machine learning methods for semantic integration has been witnessed. A range of machine learning techniques, e.g. neural network, naïve bayes, etc., have been practically demonstrated to produce promising results. Since in many cases machine learning overlays heavily with statistics and some *match learner* is essentially based on statistic analysis of data, these two techniques are discussed together.

Bayesian approaches have been applied on ontology mapping and schema matching in many forms. Bayesian refers to a family of probability and statistics methods, including, among others, Bayesian probability and Bayesian belief network. Prasad and colleagues [73] calculate Bayesian probability to sort out the best mapping concept B in ontology O_2 for a concept A in ontology O_1. They first compute two raw similarity matrices between concepts from O_1 and O_2 by analysing a set of "exemplar documents" that have already been classified as being associated with each concept and its children. Then Bayesian probability of a leaf node A_j, given a non-leaf node B_i is calculated as

$$P(A_j|B_i) = P(A_j| \vee_k B_k) \forall B_k \in \texttt{children}(B_i)$$

Criteria are set up to filter out the best mapping between concepts O_1 and O_2.

Bayesian Net has recently been applied in ontology mapping. Bayesian belief network is a directed acyclic graph with nodes representing variables and arcs the dependence relations among variables. An arc that goes from node A to node B means that the status of A has non-trivial impact on the status of B. Mitra and colleagues[64] have constructed a Bayesian Net for propagating confidence of mappings along the semantic relationships in source ontologies. Further details of their approach will be given at a later section. Other systems which rely on Bayesian learning techniques include Automatch[13].

An artificial neural network is a mathematical or computational model for information processing. An approximate definition is that a neural network is a series of relatively simple processing elements the sum of which defines the global behaviour. Ehrig and Sure [26] apply neural networks to combine different mapping methods. They construct a three layer neural network consisting of a linear input layer, a hidden layer with a *tan* function, and a *sigmoid* output function.

In some cases, machine learning relies on existing instances and attributes. For instance, GLUE makes a strong assumption that the source ontologies "already have associated data instances". The similarity of two concepts is computed as

$$\textit{Jaccard-sim}(A,B) = P(A \cap B)/P(A \cup B) = \frac{P(A,B)}{P(A,B) + P(A,\bar{B}) + P(\bar{A},B)}$$

Contents and names of an instance are used to compute $P(A|d)$ where d is the set of tokenised words appear in the contents and names of instances. The predications are combined with respect to the whole dataset to calculate the similarity using the above equation.

Matching is sometimes viewed as the mining of association and correlation. He and colleagues [38] proposed an approach based on the observation that grouping attributes which tend to be present jointly in schemata and are statistically positively correlated while synonym attributes rarely co-occur in schemata, therefore represent statistically negative correlation. Hence, the matching problem, in their point of view, is effectively a data mining problem with a strong assumption that the source schemata are inherently similar.

Conditional probability is also at the heart of the method proposed by Wiesman and Roos [87]. The authors focus on the interoperability among communicating agents. Ontologies possessed by different agents are first flattened by converting the attributes of a concept into a canonical format. Instances with associated canonical attributes (denoted as *utterance*) are then passed from the source to the target agent. The instances corresponding to the one received from source agent are identified and the pair of instances is referred to as *joint attention*. Concepts whose instances are present in the *joint attention* are computed as the mapping ones. Probabilistic measures are used when trying to search the corresponding instances of *utterance* in the target ontology and calculating the similarity between concepts from the *joint attention*.

Among other techniques, MOMIS[11] clustering classes into groups at different levels of affinity that forms a hierarchical structure. It starts by treating each class as a cluster and then repetitively merges clusters that have the greatest affinity value. The iteration terminates when only one cluster is left. More details about MOMIS system is given in later sections.

3.2.8 DL-based Approaches

Description Logics (DLs) are a family of knowledge representation and reasoning formalisms [5]. DLs are based on the notions of concepts that are unary predicates and properties that are binary predicates. Using different constructors, composite concepts can be built up from atomic ones. The virtue of DLs is that the semantics of each and every constructor is formally defined based on model theory. With the unambiguous semantics, DLs lend themselves to powerful reasoning algorithms that can automatically classify defined concepts in hierarchical structures, memorised to provide a cache for further reasoning. Typical DL-based reasoning includes concept satisfiability, concept subsumption, instantiation, and concept realisation.

DLs are employed in different occasions in ontology mapping and database schema matching. DL-based languages proposed as the foundation of mapping representation paradigm. Due to their close relation with current *de facto* standard Semantic Web modelling languages, e.g. OWL, the combination of a less expressive DL and Horn Logic is seemed as a good candidate for expressing mappings without resulting in undecidability [21]. The advantage of using DL-based mapping language can also be seen from the reasoning capabilities inherited in DLs. One such example is the Semantic Bridge Ontology (SBO) of the MAFRA framework. SBO is represented in DAML+OIL and specifies a variety of semantic bridges which are organised in a taxonomy. A particular relation between a pair of ontologies is created as an instance in SBO which transform instances from the source ontology to the target one. SBO is illustrated in the Figure 3.7 in UML notation.

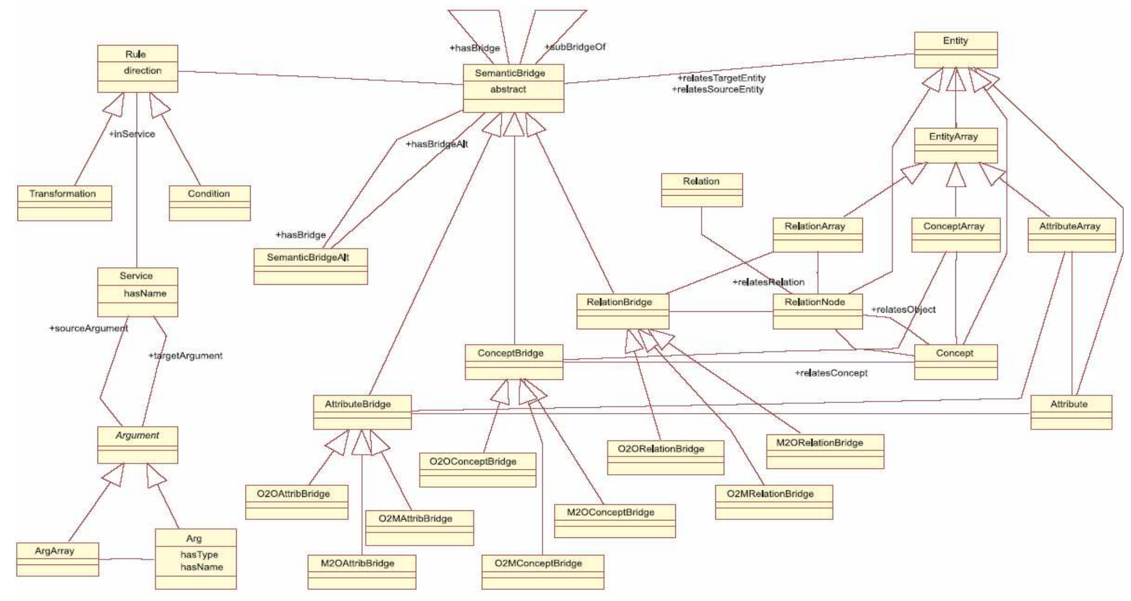


Figure 3.7: Fragment of SBO.

Other DL-based approaches include the OntoMapO[46], a simple meta-ontology for uniformly representing ontologies and mappings between ontologies. The authors argued that relations within an ontology can be considered as a subset of relations that can be used for mapping of concepts in different ontologies.

One measure to simplify the semantic integration problem is to translate the structured or semi-structured heterogeneous data sources into a uniform representation paradigm. DL-based formalisms are frequently used in such a situation. ODL$_{I^3}$ is an object-oriented language with an underlying DL. Inter-schema knowledge can then be represented as terminological relationships, e.g. *Synonym-of*, *Broader*, *Related* whose semantics are captured by translating ODL$_{I^3}$ into OLCD, the underlying DL for leveraging DL-based reasoning. In a similar fashion, Meisel and Compatangelo use $\mathcal{EDDL}_{\mathrm{EER}}$, an enhanced entity-relationship modelling language which can be mapped into DLs to take full advantage of the DL-based reasoning. Mena and colleagues [62] use a DL-based modelling language to define a series of domain-specific ontologies. Queries submitted to their system, OBSERVER, are composed in DL using terms from a user ontology and rewritten via links between domain-specific ontologies and wrapped underlying data sources. Three types of inter-ontology relationships are defined: Synonym (equivalent), Hyponym (subsumed), and Hypernym (subsume) in the DL sense.

Finally, the satisfiability techniques leveraged in CTXMatch and S-Match[81] is essentially a DL-based concept satisfiability problem as DLs are adopted as the underlying logic model for representing WordNet sense composition and WordNet relations.

3.3 Platforms

Platforms are integrated tools. Besides mapping they normally support other crucial functionalities of knowledge management. In the following sections different platforms will be further divided into those that target ontology mapping, database schema matching and general information integration tasks.

3.3.1 Information Integration

DIKE

The DIKE [77] system uses semi-automatic techniques for discovering synonyms, homonyms and object intrusion relationships from database schemata. An algorithm for integrating and abstracting database schemata integrates multiple ER schemata by using the following principle: similarity of schema elements depends on the similarity of elements in their vicinity (nearby elements influence match more than ones farther away). They use LSPD (Lexical Synonymy Property Dictionary) that contains linguistic similarity coefficients between the elements of the two schemata. to quote the authors: “the schemas are interpreted as graphs with entities, relationships, and attributes as nodes. The similarity coefficient of two nodes is initialized to a combination of their LSPD entity, data domain and keyness. This coefficient is re-evaluated based on the similarity of nodes in their corresponding vicinities - nodes further away contribute less. Conflict resolution is also performed on the schemas, e.g. an attribute might be converted to an entity to get a better integrated schema,. The output is an integrated schema, and an abstracted schema.”

Information Manifold

Aiming at taking over from human users the task of how to obtain answers of queries posted to a multitude of structured or unstructured data sources, Information Manifold [50] employs a domain knowledge base in locating information sources needed and re-formulating sub-queries. As the authors argued, their approach to integration is one harvesting fruits from both Artificial Intelligent and Database research areas in that they developed a representation language and a language to meet the requirements on flexibility from the database perspective. This claim is underpinned by an expressive language CARIN, a hybrid language that combines the DLs power of constructing taxonomies with Horn Rule languages.

Using the languages they developed, Levy and colleagues represented various aspects of the domain including physical characteristics of the information sources, their internal structures as well as a rich topic hierarchy. Meanwhile, the contents of each information source are captured and their mappings to the relations in the domain knowledge base are established and represented as CARIN constraints.

When a query is submitted to the Information Manifold system, it first decides what domain knowledge is relevant with respect to the query and identifies the data sources that might be able to contribute to the answers. When a large number of candidate data sources exist, an ordering of accessing the data sources is deduced. The user query is then reformulated into sub-queries and the answers are compiled subsequently.

InfoSleuth

InfoSleuth [28] was a large agents' project in the mid nineties that investigated, among other things, techniques for semantic interoperability between distributed agents. The context in which semantic interoperability was sought after, was the Environmental Data Exchange Network (EDEN), a collaborative effort of US's Environmental Protection Agency (EPA), Department of Defense (DoD), Department of Energy (DoE) and the EU's European Environmental Agency (EEA) to share data.

The mapping is done by resource agents and it is similar: "to the mapping is done traditionally between an internal schema and a conceptual schema in a multi-database", the authors report. InfoSleuth uses mostly linguistic-based techniques to map values of data elements belonging to different schemata. To assist mapping, InfoSleuth developers used mapping ontologies where all the different representations of a value domain were represented, for example: the chemical name "Mercury" has alternative value domain, "Chemical Abstracts Service (CAS)", registry number "7439-97-6", raw CAS number (dashes removed 7439976) and common name "quicksilver".

MOMIS

The MOMIS (MediatOr Environment for Multiple Information Systems (MOMIS)) system [11] which targeted the OntoWeb-1, SIG-1, e-Commerce Product Classification challenge, uses name affinity and structural affinity. It also uses a DL reasoner to exploit constraints. The classes of the input schemata are clustered to obtain global classes from the mediated schema. Linguistic matching is assisted by using WordNet—it accepts schemata as class definitions: it targets, relational, object-oriented and semi structured data (XML). The goal is to generate a global schema. The system that does the schema integration, Artemis, enriches the structural affinity among inter-schema concepts, its a semiautomatic system and does the clustering.

They use four loosely defined levels to do integration: (a) derivation of schema, when schema related information is used; (b) match granularity, when they combine different parameters in computing the matching values; (c) derivation of language, when they use linguistic resources (WorldNet); and (d) auxiliary information, user input. At the terminological level, they work with three relationships: `SYN`, for synonyms; `BT`, for broader terms; `NT`, for narrower terms; and `RT` for related terms.

A key component in the integration process is the generation of a common thesaurus of terminological intentional and extensional relationships describing intra and inter-schema knowledge about classes and attributes of the source schemata. Then MOMIS evaluates affinity between these and clusters together similar ones, that way the virtual schema is created. Manual input is needed when there are polysemy relationships and the user needs to select the right one.

SIMS

The SIMS [4] system aims to create a global schema definition by exploiting the use of DLs. Its an application domain peripheral to semantic integration, it is mostly query planning and planning domain in general. The integration axioms are an application of rewriting rules and techniques. Their goal is query optimization rather than integrating semantically heterogeneous schemata.

TSIMMIS

TSIMMIS [33] is a framework for integrating information. It offers a data model and a common query language that are designed to support integration of information using different sources. This integration is done with a mediator system which sits at the heart of TSIMMIS. These are typical wrappers although they are employed in TSIMMIS in conjunction with other components like:

(1) an Object Exchange Model (OEM): a data model that uses object labels to represent both class information and attributes (instance variables) of objects. This OEM model can represent objects of varying structure, thus tackling the problem of heterogeneous resources;

(2) a proprietary query language, MSL: that allows mediators to choose components for resolving data queries to heterogeneous sources. This language also allows to do query normalization; and

(3) finally, the mediators are also supported by a Lightweight Object REpository (LOREL) DBMS system which manages heterogeneous resources that are accessed by the mediators.

3.3.2 Database Schemata Integration

Automatch

Automatch [13] considers database schema as a finite set of attributes. Therefore, pairwise schema matching is reduced to finding the best matching attributes from the source schemas. Automatch assumes the existence of a predefined reference lexicon, attribute dictionary. Machine learning methods are used to predicate the possibility of attributes from each source schema to the attribute dictionary. A scoring function is responsible for ranking the predications and giving the best matching of attributes. Automatch is a single-strategy system with the ability to give only one-to-one matching.

Autoplex

Autoplex [12] belongs to a family of intelligent information integration tools (e.g. Multiplex, Fusionplex, etc.), that feature machine learning techniques. The idea behind Autoplex is when constructing virtual database systems, features of existing mappings can be learnt and applied to incorporate new data sources when joining the virtual database system.

Machine learning in Autoplex is based on Bayesian techniques with the set of mappings identified by human experts used as the initial training set. Knowledge of the given mappings is learnt using a Bayesian learner that acquires the behaviour of each individual column of the relation schema R from the virtual database and the behaviour of each row of data in the training set with respect to schema R.

When a candidate data source joins the virtual database, its relations are evaluated by the Bayesian classifier. Each column of the schema is examined against that of the virtual relation to compute the optimal overall matching of the candidate relation which is referred to as the projective transformation. Also, data rows of the candidate relation are partitioned into contributing and non-contributing ones. Rules learnt based on this partition are converted into a selection predicate which is referred to as the selective transformation.

Clio

Clio [88] is a research prototype of a schema mapping creation tool. The authors proposed a local query rewriting algorithm without constructing the unified integrated schema. Clio takes user defined value correspondences as inputs and tries to discover the mapping between more than one source schema on one hand and the target schema on the other, i.e. how to compose the source schemata to generate the target view. When multiple mappings are obtained, a ranking mechanism is applied. Queries are generated as results of Clio mapping to facilitate further direct access to source data.

COMA

Do and Rahm [23] proposed a platform to combine several matching techniques. In a single iteration of matching, based on the characteristics of the input schemata, the system invokes several matchers from a *Matcher Library.* Matching scores between elements from the two schemata, e.g. S_1 and S_2, are aggregated as a similarity cube. Matching scores produced by different matchers are combined into a similarity matrix between S_1 and S_2. A filter strategy is then applied to determine the most plausible ones as the resultant scores of the combined matching process.

CUPID

The CUPID [56] approach adopts the following strategy: compute the similarity coefficients between the two schemata and then deduce the mappings using those coefficients. In detail:

- 1st phase, linguistic matching: match individual schema elements based on their names, data types, domains, etc. Use of a thesaurus to help match names by identifying commonly used abbreviations (Qty $\rightarrow$ Quantity) acronyms (UoM $\rightarrow$ UnitOfMeasure) and synonyms (Bill $\rightarrow$ invoice). The result is a linguistic similarity coefficient, this matching includes: normalization, categorization, and comparison; each of these employ a number of popular natural language processing techniques from text engineering such as tokenization, stemming, tagging, elimination, expansion, etc. clustering and similarity measure.

- 2nd phase, structural matching: match schema elements based on the similarity of their contexts or vicinities. For example, *Line* is mapped to *ItemNumber* because they share a common parent, *Item*, and the other two children, *Qty/Quantity* and

UoM/UnitOfMeasure already match. The structural match depends partly on linguistic matches calculated previously. For example, *City* and *Street* under *POBillTo* match *Street* and *City* under *InvoiceTo*, rather than under *DeliverTo* because *Bill* is synonym to *Invoice* but not to *Deliver*. The result is a structural similarity coefficient whose computation algorithm checks first for atomic elements (leaves) that are similar (individually similar - linguistic and Data type) and if their respective vicinities (children, parents) are similar. Then it checks for non-leaf elements (if their subtrees are similar), and non-leaf elements that their immediate children do not match but are still highly similar (leaves represent the atomic data that the schema ultimately describes).

- 3rd phase: a mapping is created by choosing pairs of schema elements with maximal weighted similarity. The authors claim that once mappings are generated then they can be enriched by regarding sub-elements of mapped elements as mapped, e.g, two mapped XML elements will have their attributes (sub-elements) mapped as well. This is similar to the treatment of OWL `SameAs` but there are some practical considerations here with respect to the type of mapping extensions to CUPID: it generates generic schemata, hence, not apply the tree-based algorithms as "real world schemas are rarely trees, since they share sub-structure and have referential constraints", the authors argue. They use XSD of which elements are regarded as XML elements, and they define and use three types of relationships: *containment*, *aggregation* and *isDerivedFrom* (generalization of IS-A and isTypeOf). They also match referential constraints in the sense that they interpret them as potential join views.

Some engineering tuning of algorithm includes: (i) optional vs. required elements (helps with choosing the right nodes for the coefficients' calculations) (ii) initial-mapping (similar idea to kick-off mappings used in IF-Map) used to reduce computation time and complexity (iii) views (where views are defined as referential constraints) (iv) lazy expansions, and (v) pruning leaves, if root and immediate children are the same, the rest is skipped.

DELTA

DELTA (Data Element Tool-Based Analysis) project [19] at MITRE is based on string matching. It looks for metadata and not only use information available at the schema, but also use other common techniques for string parsing (stemming, etc.). An external tool (PersonalLibrarian) is used to do the text classification and search. They then convert the metadata (similar to those processed in the SEMINT work) to a simple text string and then put them together into a document; many documents together about different database attributes metadata form a document base. A human inspects the document and document base and creates search patterns for the attributes she/he wants. These are searched against the document base. Pattern matching is based on string similarity. They also use data dictionaries and table definitions for metadata.

GARLIC

Garlic [36] builds a wrapper-based architecture to describe the local source data using an object oriented language. An exemplar of the authors' mapping scenarios:

> “A DBA wants to add data from a new source to an existing warehouse, the data in the new source may not match the existing warehouse schema. The new data may also be partially redundant with that in the existing warehouse, or formatted differently. Other applications may need to integrate data more dynamically, in response to user queries.
>
> Mapping case: one or more data sets must be mapped into a single target representation - that could include schema transformations, data transformations and cleansing.”

They propose an approach where data and schema transformations are handled in a uniform fashion. Garlic is in principle a database middleware system, not a dedicated schema matching system, per se. Its main component is a query processor which optimises and executes queries over diverse data sources posed in an object-extended SQL. The system interacts with wrappers which do the data transformations from source data to the middleware's model either directly or by constructing views over the middleware's schema.

Garlic wrappers use an abstraction, Garlic objects, and the Garlic Definition Language to describe data from diverse resources into a uniform format, the wrapper's format. This is then used to produce the global schema.

These objects are also used when they construct views as they encapsulated multiple data sources. To a certain extent, these object-based views are creating an articulation (a la ConcepTool[20]) but are used for generating queries.

Although they acknowledge the problem of schematic heterogeneity (data under one schema are represented as metadata in another), they don't tackle it directly. Instead, they have built a semi-automatic tool, Clio, which helps to integrate data and schema transformation processes.

GLUE and iMAP

GLUE [25] is an extension of LSD system [24]. Both are based on machine learning to calculate the similarity between names and contents. Predications from different base learners are then manually weighted (confidence level) and automatically compiled by the meta-learner to give the overall matching result. GLUE is declared to be able to discover semantic matching by leveraging the instances associated with each schema. It is different from the previous LSD system which produces mappings between a local schema and a predefined global one. GLUE can work directly on local source schema. iMAP [22] is the latest development along this direction that has the capability to discover the so-called “complex matching”, i.e. $1 : n$ matching.

Given two ontologies, for each concept in one ontology, GLUE finds the most similar concept in the other ontology by using probabilistic definitions of several practical similarity measures. The authors claim that this is the difference when comparing their work with other machine learning approaches, where only a single similarity measure is used. In addition to this, GLUE also “[...] uses multiple learning strategies, each of which exploits a different type of information either in the data instances or in the taxonomic structure of the ontologies [...]” the authors continue. The similarity measures they employ is the joint probability distribution of the concepts involved, so “[...] instead of committing to a particular definition of similarity, GLUE calculates the joint distribution of the concepts, and lets the application use the joint distribution to compute any suitable similarity measure.”

GLUE uses a multi-learning strategy, the authors continue, because there are many different types of information a learner can glean from the training instances in order to make predictions. It can exploit frequencies of words in the text value of instances, instance names, value formats, or characteristics of value distributions. To cope with this diversity, the authors developed two learners, a content learner and a name learner. The former uses a text classification method, called Naive Bayes learning. The name learner is similar to the content learner but uses the full name of the instance instead of its content. They then developed a meta-learner that combines the predictions of the two learners. To each one of them, assigns a learner weight that indicates how much it trusts its predictions. The authors also used a technique, relaxation labelling, that given a set of constraints, assigns labels to nodes of a graph. This technique is based on the observation that the label of a node is typically influenced by the features of the node's neighbourhood in the graph. The authors applied this technique to map two ontologies' taxonomies, O_1 to O_2, by regarding concepts (nodes) in O_2 as labels, and recasting the problem as finding the best label assignment to concepts (nodes) in O_1, given all knowledge they have about the domain and the two taxonomies. That knowledge can include domain-independent constraints like 'two nodes match if nodes in their neighbourhood also match'—where neighbourhood is defined to be the children, the parents or both—as well as domain-dependent constraints like "if node Y is a descendant of node X, and Y matches professor, then it is unlikely that X matches assistant-professor". The system has been empirically evaluated with mapping two university courses catalogues.

Holistic matching

Holistic matching [38] establishes *n:m* mappings among more than one schema at the same time, i.e. *m:n:k*. A hidden global schema model is assumed by the authors and shared by Web resources describing the same domain. The task is then reduced from matching schemata to discovering the hidden global schema. Data mining technologies are leveraged based on a series of observations made by the authors.

OBSERVER

Mena and colleagues developed the Ontology Based System Enhanced with Relationships for Vocabulary hEterogeneity Resolution (OBSERVER) [62] in order to access heterogeneous, distributed, and independently developed data repositories. Their aim was to tackle the problem of semantic information integration by leveraging the relationships between domain-specific ontologies. They use interontology relationships such as synonyms, hyponyms and hypernyms defined between terms in different ontologies to assist the brokering of information across domain-specific ontologies. Their system is based on a query-expansion strategy where the user poses queries in one ontology's terms and the system tries to expand the query to other ontologies' terms. This is supported by algorithms to manage the relevance of information returned. As far as the mappings are concerned, they use the data structures underlying the domain-specific ontologies and the synonymy, hyponymy and hypernymy relations to inform linguistic matches between concepts.

OntoBuilder

OntoBuilder [30] aims at the reconciliation of databases hiding behind their Web interfaces. As argued by the authors, databases from "deep Web" do not provide access to their underlying schema via URLs but rather via form-based interfaces, e.g. airline reservation interface. Matching schemata based on such limited knowledge are necessary for the shift towards a machine understandable version of the Web. OntoBuilder leverages a compound approach by combining a Name matcher facilitated by WordNet and a Constraint matcher based on the domains of target entities. One-to-one mappings between a pair of schemata are represented as fuzzy relations. The authors also proposed a metric to evaluate the candidate mappings between two schemas so as to find the closest mappings to the exact one which is assumed to be the one given by a human observer.

SEMINT

SEMINT [52] is an instance-based matcher that associates attributes in the two schemata with match signatures - but it does not use structural information - SEMantic INTegrator is a tool that is based on neural networks to assist in identifying attribute correspondences in heterogeneous databases.

In SEMINT they utilise metadata available in DBs for identifying attribute correspondence. Metadata here are viewed as information at three levels: dictionary level (attribute names), field specification level (schema information), data content level (data contents and statistics).

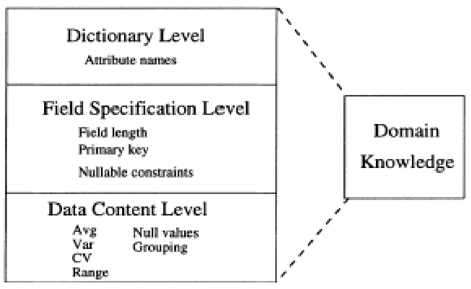
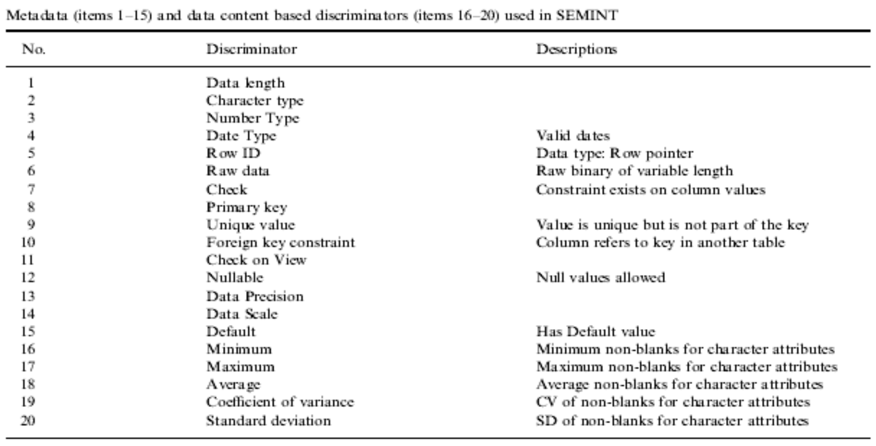


Figure 3.8: Semint metadata levels.

Existing approaches rely on using attributes names for determining correspondences but synonyms occur when objects with different names represent the same concepts and homonyms occur when the names are the same but different concepts are represented (this seems to be the big problem with most DB approaches that rely on lexica, gagdeteers, etc). SEMINT uses neural networks to learn how metadata (schema and constraints information plus data patterns and statistics) characterize the semantics of the attributes in a particular domain. The knowledge of how to determine matching data elements is discovered from the metadata directly, it is not pre-programmed. The output is a similarity metric.

The different sorts of metadata extracted are illustrated in Figure 3.9. The metadata are then normalised and represented in the range [0-1] (a neuron is triggered or not) for the classifier to kick-off (as shown in Figure 3.10).

This representation is easy for true/false values (has or has not a primary key) but for ranges (length of a data field) they use a SIGMOID-like function to normalize numeric ranges. So, for the metadata representation of the attribute book_title, they have a series of values calculated as in Figure 3.11. They then use a classifier to cluster attributes into categories in a single database. They do that with a self-organizing map unsupervised learning algorithm. This algorithm will find the initial clusters to be used for training the network (as shown in Figure 3.12).


Metadata (items 1–15) and data content based discriminators (items 16–20) used in SEMINT

No.	Discriminator	Descriptions
1	Data length	
2	Character type	
3	Number Type	
4	Date Type	Valid dates
5	Row ID	Data type: Row pointer
6	Raw data	Raw binary of variable length
7	Check	Constraint exists on column values
8	Primary key	
9	Unique value	Value is unique but is not part of the key
10	Foreign key constraint	Column refers to key in another table
11	Check on View	
12	Nullable	Null values allowed
13	Data Precision	
14	Data Scale	
15	Default	Has Default value
16	Minimum	Minimum non-blanks for character attributes
17	Maximum	Maximum non-blanks for character attributes
18	Average	Average non-blanks for character attributes
19	Coefficient of variance	CV of non-blanks for character attributes
20	Standard deviation	SD of non-blanks for character attributes

Figure 3.9: SEMINT metadata.

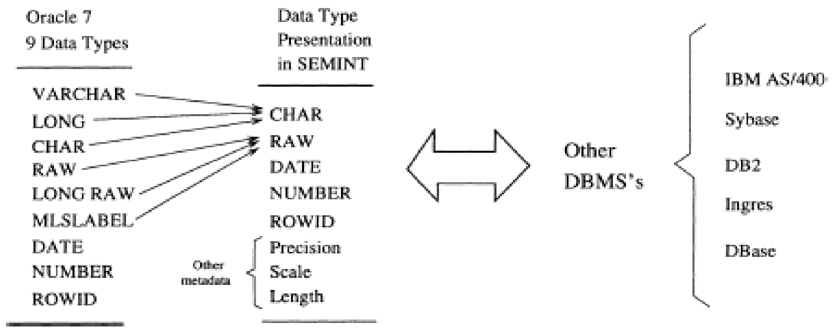


Figure 3.10: SEMINT classifier data.

These clusters are used for training a back-propagation network, using a supervised learning algorithm. This will yield the identified corresponding attributes.

TranScm

TranScm system [63] focuses more on the post-matching process, i.e. how to use the matching to facilitate the translation between schemata. Matching is not the major concern in their work. The authors proposed an approach that: (a) defines a set of matching rules; (b) transforms the source schemata to a graph-based middleware schema representation; and (c) matches two schemata expressed in the middleware representation in a top-down manner starting from the root of different schemata, respectively. This is a manual system and user interactions is deemed necessary.

W3TRANS

Abiteboul and colleagues [1] elaborate on a middleware data model and on declarative rules for integrating heterogeneous data. Although their work is more akin to the database world, their techniques for integration could be useful for ontology mapping. In their data

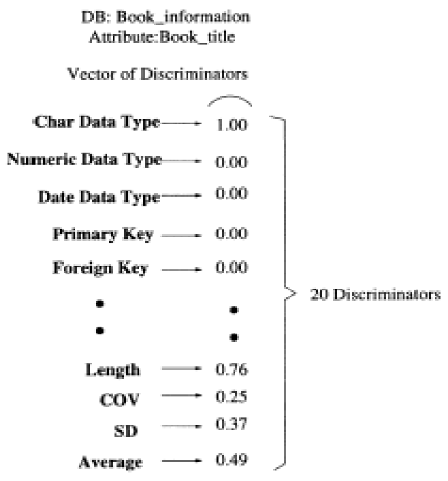


Figure 3.11: SEMINT metadata values.

model, the authors use a structure which consists of ordered labelled trees. The authors claim: "This simple model is general enough to capture the essence of formats we are interested in. Even though a mapping from a richer data model to this model may loose some of the original semantics, the data itself is preserved and the integration with other data models is facilitated."

They then define a language for specifying correspondence rules between data elements and bi-directional data translation. These correspondences could serve for other purposes, for example, as an aid for ontology mapping. These ideas have been implemented in a prototype system, W3TRANS, which uses the middleware data model and the rule language for specifying the correspondences mentioned above.

3.3.3 Ontology Integration

Ontology-oriented Semantic Integration serves as a core function of various ontology management tasks. For instance, it could be used in ontology integration by merging distributed and independently developed ontologies, ontology mapping, version control by means of identifying differences between versions, ontology evolution, ontology population, ontology translation, etc.

Breis and Bejar

Breis and Bejar [15] describe a cooperative framework for integrating ontologies. In particular, they present a system that "[...] could serve as a framework for cooperatively built, integration-derived (i.e., global) ontologies."

Their system is aimed towards ontology integration and is intended for use by normal and expert users. The former are seeking information and provide specific information with regard to their concepts, whereas the latter are integration-derived ontology constructors, in the authors jargon. As the normal users enter information regarding the concepts' attributes, taxonomic relation, and associated terms in the system, the expert users process this information, and the system helps them to derive the integrated ontology. The algorithm that supports this integration is based on taxonomic features and on detection

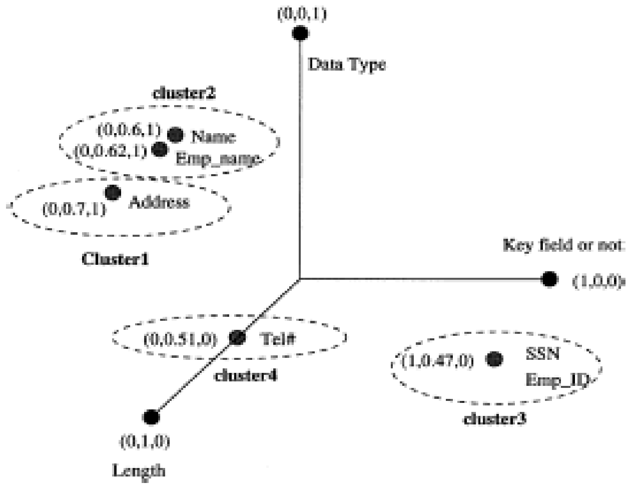


Figure 3.12: SEMINT training clusters.

of synonymous concepts in the two ontologies. It also takes into account the attributes of concepts, and the authors have defined a typology of equality criteria for concepts. For example, when the name-based equality criterion is called upon, both concepts must have the same attributes.

CAIMAN

Lacher and Groh present CAIMAN [47], another system which uses machine-learning for ontology mapping. The authors elaborate on a scenario where members of a community would like to keep their own perspective on a community repository. They continue by arguing that "[...] each member in a community of interest organizes her documents according to her own categorization scheme (ontology)."

This rather weak account of an ontology justifies, to a certain extent, the use of a user's bookmark folder as a "personal" ontology. The mapping task is then to align this ontology with the directory structure of CiteSeer. The use of more formal community ontologies is not supported by the authors, who argue: "Information has to be indexed or categorized in a way that the user can understand and accepts [...] [This] could be achieved by enforcing a standard community ontology, by which all knowledge in the community is organized. However, due to loose coupling of members in a Community of Interest, this will not be possible."

Their mapping mechanism uses machine learning techniques for text classification, it measures the probability that two concepts are corresponding. For each concept node in the *personal* ontology, a corresponding node in the community ontology is identified. It is also assumed that repositories both on the user and on the community side may store the actual documents, as well as links to the physical locations of the documents. CAIMAN is then offering two services to its users: document publication, which publishes documents that a user has newly assigned to one of the concept classes to the corresponding community concept class, and retrieval of related documents, which delivers newly added documents from the community repository to the user.

Chimeara

McGuinness and colleagues [59] developed a similar tool for the Ontolingua editor. As in PROMPT [68], Chimeara, is an interactive tool, and the engineer is in charge of making decisions that will affect the merging process. Chimeara analyses the ontologies to be merged, and if linguistic matches are found, the merge is done automatically. Otherwise the user is prompted for further action. When comparing it with PROMPT, these are quite similar in that they are embedded in ontology editing environments, but they differ in the suggestions they make to their users with regard to the merging steps.

ConcepTool

Compatangelo and Meisel [20] developed a system, ConcepTool, which adopts a description logic approach to formalise a class-centred, enhanced entity relationship model. Their work aims to facilitate knowledge sharing, and ConcepTool is an interactive analysis tool that guides the analyst in aligning two ontologies. These are represented as enhanced entity-relationship models augmented with a description logic reasoner. They also use linguistic and heuristic inferences to compare attributes of concepts in both models, and the analyst is prompted with relevant information to resolve conflicts between overlapping concepts. Their approach is similar to MAFRA's framework in that they both define semantic bridges, as the authors argue: "Overlapping concepts are linked to each other by way of semantic bridges. Each bridge allows the definition of transformation rules to remove the semantic mismatches between these concepts."

The methodology followed when using ConceptTool consists of 6 steps: (1) analysis of both schemata to derive taxonomic links, (2) analysis of both schemata to identify overlapping entities, (3) prompt the analyst to define correspondences between overlapping entities, (4) automatic generation of entities in the articulation schema for every couple of corresponding entities, (5) prompt the analyst for defining mapping between attributes of entities, and (6) analysis of the articulated schema.

FCA-Merge

Stumme and Maedche [83] presented the FCA-Merge method for ontology merging. It is based on Ganter and Wille's work on Formal Concept Analysis [32] and lattice exploration. The authors incorporate natural language techniques in FCA-Merge to derive a lattice of concepts. The lattice is then explored manually by a knowledge engineer who builds the merged ontology with semi-automatic guidance from FCA-Merge. In particular, FCA-Merge works as follows: the input to the method is a set of documents from which concepts and the ontologies to be merged are extracted. These documents should be representative of the domain at question and should be related to the ontologies. They also have to cover all concepts from both ontologies as well as separating them well enough. These strong assumptions have to be met in order to obtain good results from FCA-Merge. As this method relies heavily on the availability of classified instances in the ontologies to be merged, the authors argue that this will not be the case in most ontologies, the authors opt to extract instances from documents: "The extraction of instances from text documents circumvents the problem that in most applications there are no objects which are simultaneously instances of the source ontologies, and which could be used as a basis for identifying similar concepts."

In this respect, the first step of FCA-Merge could be viewed as an ontology population mechanism. This initial step could be skipped, though, if there are shared classified instances in both ontologies. Once the instances are extracted, and the concept lattice is derived, Stumme and Maedche use Formal Concept Analysis techniques to generate the formal context for each ontology. They use lexical analysis to perform, among other things, retrieval of domain-specific information: "It associates single words or composite expressions with a concept from the ontology if a corresponding entry in the domain-specific part of the lexicon exists."

Using this lexical analysis the authors associate complex expressions, like Hotel Schwarzer Adler with concept Hotel. Next, the two formal contexts are merged to generate a pruned concept lattice. This step involves disambiguation (since the two contexts may contain the same concepts) by means of indexing. The computation of the pruned concept lattice is done by an algorithm, TITANIC, which computes formal contexts via their key sets (or minimal generators). In terms of Formal Concept Analysis, the extents of concepts are not computed (these are the documents that they originate from, and are not needed for generating the merged ontology, the authors say), only the intents are taken into account (sets of concepts from the source ontologies). Finally, Stumme and Maedche do not compute the whole concept lattice, "[...] as it would provide too many too specific concepts. We restrict the computation to those formal concepts which are above at least one formal concept generated by an (ontology) concept of the source ontologies."

Having the pruned concept lattice generated, FCA-Merge enters its last phase, the non-automatic construction of the merged ontology, with human interaction. This construction is semi-automatic as it requires background knowledge about the domain. The engineer has to resolve possible conflicts and duplicates, but there is automatic support from FCA-Merge in terms of a query/answering mechanism, which aims to guide and focus the engineer's attention on specific parts of the construction process. A number of heuristics are incorporated in this phase (like using the key sets of concepts for evidence of class membership), and the *is_a* lattice is derived automatically.

IF-Map

Kalfoglou and Schorlemmer [43] developed an automatic method for ontology mapping, IF-Map, based on the Barwise-Seligman theory of information flow [6]. Their method draws on the proven theoretical ground of Barwise and Seligman's channel theory, and provides a systematic and mechanized way for deploying it on a distributed environment to perform ontology mapping among a variety of different ontologies. In the figure below we illustrate IF-Map's underpinning framework for establishing mappings between ontologies. These mappings are formalised in terms of logic infomorphisms. This figure clearly resembles Kent's proposed two-step process for ontology sharing (see [45]), but it has differences in its implementation. The solid rectangular line surrounding Reference ontology, *Local ontology 1*, and *Local ontology 2* denotes the existing ontologies. We assume that *Local ontology 1* and *Local ontology 2* are ontologies used by different communities and populated with their instances, while Reference ontology is an agreed understanding that favours the sharing of knowledge, and is not supposed to be populated. The dashed rectangular line surrounding *Global ontology* denotes an ontology that does not exist yet, but will be constructed "on the fly" for the purpose of merging. This is similar to Kent's virtual ontology of community connections. The solid arrow lines linking Reference ontology with *Local ontology 1* and *Local ontology 2* denote information flowing between these ontologies and are formalized as logic infomorphisms. The dashed arrow lines denote the embedding from *Local ontology 1* and *Local ontology 2* into *Global ontology*. The latter is the sum of the local ontologies modulo Reference ontology and the generated logic infomorphisms.

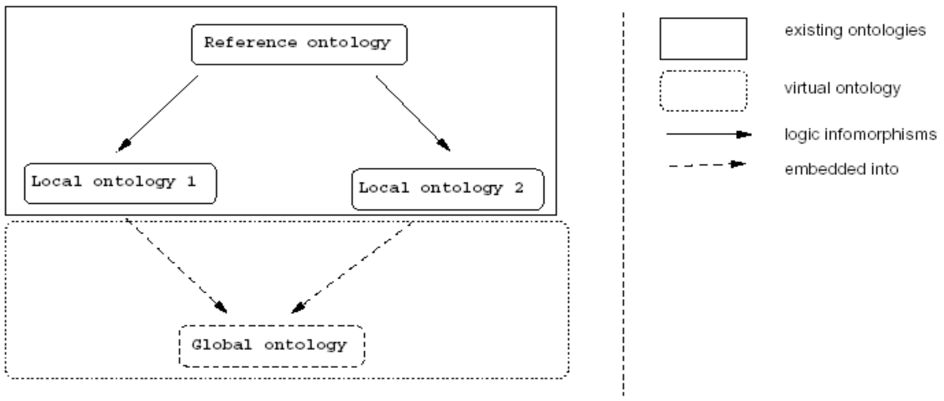


Figure 3.13: Using a global ontology in mapping.

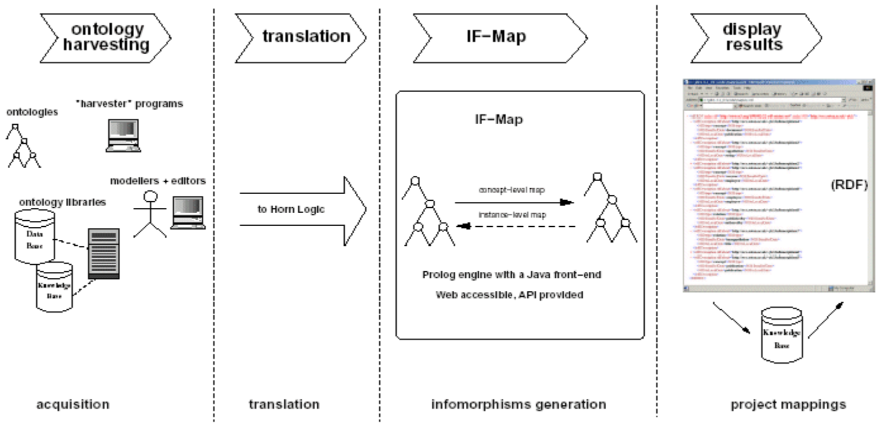


Figure 3.14: Four steps process for ontology mapping in IF-MAP.

In the figure above we illustrate the process of IF-Map. The authors built a step-wise process that consists of four major steps: (a) ontology harvesting, (b) translation, (c) infomorphism generation, and (d) display of results. In the ontology harvesting step, ontology acquisition is performed. They apply a variety of methods: using existing ontologies, downloading them from ontology libraries (for example, from the Ontolingua or WebOnto servers), editing them in ontology editors (for example, in Protégé), or harvesting them from the Web. This versatile ontology acquisition step results in a variety of ontology language formats, ranging from KIF and Ontolingua to OCML, RD, Prolog, and native Protégé knowledge bases. This introduces the second step in their process, that of translation. The authors argue: "As we have declaratively specified the IF-Map method in Horn logic and execute it with the aim of a Prolog engine, we partially translate the above formats to Prolog clauses." Although the translation step is automatic, the authors comment: "We found it practical to write our own translators. We did that to have a partial translation, customised for the purposes of ontology mapping. Furthermore, as it has been reported in a large-scale experiment with publicly available translators, the Prolog code produced is not elegant or even executable." The next step in their process is the main mapping mechanism—the IF-Map method. This step finds logic infomorphisms, if any, between the two ontologies under examination and displays them in RDF format. The authors provide a Java front-end to the Prolog-written IF-Map program so that it can be accessed from the Web, and they are in the process of writing a Java API to enable external calls to it from other systems. Finally, they also store the results in a knowledge base for future reference and maintenance reasons.

ITTalks

Prasad and colleagues [74] presented a mapping mechanism which uses text classification techniques as part of their web-based system for automatic notification of information technology talks (ITTalks). Their system "[...] combines the recently emerging semantic markup language DAML+OIL, the text-based classification technology (for similarity information collection), and Bayesian reasoning (for resolving uncertainty in similarity comparisons)."

They experimented with two hierarchies: the ACM topic ontology and a small ITTalks topic ontology that organises classes of IT related talks in a way that is different from the ACM classification. The text classification technique they use generates scores between concepts in the two ontologies based on their associated exemplar documents. They then use Bayesian subsumption for subsumption checking: "If a foreign concept is partially matched with a majority of children of a concept, then this concept is a better mapping than (and thus subsumes) its children."

An alternative algorithm for subsumption checking, the authors continue, is to take a Bayesian approach that considers the best mapping being the concept that is the lowest in the hierarchy and the posterior probability greater than 0.5.

MAFRA

Maedche and Staab [57] devised a mapping framework for distributed ontologies in the Semantic Web. The authors argue that mapping existing ontologies will be easier than creating a common ontology, because a smaller community is involved in the process. MAFRA is part of a multi-ontology system, and it aims to automatically detect similarities of entities contained in two different department ontologies. Maedche and Staab argue: "Both ontologies must be normalized to a uniform representation, in our case RDF(S), thus eliminating syntax differences and making semantic differences between the source and the target ontology more apparent."

This normalisation process is done by a tool, LIFT, which brings DTDs, XML-Schema and relational databases to the structural level of the ontology. Another interesting contribution of the MAFRA framework is the definition of a semantic bridge. This is a module that establishes correspondences between entities from the source and target ontology based on similarities found between them. All the information regarding the mapping process is accumulated, and populate an ontology of mapping constructs, the so called Semantic Bridge Ontology (SBO). The SBO is in DAML+OIL format, and the authors argue: "One of the goals in specifying the semantic bridge ontology was to maintain and exploit the existent constructs and minimize extra constructs, which could maximize as much as possible the acceptance and understanding by general semantic web tools."

OMEN

Mitra and colleagues [64] proposed the Ontology Mapping ENhancer. It derives mismatches and invalidates existing false matches by taking advantage of a probabilistic method and existing ontology mappings. An ontology is reduced to concepts and properties as a simplified view. In OMEN, a Bayesian Net is constructed based on the mappings defined a priori that are considered as the evidence, and dependencies among different pairs of mappings that are obtained based on a set of meta-rules, which are combined to give the probability distribution of a mappings. A distance value is set up as the threshold to prune the size of the Bayesian Net.

$$P(C_1\,\theta\,C_1',\,x) \wedge P(q = q', 1) \wedge q(C_1, C_2) \wedge q'(C_1', C_2') \Rightarrow P(C_2\,\theta\,C_2', \min(1, x + \delta))$$

The major contribution of OMEN, as argued by the authors, is that they consider the semantic relationships between concepts and properties from the source ontologies as the clue to develop further mappings. The probabilistic refiner, the authors continue, can be complementary to the techniques for automatic or semi-automatic ontology mapping.

ONION

Mitra and Wiederhold [65] developed the Ontology compositION system (ONION) which provides an articulation generator for resolving heterogeneity in different ontologies. The authors argue that ontology merging is inefficient: "A merging approach of creating a unified source is not scalable and is costly [...] One monolithic information source is not feasible due to irresolvable inconsistencies between them that are irrelevant to the application."

They then argue that semantic heterogeneity can be resolved by using articulation rules which express the relationship between two (or more) concepts belonging to the ontologies. Establishing such rules manually, the authors continue, is a very expensive and laborious task; on the other hand, they also claim that full automation is not feasible due to inadequacy of today's natural language processing technology. So, they take into account relationships in defining their articulation rules, but these are limited to subclass_of, part_of, attribute_of, instance_of, and value_of. They also elaborate on a generic relation for heuristic matches: "Match gives a coarse relatedness measure and it is upon to the human expert to then refine it to something more semantic, if such refinement is required by the application."

In their experiments the ontologies used were constructed manually and represent two websites of commercial airlines. The articulation rules were also established manually. However, the authors used a library of heuristic matchers to construct them. Then, a human expert, knowledgeable about the semantics of concepts in both ontologies, validates the suggested matches. Finally, they include a learning component in the system which takes advantage of users' feedback to generate better articulation in the future while articulating similar ontologies. The algorithms used for the actual mapping of concepts are based on linguistic features.

OntoMapO

Kiryakov and colleagues [46] developed a framework for accessing and integrating upper level ontologies. They provide a service that allows a user to import linguistic ontologies onto a Web server, which will then be mapped onto other ontologies. The authors argue for "[...] a uniform representation of the ontologies and the mappings between them, a relatively simple meta-ontology OntoMapO of property types and relation-types should be defined."

Apart from the OntoMapO primitives and design style, which is peripheral to our survey, the authors elaborate on a set of primitives that OntoMapO offers for mapping. There are two sets of primitives defined, *InterOntologyRel* and *IntraOntologyRel*, each of which has a number of relations that aim to capture the correspondence of concepts originating from different Ontologies (i.e., equivalent, more-specific, meta-concept). A typology of these relations is given in the form of a hierarchy and the authors claim that an initial prototype has been used to map parts of the CyC ontology to EuroWordNet.

OntoMorph

Chalupksy [18] developed a translation system for symbolic knowledge—OntoMorph. It provides a powerful language to represent complex syntactic transformations, and it is integrated within the *PowerLoom* knowledge representation system. The author elaborates on criteria for translator systems: "Translation needs to go well beyond syntactic transformations and occurs along many dimensions, such as expressiveness or representation languages, modelling conventions, model coverage and granularity, representation paradigms, inference system bias, etc., and any combination thereof."

OntoMorph uses syntactic rewriting via pattern-directed rewrite rules that allow the concise specification of sentence-level transformations based on pattern matching; and semantic rewriting, which modulates syntactic rewriting via (partial) semantic models and logical inference supported by PowerLoom. OntoMoprh performs knowledge morphing as opposed to translation. To quote Chalupksy: "A common correctness criterion for translation systems is that they preserve semantics, i.e., the meaning of the source and the translation has to be the same. This is not necessarily desirable for our transformation function T, since it should be perfectly admissible to perform abstractions or semantic shifts as part of the translation. For example, one might want to map an ontology about automobiles onto an ontology of documents describing these automobiles. Since this is different from translation in the usual sense, we prefer to use the term knowledge transformation or morphing."

An interesting technique of OntoMorph is semantic rewriting. When, for example, someone is interested in conflating all subclasses of truck occurring in some ontology about vehicles into a single truck class, semantic rewriting allows for using taxonomic relationships to check whether a particular class is a subclass of truck. This is achieved through the connection of OntoMorph with PowerLoom, which accesses the knowledge base to import source sentences representing taxonomic relationships, like subset and superset assertions.

QOM

Ehrig and Staab [26] present their Quick Ontology Mapping (QOM) system. QOM combines previous work by the authors, for example their linguistic-based Naive Ontology Mapping (NOM). The goal of QOM is to optimize the NOM towards efficiency, a.k.a. quality of mappings. The authors argue that there is a trade-off between effectiveness (quality) and efficiency for most mapping generation algorithms. To overcome this trade-off, they produced QOM which, allegedly, had low run-time complexity and it is possible to produce high-quality mappings.

They described a generic 6 steps process for generating mappings which we depict in Figure 3.15:

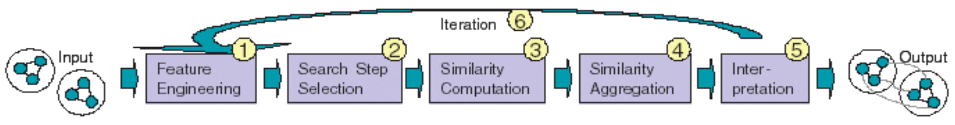


Figure 3.15: Quick Ontology Mapping (QOM) mapping process.

In **step 1**, "feature engineering", transforms the initial representation of ontologies into a format, "digestible for similarity calculations", the authors describe. **Step 2**, searches and chooses to compute similarity in a restricted subset of candidate concept pairs. In **step 3**, we have the similarity computation followed by similarity aggregation in **step 4**. There may be several similarity values for a candidate pair of concepts and these must be aggregated into a single aggregated similarity value. **Step 5**, "Interpretation", uses the individual or aggregated similarity values to derive mappings between concepts. Finally, in **step 6** we have the iteration that some algorithms perform over the whole process in order to bootstrap the amount of structural knowledge. This may stop when no new mappings are proposed. Eventually the mapping is returned as a mapping table.

This six step process guided the development of QOM system with modification which characterizes the QOM. For example, in step 2, QOM optimizes the search space algorithms to lower the number of candidate mappings. Their mechanism uses several heuristics' strategies to restrict the search space of candidate mappings: (1) *Random*, which selects a fixed number or a percentage of all possible mappings; (2) *Label*, which uses concept labels' similarity' (3) *Change propagation*, which explores adjacent concepts to one found in previous iterations as candidate mappings; (4) *Hierarchy*, which explores the hierarchy of concepts in a top down fashion; and (5) *Combination*, which combines all the previous strategies in one step.

SKAT

SKAT [66] represents late work of the algebra-based schema integration techniques for ontology articulation and fusion (Mitra and Wiederhold work). SKAT is more elementary in that it uses concepts' labels for discovering matches and needs the manual input of articulation rules which disambiguate semantic mismatches.

There exists some work on a typology of correspondences, like attributed equivalence [48] from Larson—early work (1986, published in 1989). They define 3 types of attribute equivalence, *strong*, *weak* and *disjoint* (basic equivalence properties), the formal definitions of which use isomorphism and mapping functions. They also define object and relationship equivalence (with more properties like equal, contains, contained-in, overlap and disjoint). Establishing equivalence in objects is based on establishing equivalence in the identifier (key) classes of those objects (which is done as in their basic properties equivalence calculation). Similarly, relationship equivalence uses the identifier attributes and their computed equivalence. Their techniques still don't capture the semantics used, in their example (p.458) they claim that matching the values of attributed SSN and LIC# of one relationship "car-ownership" to the aggregation of the values of the attributes SSN and LIC# from another relationship ("owns" holds over "person", "car") allows us to integrate *car-ownership* with *owns*, which would be inappropriate if there was another pair of concepts with the same attribute values that is linked via a *owns* relationship.

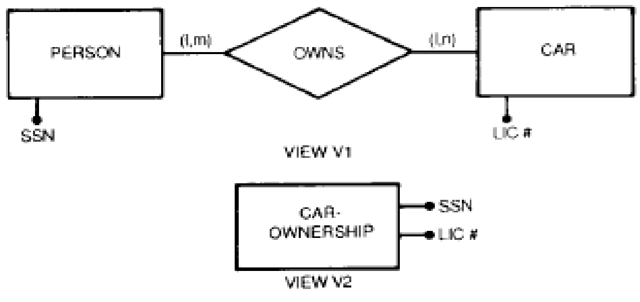


Figure 3.16: Example of SKAT system

SMART, PROMPT and PROMPTDIFF

Noy and Musen have developed a series of tools over the past three years for performing ontology mapping, alignment and versioning. These tools are SMART [67], PROMPT [68] and PROMPTDIFF [69]. They are all available as a plug-in for the open-source ontology editor, Protégé-2000 [34].

The tools use linguistic similarity matches between concepts for initiating the merging or alignment process, and then use the underlying ontological structures of the Protégé-2000 environment (classes, slots, facets) to inform a set of heuristics for identifying further matches between the ontologies. The authors distinguish in their work between the notions of merging and alignment, where merging is defined as "[...] the creation of a single coherent ontology and alignment as establishing links between [ontologies] and allowing the aligned ontologies to reuse information from one another."

The SMART tool is an algorithm that "[...] goes beyond class name matches and looks for linguistically similar class names, studies the structure of relations in the vicinity of recently merged concepts, and matches slot names and slot value types" the authors describe. Some of the tasks for performing merging or alignment, like the initial linguistic similarity matches, can be outsourced and plugged into the PROMPT system by virtue of Protégé-2000's open-source architecture. PROMPT is a (semi-)automatic tool and provides guidance for the engineer throughout the steps performed during merging or alignment: "Where an automatic decision is not possible, the algorithm guides the user to the places in the ontology where his intervention is necessary, suggests possible actions, and determines the conflicts in the ontology and proposes solutions for these conflicts."

Their latest tool, PROMPTDIFF, is an algorithm which integrates different heuristic matchers for comparing ontology versions. The authors combine these matchers in a fixed-point manner, using the results of one matcher as input for others until the matcher produces no more changes. PROMPTDIFF addresses structure-based comparison of ontologies as its comparisons are based on the ontology structure and not their text serialisation, the authors argue. Their algorithm works on two versions of the same ontology and is based on the empirical evidence that a large fraction of frames remains unchanged and that, if two frames have the same type and have the same or very similar name, one is almost certainly an image of the other. All Protégé specific tools from Noy and Musen have been empirically evaluated in a number of experiments using the Protégé ontology editing environment.

S-Match

Guinchiglia and colleagues [35] presented an approach to computing semantic matching between two graph structures, S-Match. They define semantic matching as the process in which semantic correspondences are discovered by computing and returning as a result the semantic information implicitly or explicitly codified in the labels or nodes and arcs of a graph. Their approach, allegedly, differs from previous work in computing similarity coefficients between entities in that it is taking into account information about the concepts involved, not only the labels that characterize them.

They implemented a system, CTX, which illustrates the idea of semantic matching. In figure 3.17 we depict an example case where semantic matching is used:

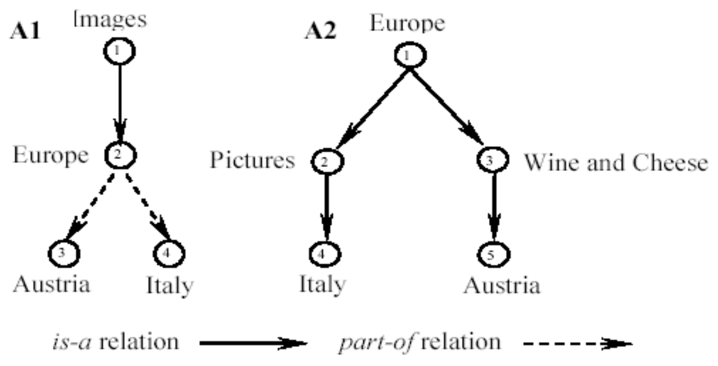


Figure 3.17: An example case of S-Match semantic matching.

One of the characteristics about S-Match is that it makes a clear distinction about the semantics of "concept of a node" and "concept at a node". So, for example, in figure 3.17, "Images" does not mean images but "documents which are (about) images". In the same fashion, "Europe" means documents about Europe, so there is an intended meaning which is being revealed here. The authors also devised a typology of semantic relations, like, *equivalence*, *less general*, *more general*, *overlapping*, *mismatch* for which logic-drawn operators are available. Semantic matching is then an algorithm that given two graphs, computes the mapping elements that have the strongest semantic relationship between nodes of those concepts. As opposed to other syntactic matching approaches, the mappings produced by S-Match are using the typology given above and not numerical values in the range [0-1].

Chapter 4

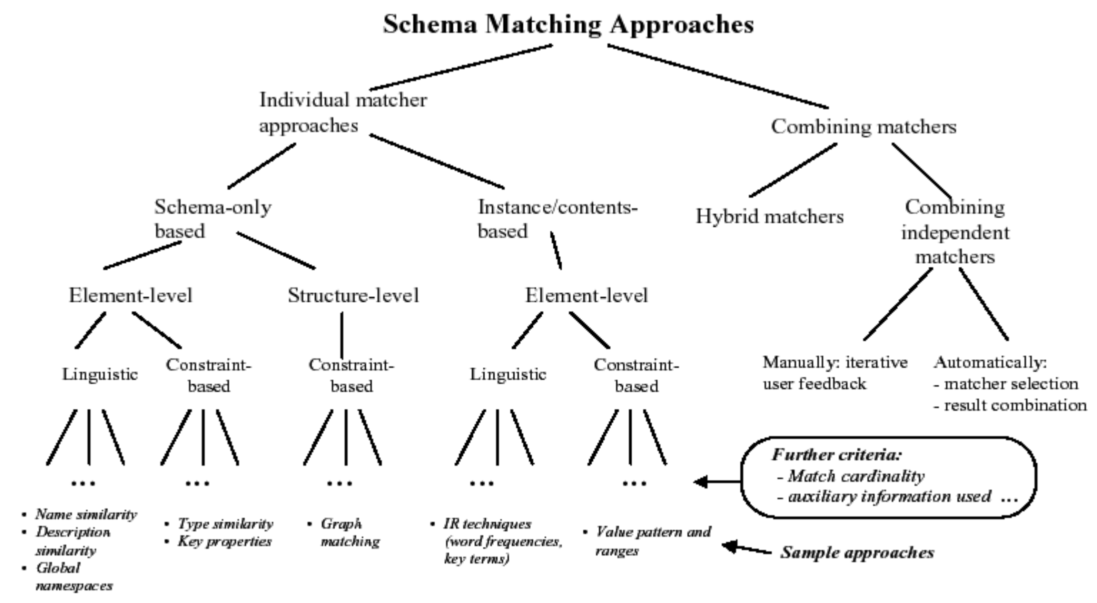
Typologies and classification

In this section we categorise the works presented in the previous section in concise typologies and classifications. We report on those that have been proposed in the literature either explicitly or implicitly. We also reflect and argue for the important of such classification schemes.

4.1 Typology of matching and mapping techniques

4.1.1 Rahm and Bernstein typology

One of the most complete and well cited typology was proposed by Rahm and Bernstein [76]. It targets database schema matching approaches, however, as we will see in the sequel, some researchers have use and revise this typology for ontology-based mapping systems. The typology is depicted diagrammatically in the following figure:



Schema Matching Approaches
Individual matcher approaches
Combining matchers
Schema-only based
Instance/contents-based
Hybrid matchers
Combining independent matchers
Element-level
Structure-level
Element-level
Linguistic
Constraint-based
Constraint-based
Linguistic
Constraint-based
Manually: iterative user feedback
Automatically: - matcher selection - result combination
Further criteria: - Match cardinality - auxiliary information used ...
• Name similarity • Description similarity • Global namespaces
• Type similarity • Key properties
• Graph matching
• IR techniques (word frequencies, key terms)
• Value pattern and ranges
Sample approaches

Figure 4.1: Typology of schema matching approaches by Rahm and Bernstein.

A top level distinction in this typology is *individual* versus *combinational* matcher approaches. An individual matcher uses a single algorithm to perform the match. A combinational matcher, on the other hand ,can be hybrid which uses multiple criteria to perform the matching, or composite, which runs independent matchers and then combine their results.

The individual matchers branch further distinguishes between: (a) *schema-only* based matchers and (b) *instance* based matchers. The former considers only schema information, not instance data. Schema information includes names, descriptions, relationships, constraints, etc. Instance based matchers use metadata and statistics collected from data instances to annotate the schema, or directly find correlated schema elements, e.g., by using machine learning.

The next level in this hierarchy distinguishes between *element* and *structure* with variable degrees of granularity. Element level granularity computes a mapping between individual schema elements, for example, an attribute matcher. Structure level granularity compares combinations of elements that appear together in a schema, for instance, classes or tables whose attribute sets only match approximately.

The next level down in the hierarchy includes *linguistic* and *constraint* based matchers. Linguistic based matchers use names of schema elements and other textual descriptions. Name matching involves putting the name into a canonical form by stemming and tokenization, comparing quality of names, comparing synonyms and hypernyms using generic and domain-specific thesauri and matching sub-strings. Information retrieval techniques can be used to compare descriptions that annotate some schema elements. Constraint-based matchers use schema constraints such as data types and value ranges, uniqueness, requiredness, cardinalities, etc. It might also use intraschema relationships such as referential integrity.

Rahm and Bernstein include more criteria at this stage which could be used to further distinguish between matchers: (i) *matching cardinality* where most matchers produce a 1:1 mappings between schema elements. Others produce n:1 mappings, e.g., dailyWages and workingDays to MonhtlyPay; (ii) *auxiliary information* with tools such as dictionaries, thesauri and input match-mismatch information. Reusing past match information can also help when computing mappings that are compositions of previously computed mappings.

4.1.2 Do and Rahm revision

A revision of this work was presented by Do and Rahm [39]. The authors enhanced existing branches in the hierarchy and added more details in some of the nodes. A fragment of the revised hierarchy is depicted diagrammatically in figure 4.2.

In particular, they elaborate on the schema based matchers with respect to their granularity: element-level matchers match single schema elements or leaf elements with inner elements. Structure-level matchers, on the other hand, match relationships between structures by combining, for example, schema elements.

Within element-level matchers, the authors elaborate in detail about linguistic matchers: these include syntactical approaches which approximate string matching (e.g., N-grams, edit distance, soundex techniques) and semantic approaches where we have terminology relationships (synonyms vs. hypernyms and/or hyponyms), and special purpose or general purpose thesauri (like WordNet). The constraint based matchers consider data types, key constraints, uniqueness and relationship types.

The instance based matchers branch deals with characterisation of instances. At the element-level matchers we see linguistic methods such as extracting keywords, themes

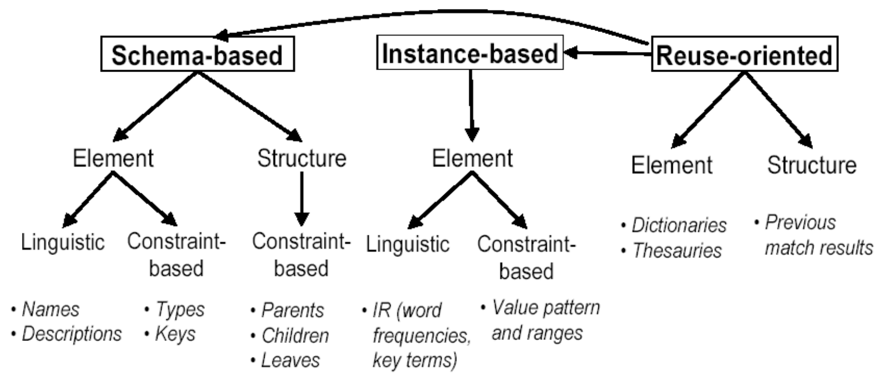


Figure 4.2: Revised typology of schema matching approaches by Do and Rahm.

based on word frequencies, and word combinations. There are also constraint based techniques where we see methods for computing value ranges, character patterns, and distribution of values (e.g., min, max, average and variance). A notable addition to their hierarchy is the inclusion of machine learning techniques. These are used for training and based on manually specified matches. Matching is based on prediction of the most similar element.

Another category that was added in this revised version is that of reuse oriented matchers. These use a series of auxiliary sources like general or domain specific thesauri and dictionaries, user specified synonym tables and previously determined match results.

4.1.3 Shvaiko's revision

Another revision of the Rahm and Bernstein typology was given by Shvaiko in [81]. His revision is depicted diagrammatically in figure 4.3.

The main difference with the previous typology is the introduction of two new branches: *heuristic* vs. *formal* techniques and *implicit* vs. *explicit*. The author argues that "matching techniques can have either heuristic or formal ground. The key characteristic of the heuristic techniques is that they try to guess relations which may hold between similar labels or graph structures [...] formal techniques is that they have model-theoretic semantics which is used to justify their claims." [80]. The distinction between implicit and explicit is driven by the codification, or not, of semantic information. The author argues that implicit techniques are syntax driven whereas explicit techniques exploit the semantics of labels and they could use tools which explicitly codify semantic information (thesauri, ontologies, etc.).

Other notable differences is the additional detail in some of the nodes. For example, auxiliary information in the explicit element-level matchers could include domain ontologies, pre-compiled thesauri and lexica. Pre-compiled thesauri provide information for synonyms, hypernyms and other related relationships. Lexica used to obtain the meaning of terms.

In the implicit structure-level matchers branch, the author includes string based approaches like name similarity. These matchers build long labels by concatenating all labels at a given node in a path into a single string. This is then processed and analyzed with

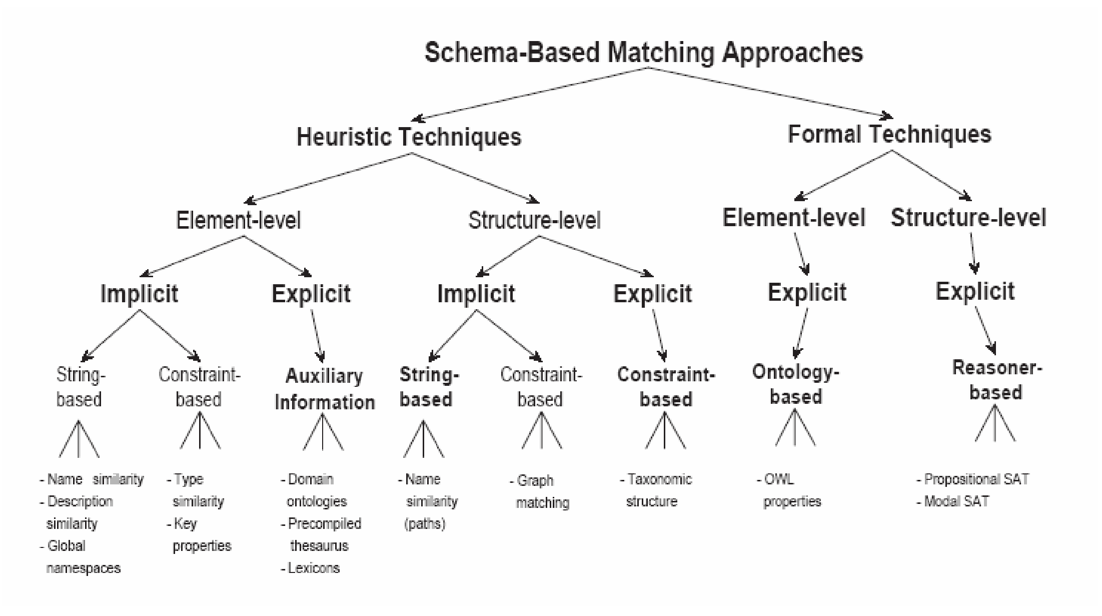


Figure 4.3: Revised typology of schema matching approaches by Shvaiko.

string based techniques.

In the formal techniques branch, the author introduces ontology based techniques for explicit element-level matchers. He argues that formal ontologies encoded in OWL, for example, and their properties could be exploited to help with matching elements. Build-in constructs like the ⟨`owl:sameAs`⟩ could help automate and represent matching elements.

Finally, another category introduced by the author, is the extension of the explicit structure-level matchers. These now have reasoner based approaches, like propositional SAT solvers and model based SAT. The author argues that in propositional satisfiability approaches, the aim is to translate the matching problem into a propositional formula and then to check it for its validity. The matching problem could include the two graphs and mapping queries. These are a pair of nodes and a possible semantic relationship between them. Model based SAT approaches aim to delimit propositional SAT which allows handling only unary predicates (e.g., classes) by admitting binary predicates (e.g., attributes).

4.2 Classification of ontology mismatches

Another area which has seen analyzes in the past is the different types of ontology mismatches. This sort of work is useful when selecting the mapping strategy and system to apply to a mapping case scenario. Similar work in the database world has been published in the early nineties by Sheth and Larson [80] although their analysis on mismatches is not as extensive as the ones we report here.

4.2.1 Visser and colleagues' mismatches

In their work of resolving agent communication problems, Visser and colleagues worked on a classification to characterise ontology mismatches. Their main distinction is between *conceptualization* and *explication* mismatches.

In the former the authors refer to matches between two (or more) conceptualizations of a domain. There we can have *class mismatch* which is related with distinguished classes in the conceptualization. The authors identified two cases in this category: (i) *categorization* mismatch, when two conceptualizations distinguish the same class but divide it into different subclasses and (ii) *aggregation-level* mismatch, when the two conceptualizations both recognise the existence of a class but define it at a different level of abstraction.

Another commonly occurred case is relation mismatch which refers to distinguished relationships in the conceptualization. The authors identify three cases here: (i) *structure* mismatch, when two conceptualizations distinguish the same set of classes but differ in the way these classes are structured by means of relations, (ii) *attribute-assignment* mismatch, when two conceptualisations differ in the way they assign an attribute (class) to other classes, and (iii) *attribute-type* mismatch, when two conceptualisations distinguish the same (attribute) class but differ in their assumed instantiations.

The second main category is that of *explication* mismatch. It deals with the way the conceptualisation is specified. The authors identify a number of mismatch cases related to this: *concept and term*, *concept and definiens*, and *concept to concept.* Concept mismatch refers to cases when both ontologies have the same term and definiens but differ in the concept they define. Term and definiens mismatch is about two ontologies that define the same concept but differ in the way they define it; both with respect to the term and the definiens.

4.2.2 de Bruijn and colleagues' mismatches

In similar fashion, de Bruijn and colleagues [21] worked on a typology of mismatches commonly occur when merging and aligning: *ontology-level* mismatches and the *language-level* mismatches. The former include mismatches in the meaning or encoding of concepts in different ontologies. These are classified using Visser and colleagues' typology, namely conceptualisation and explication, but with different instantiations. They introduce notions such as *Scope* mismatch where two classes (concepts) have some overlay in their extension but the extensions are not exactly the same.

In explication mismatches category they introduce *style of modelling mismatch* which includes paradigm, and *concept description* which is about mismatches in the way a concept is described. Other kinds of mismatch in the explication category are terminological mismatches(for example *synonym* vs. *homonym*) terms and encoding values in different ontologies that might be encoded differently.

Another category is concerned with language-level mismatches which deals with cases when the source and target ontologies are not represented in the same language. There the mismatch has to do with syntax, logical representation, semantics of primitives when they are syntactically equivalent but have a different meaning attached, and language expressivity

4.3 Other categorizations

4.3.1 Kalfoglou and Schorlemmer's categorization

In a different style, and for a different purpose, Kalfoglou and Schorlemmer [44] devised a categorization of ontology mapping systems to classify work done by diverse communities. These categories are not by any means standard, but merely identify the diversity of work. In addition, some of them belong to more than one category. In such a case, they include the cited work in both categories with emphasis given on its primary category. The categories were as follows:

- *Frameworks*: These are mostly a combination of tools, they provide a methodological approach to mapping, and some of them are also based on theoretical work.
- *Methods and tools*: These are either stand-alone or embedded in ontology development environments, and methods used in ontology mapping.
- *Translators*: Although these works might be seen as peripheral to ontology mapping, they are mostly used at the early phases of ontology mapping.
- *Mediators*: Likewise, mediators could be seen as peripheral, but they provide some useful insights on algorithmic issues for mapping programs.
- *Techniques*: This is similar to methods and tools, but not so elaborated or directly connected with mapping.
- *Experience reports*: survey reports on doing large-scale ontology mapping, as it provides a first-hand experience on issues of scalability and of resources involved.
- *Theoretical frameworks*: This is theoretical work that has not been exploited yet by ontology mapping practitioners.
- *Surveys*: This is similar to experience reports but they are more comparative in style.

4.4 Semantic Intensity Spectrum

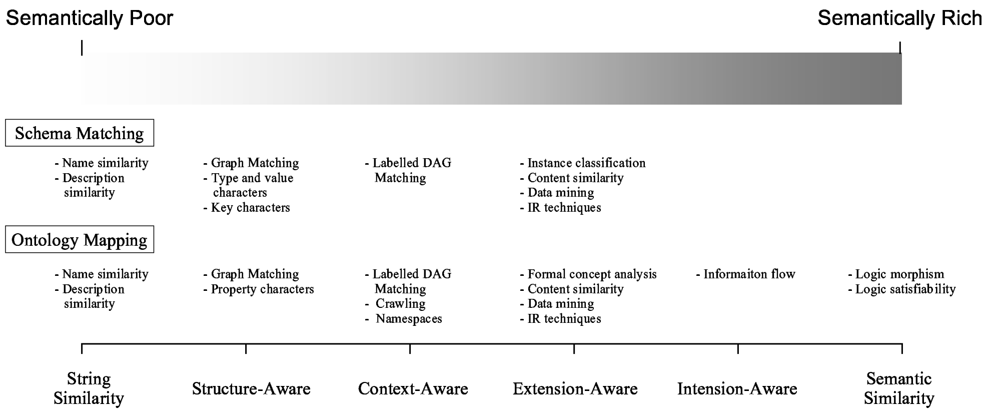


Figure 4.4: Semantic Intensity Spectrum.

We observe a common trend for DB and AI semantic integration practitioners to progress from semantically-poor to semantically-rich solutions, so to speak. We therefore, use this metaphor of semantic richness to classify works from both communities along a *semantic intensity spectrum.* Along this spectrum, we mark several interim points to address string similarity, structure, context, extension and intension awareness as different layers of semantic intensity (see Figure 4.4).

String similarity, occupying the semantically-poor end of the spectrum, compares names of elements from different semantic models. A refinement of such techniques enhances the result by also taking into account the lengthy textual descriptions (a.k.a., comments) associated with concepts and properties. These techniques are based on the assumption that concepts and properties names representing semantic similarity will have similar syntactic features. A string matcher usually first normalises the input string of names and/or descriptions via stemming and tokenisation. In the simplest form, the equality of tokens will be obtained and combined to give a score of the equality for the whole string. In a slightly more complicated form, similarity of two strings is computed by evaluating their substrings, edit distance, etc. Nowadays, pure string similarity measures are seldom used in practice, but rather in combination with external resources, like user-defined lexica and/or dictionaries.

Linguistic Similarity, at a position very close to the semantically-poor end, is an example of string similarity measures blended with some sense of semantics. For instance, pronunciation and soundex are taken into account to enhance the similarity purely based on strings. Also, synonyms and hypernyms will be considered based on generic and/or domain-specific thesauri, e.g. WordNet, Dublin Core. In many cases, user-defined name matches are often treated as useful resources. For lengthy descriptions, Information Retrieval (IR) techniques can be applied to compare and score similarities.

As a basic group of matching techniques, linguistics usually are the initial step to suggest a set of raw mappings that other matchers can work with. Many systems invoke linguistic matchers at some stage: PROMPT [68] relies on a linguistic matcher to give initial suggestions of potential mappings which are then refined and updated in later stages; CUPID [56] employees linguistics at the first phase of its matching process when a thesaurus for short forms, acronyms and synonyms matches individual schema elements based on their names, data types, domains, etc.

Structure-aware, refers to approaches that take into account the structural layout of ontologies and schemata. Going beyond matching names (strings), structural similarity considers the entire underlying structure. That is, when comparing ontologies there is a hierarchical, partially ordered lattice where ontology classes are laid out. Similarly, DB schemata use a lattice of connections between tables and classes, not necessarily in a hierarchical fashion though.

In pure structural matching techniques, ontologies and schemata are transformed into trees with labelled nodes, thus matching is equivalent to matching vertices of the source graph with those of the targeted one. Similarity between two such graphs, G_1 and G_2 is computed by finding a subgraph of G_2 that is *isomorphic* to G_1 or vice versa. Although nodes of such graphs are labelled, their linguistic features rarely play a significant role in computing the similarity. Furthermore, labels of edges are normally ignored with the assumption that only one type of relation holds between connected nodes. For instance, suppose we have two fragments of e-Commerce schemata, one describing an arbitrary `Transaction` and the other one a `PurchaseOrder` (see Figure 4.5). Graph's isomorphism then gives us, among other possible mappings: $\{\mathrm{PO} \leftrightarrow \mathrm{PurchaseOrder}, \mathrm{POShipTo} \leftrightarrow \mathrm{Address1}, \mathrm{POBillTo} \leftrightarrow \mathrm{Address2}, \ldots\}$.

Analogous to pure *string similarity* methods, structure matching approaches, such as the one presented in [86] are not common in practice, but they are usually enhanced with other matching techniques. We deliberately use the notion of *structure similarity* in a broad sense in order to accommodate many relevant methods that relate to each other, and which could, and sometimes are used in such a combined fashion.

Typically, algorithms that do structure to structure comparison use the properties

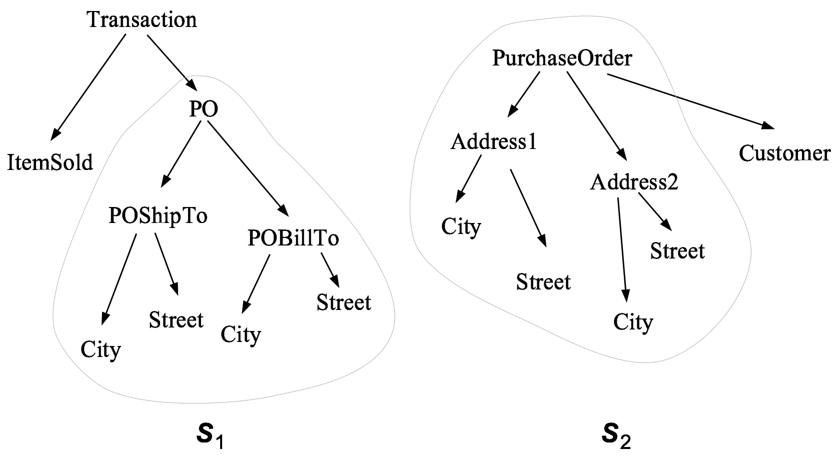


Figure 4.5: Structure Awareness.

found in these structures (transitivity, cardinality, symmetry, etc) as well as their tree form similarity (for example, similar branches). Other algorithms use information at the nodes other than label, for example, attributes such as datatype, range and domain, etc., [63]. These are used as if they were labels (strings) with the range of methods discussed above available for comparing them.

Context-aware, in many cases there are a variety of relations among concepts or schema elements which makes it necessary to differentiate distinct types of connections among nodes. This gives rise to a family of matching techniques which are more semantically rich than *structure similarity* ones.

Both DB schema and ontology can be transferred into a labelled directed graph of which nodes could be elements and concepts, and edges, could be attributes and properties, respectively, with the names of attributes and properties as labels. A *context*, defined in graph jargon, is an arbitrary node together with nodes that are connected to it via particular types of edges which at the same time satisfy certain criteria, e.g., a threshold of the length of paths.

Sometimes, *context-aware* approaches group and weigh the edges from and to a node to impose a view of the domain of discourse from the end user perspective. Depending on whether importing external resources is allowed, there are two types of context-awareness.

In the simplest form, algorithms that compare nodes from two schemata also traverse downwards several layers along the direction of edges from the node under consideration, or upwards against the direction of edges to the node under consideration. All the visited nodes, together with the information about edges connecting them (taxonomic relationships like part-of, subclass-of, etc) are evaluated as a whole to infer further mappings between nodes in the context. For instance, in Figure 4.6(a), the issue whether "Norway" in S_1 corresponds to "Norway" in S_2 is evaluated together with the information provided by their ancestors along the part-of relationship path. It is evident that these two nodes do not match, as "Norway" in S_1 refers to a map of this country while "Norway" in S_2 refers to the country itself.

Similarity flooding [61] is an example of a *context-aware* approach. An arbitrary schema S_n is first transformed into a directed labelled graph. The initial mappings between two schemata, S_1 and S_2, are obtained using certain mapping techniques, e.g., a simple string matcher comparing common prefixes and suffixes of literals, and captured to a Pairwise

Connectivity Graph (PCG). Nodes of a PCG are elements from $S_1 \times S_2$, denoted as $\mathrm{N}_{S_1 \times S_2}$. An edge labelled $\alpha : (m \times k) \to (n \times l)$ ($m, n \in S_1$ and $k, l \in S_2$) of a PCG means that an α edge is present in the original schemata between m and n as well as k and l, i.e. $\alpha : m \to n$ and $\alpha : k \to l$.

From a PCG, a similarity propagation graph is induced which assigns to each edge in the PCG a propagation coefficient to indicate the influence between nodes of the PCG. In other words, the weighted edges indicate how well the similarity of a given PCG node propagates to its neighbour. The accumulation of similarity is performed until a pre-set threshold is reached or terminated by the user after some maximal number of iterations. A series of filter methods are then adopted to reduce the size of the resultant mapping candidates and select the most plausible ones.

Following the same philosophy—similarity propagation, Palopoli and colleagues [**?**] integrates multiple ER schemata by using the following principle: similarity of schema elements depends on the similarity of elements in their vicinity (nearby elements influence match more than those farther away). ER schemata are first transformed into graphs with entities, relationships, and attributes as nodes. The similarity coefficient is initialised by standard thesauruses and re-evaluated based on the similarity of nodes in their corresponding vicinities.

With the use of *namespaces*, along comes another type of *context awareness*. As illustrated in Figure 4.6(b), "UnitedKingdom" belongs to both "World Countries Ontology" and "UK Ontology". Articulating these two ontologies summons the resolution of different namespaces that might involve string matchers in certain forms. An example of dealing with co-reference resolution of such namespaces is given in [3].

Extension-aware, when a relatively complete set of instances can be obtained, semantics of a schema or ontology can be reflected through the way that instances are classified. A major assumption made by techniques belonging to this family is that instances with similar semantics might share features [55], therefore, an understanding of such common features can contribute to an approximate understanding of the semantics.

Formal Concept Analysis (FCA) [32] is a representative of instance-aware approaches. FCA is a field of mathematics emerged in the nineties that builds upon lattice theory and the work of Ganter and Wille on the mathematisation of concept in the eighties. It is mostly suited for analysing instances and properties of entities (concepts) in a domain of interest. FCA consists of formal contexts and concept lattices. A formal context is a triple $\mathrm{K}=(\mathcal{O}, \mathcal{P}, \mathcal{S})$, where $\mathcal{O}$ is a set of objects, $\mathcal{P}$ is a set of attributes (or properties), and $\mathcal{S} \subseteq \mathcal{O} \times \mathcal{P}$ is a relation that connects each object o with the attributes satisfied by o.

The intent (set of attributes belonging to an object) and the extent (set of objects having these attributes) are given formal definitions in [32]. A formal concept is a pair $\langle A, B \rangle$ consisting of an extent $A \subseteq O$ and an intent $B \subseteq P$, and these concepts are hierarchically ordered by inclusion of their extents. This partial order induces a complete lattice, the concept lattice of the context. FCA can be applied to semi-structured domains to assist in modelling with instances and properties in hierarchical, partially ordered lattices. This is the main structure most the mapping systems work with. Thus, FCA albeit not directly related to mapping, it is a versatile technology which could be used at the early stages of mapping for structuring a loosely defined domain.

Intension-aware refers to the family of techniques that establish correlations between relations among extent and intent. Such approaches are particularly useful when it is impossible or impractical to obtain a complete set of instances to reflect the semantics.

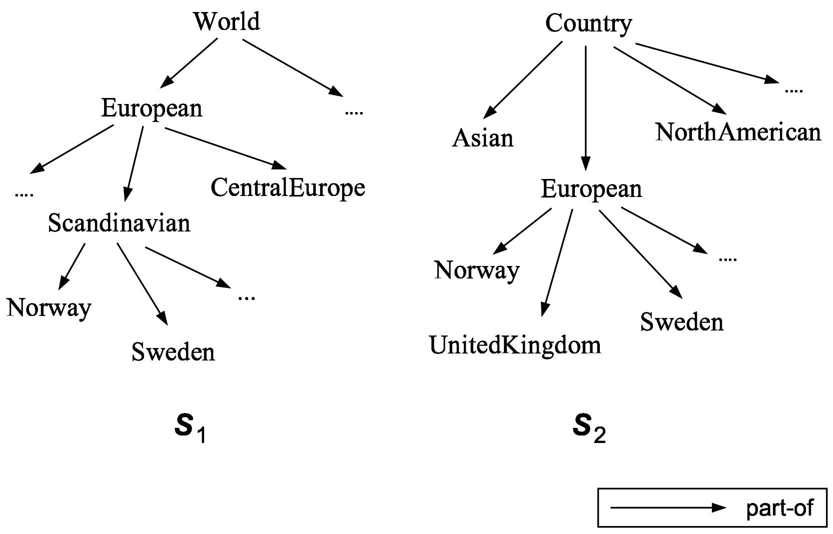
Barwise and Seligman [6] propose a mathematical theory, *Information Flow*, that aims at establishing the laws that govern the flow of information. It is a general theory that attempts to describe information flow in any kind of a distributed system. It is based on the understanding that information flow results from regularities in a distributed system, and that it is by virtue of regularities among the connections that information of some components of a system carries information of other components. As a notion of a component carrying information about another component, Barwise and Seligman follow the analogy of types and tokens where tokens and its connections carry information. These are classified against types and the theory of information flow aims to capture this aspect of information flow which involves both types and tokens.

When integration is our major concern, the same pattern arises: two communities with different ontologies (or schemata) will be able to share information when they are capable of establishing connections among their tokens in order to infer the relationship among their types. Kalfoglou and Schorlemmer [43] argued for the relation of information flow to a distributed system like the (Semantic) Web, where the regularities of information flowing between its parts can be captured and used to do mapping. The mathematical background of information flow theory ensures that the corresponding types (concepts) respect token (instance) membership to each of the mapped types. Their approach is community-oriented, in the sense that communities on the (Semantic) Web own and control their data (instances) and they use them (i.e., classify them) against ontologies for the purpose of knowledge sharing and reuse. It is precisely this information of classifying your own instances against ontologies that is used as evidence for computing the mapping relation between communities' heterogeneous ontologies. It is evident that *information flow* goes beyond *extension-awareness* towards the tick marked by *intension-aware.*

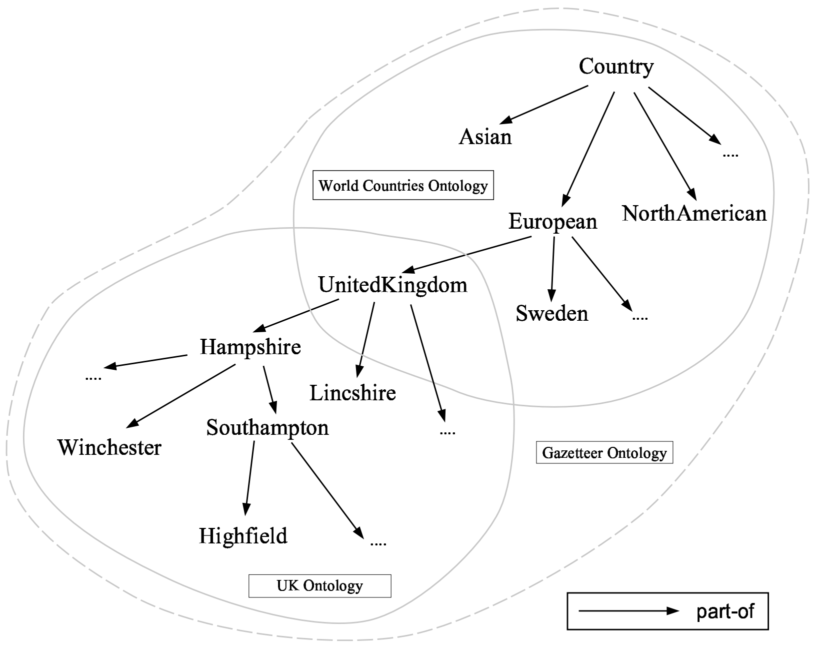
Semantic Similarity, very close to the semantically-rich end lays the family of *logic satisfiability* approaches which focus on the logic correspondences. Logic constructors play a significant role in expressive formalisms, such as DLs, implying that the discovery of similarity is more like finding logic consequence. The idea behind techniques in this category is to reduce the matching problem to one that can be solved by resorting to logic satisfiability techniques. Concepts in a hierarchical structure are transformed into well-formed logic formulae (wffs). To compute the relationships between two set of wffs amounts to examine whether (ψ, *wffs1*, *wffs2*) is satisfiable. ψ is the set of relationships normally containing not only equivalence but also "more general than" denoted as $\supseteq$, "less general than" denoted as $\subseteq$, "disjoint with" denoted as $\perp$, etc.

The major difference among these approaches is on how the wffs are computed with respect to each concept (and/or label of concept). Bouquet and colleagues [14] introduce an algorithm with the notions of *label interpretation* and *contextualization*, called CTX-MATCH. Each concept in a concept hierarchy is associated with a formula based on the WordNet senses of each word in the label of the concept. The senses associated with each label are refined according to the information provided by its ancestors and direct descendants. Matching of two concepts, C_1 and C_2, is then transformed into checking the satisfiability of a formula composed by contextualised senses associated with their labels and the known WordNet relations among senses expressed in logic formulae, e.g. `art#1` $\subseteq_{\text{WordNet}}$ `humanities#1` denotes that, according to WordNet, the first sense of the word "art" is less general than the first sense of the word "humanities" where "art" and "humanities" are words from the labels of C_1 and C_2 respectively.

S-match [35] goes one step further by distinguishing two different notions of concept, namely the *concept of label* and *the concept of node.* Concept of a label is context insensitive concerning only the WordNet senses of the labels of a concept. On the other hand, concept of a node is context-sensitive, its logic formula is computed as the "intersection of the concepts at labels of all the nodes from the root to the node itself." [35]. The concept of label matrix is constructed containing the relations exist between any two concepts of labels in the two hierarchies of which the matching is to be obtained. Based on such a matrix the concept of node matrix is calculated.



(a) "Norway" appears in different contexts



(b) Co-referencing to "UnitedKingdom"

Figure 4.6: Context Awareness

Chapter 5

Mapping criteria, desiderata, and practical issues

We wrap-up this survey with a number of issues that could guide and assist further development for semantic integration. We identify four phases for semantic integration using as a basis an classic survey on database schema matching methodologies. We extend and adopt this to recent technologies (ontologies) and platforms (Semantic Web). We then elaborate on a number of criteria and desiderata that an ideal semantic integration system should possess. Finally, a number of practical issues for consideration by semantic integration practitioners are presented.

5.1 Analyzing semantic integration

5.1.1 Batini and colleagues analysis

Batini and colleagues presented a comparative analysis of database schema matching methodologies[7]. The integration process is divided into several stages where tools and approaches are classified accordingly. Within each stage, methodologies adopted by different tools and/or systems are compared against each other. Although this survey stems from the early days of semantic integration (mid eighties work), it could be adopted to ontology mapping, as we describe in the sequel.

5.1.2 Four Phases of semantic integration

Loosely speaking, in the case of semantic integration in general, a multi-phase (multi-category) approach can be adopted, similar to that proposed by Batini and colleagues. The merit of it, is that it provides the ground for comparison by putting different systems into contexts. The four phases of semantic integration is as follows:

- Pre-integration preparation (a.k.a. normalisation);
- Similarity discovery;
- Similarity representation (also includes reasoning);
- Similarity execution (a.k.a. post-process).

As shown in Figure 5.1, the integration of two *semantic models* is performed in four consecutive phases. The first phase of integration, pre-integration preparation, includes

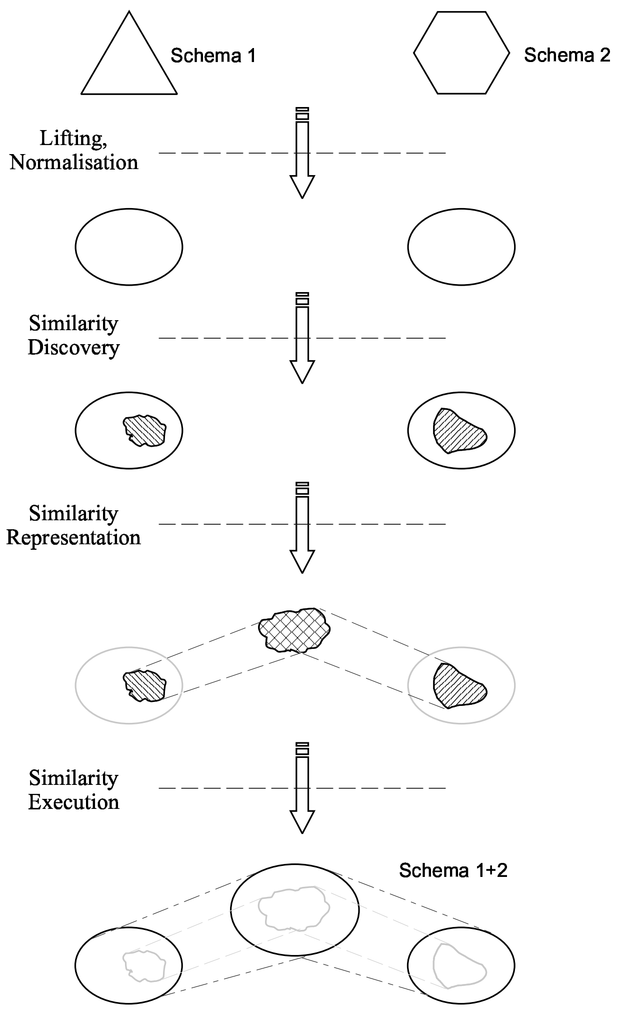


Figure 5.1: Semantic integration phases.

abstract model construction, syntax unification, etc. Source models are normalised and lifted to a uniform representation so that they remove conflicts caused by syntactic heterogeneity. For instance, in the MOMIS system [11], source data are recaptured in ODL$_{I^3}$, an object-oriented language with an underlying Description Logic (DL) [5]; in Information Manifold [50], such a task is carried out by CARIN [51], a language that combines Horn rules and DL. Other issues considered in this phase include the integration strategies, defining the scope of integration, external data gathering (e.g., training data sets, external lexica, etc.). The uniformly represented source models then become the input of *similarity discovery* phase where the correspondences are identified. Other functionalities included in this phase are matching identification, ranking, and confirmation (evaluation).

While some pure mapping and matching systems end at the second phase providing us with a set of correspondences, others proceed to the next phase of integration, *similarity representation*. The choice of representation formalisms for correspondences is critical in automated integration as well as supervised approaches. In early days, correspondences are enumerated in plain text or represented in pairwise fashion in tables for human experts to examine, which is not practical in a large, distributed environment, like the *Semantic Web*. Hence, languages such as XML and RDF [49] start to attract the attention of pracitioners, e.g., the XML output format of MOMIS. One of the purposes of formally representing correspondences is to facilitate *similarity execution*. Depending on the strategies defined in the *pre-integration* phase, the output of *similarity execution* might be a concrete global *semantic model* representing the merge of source models; a virtual global view of source models; a set of articulation rules; and/or a query rewriting formula.

In Table 5.1.2 we present a classification of the systems reviewed in this survey against the four phases of semantic integration. For instance, IF-Map focuses mainly on how to discover similarities between two ontologies and how to represent similarities, e.g. as RDF triples. On the other hand, FCA-Merge addresses only the similarity discovery.

5.2 Criteria and desiderata for semantic integration

By carefully studying existing ontology mapping and database schema matching systems, we identified the following criteria that one could look into when developing a system for semantic integration: *objectives*, *input*, *output*, *automation*, *extensibility*, *complexity* and *scalability*.

5.2.1 Objectives

Semantic integration is a core functionality that lends itself to various applications. A number of themes could be placed under the name semantic integration, ranging from federated database systems to distributed ontology development. Hence, it is beneficial to identify the objectives that a potential semantic integration is meant to serve. We found it useful to consider the following questions:

Q1 *What does the approach want to achieve?*

Tools and systems might be developed for different purposes of which mapping might be the major (or one of) the goals. For instance, *ConcepTool* [60] is mainly for *ontology articulation* whereas *OMEN* [64] is for *refining existing mappings*.

Q2 *At what stage is the tool or method applicable?*

Schema matching and integration consists of different stages, namely, pre-process, mapping, and post-process stage. Different tools or mechanisms may target a different stage of mapping. It is necessary to define different stages of a mapping process *a priori* that largely shapes the overall architecture for integration systems.

In table 5.2 we summarise the major objectives for each system reviewed in this survey. In other words, it focusses on the problem that the system addresses. For instance, IF-Map is mainly for ontology mapping while CUPID is for database schema matching. Note that a system might have multiple major objectives.

5.2.2 Input

The characteristics of input to a semantic integration system rely heavily on the nature of the subject problem. For instance,

- it is essential to understand whether the input semantic models are structured or not, as in the latter case tools are needed to extract structured representation from semi-structured data sources as in MOMIS [10] and Information Manifold [50] systems.

- input semantic models may be represented in different languages, e.g., XML, HTML, RDF, etc. When is it necessary to normalise and build an internal model to do schema translation or schema rewriting?
- to what extend, semantic models are similar to each other? Some approaches follow the assumption that the source schemata are inherently similar, so that heuristics can be applied when computing similarities. For instance, PROMPTDIFF [69] focuses on finding the differences between two versions of an ontology that are assumed to be overlapping.
- to what extend knowledge other than the schemata themselves (or ontologies) is used? Some approaches rely heavily on domain knowledge, context and/or global ontologies.
- how many semantic models can the tools or mechanisms process at one time? Some approaches claim that they can find mappings among more than one source semantic model at one time, e.g., the Holistic Matching algorithm [**?**]

5.2.3 Output

The output of a semantic integration system depends on the intended users. Obviously, human users and machines will have different requirements on the representation of the output. Therefore, the following need to be considered:

Mapping representation: the output mapping pairs are for human readers or automated mediators. Certain representations may facilitate further automated processes (post-mapping processes), e.g,. using the Semantic Web Rules Language (SWRL) [40].

Complex vs. Simple mapping: although the ability of discovering complex (n:m) mapping is seldom discussed in ontology mapping, it becomes a *de facto* criterion in database schema matching to demonstrate the capacity of a matching tool or method.

Ranking mechanism: in most cases, exact mapping (i.e, mapping that is specified by a human observer) cannot be specified. Rather, a series of mappings are given with a corresponding confident level subject to a certain ranking mechanism.

In Table 5.3 we summarise criteria for semantic integration.

5.2.4 Automation

At the early stages of both schema matching and ontology mapping, correspondences are manually crafted by domain experts and/or application engineers [80, 70]. Thus far, fully automated integration is still difficult to achieve and interactions with human and/or external resources are inevitable. To what extent the matching process relies on the input from human observers becomes one of the critical factors. Human intervention may come into the picture at pre-process, similarity-discovery and post-process stages. It is difficult to quantitatively analyze the amount of human effort in the schema matching or ontology mapping process. It is possible, however, to estimate the role played by a human observer, for example, whether human intervention is critical or marginal. Some criteria are enumerated as follows:

1. To what extent, the matching process relies on the existence of external resources. Some approaches rely on a predefined domain ontology, a dictionary or a lexicon to help identify synonyms, hypernyms (subsumers) and hyponyms (subsumees); some require a set of mappings defined by human experts to initiate further actions. The construction of such resources, if not available already, might require substantial effort.

2. Multi-strategy approaches require a mechanism to screen out and/or compile the best matchings. Such a mechanism might have various forms ranging from interaction with end users to supervised semi-automated methods to adaptive and fully automated ones.

3. Machine learning approaches are popular in schema matching. Training is not trivia in such approaches. Hence, how the training set is collected and how the results are interpreted dictate how much the system is automated.

4. Corpus-based approaches require the construction and validation of a corpus that might be the result of a previous matching circle from other integration systems or manually crafted by human experts.

5.2.5 Extensibility

Different systems normally speak different languages in the sense of input/output format and internal representation. Frequently used formats are Rational, XML, SGML, EER, HTML, RDF, OWL, KRR-specific, to name a few. Furthermore, most mapping (or matching) cases apply a variety of similarity computation techniques. For instance, *linguistic matchers* are employed for names, comments and descriptions; *structural matchers* are used for comparing hierarchical structures; and logic expressions are analysed for establishing equivalence. Hence, an ideal semantic integration system should adopt a modular design philosophy facilitating the invocation of multiple matchers and aggregation of multiple matching results. A possible architecture is illustrated in figure 5.2: different features of the input data are generated and selected to fire-off different sorts of matchers. The resultant similarity values are compiled by a multiple similarity aggregator which runs in parallel or consecutive order. The overall similarity is then computed to initiate iterations that backtrack to different stages.

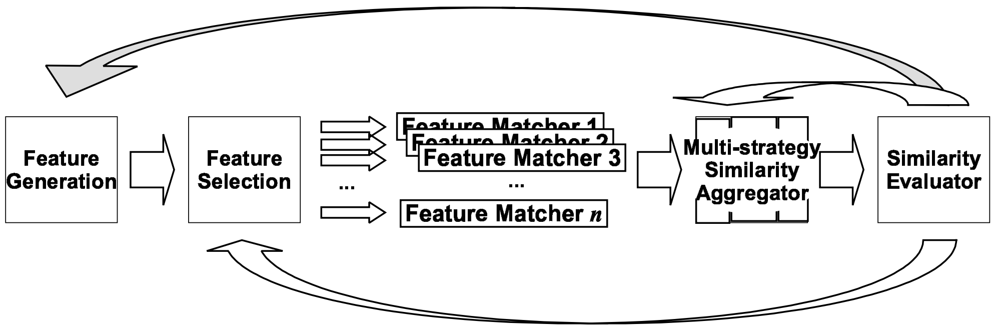


Figure 5.2: A proposed modular architecture.

Standardising the interfaces among modules might serve as a good starting point for increasing the system's extensibility and interoperability.

5.2.6 Complexity and Scalability

Semantic Integration is computationally intensive [9, 71, 80]. Thus far, there is no comparative study of the complexity and scalability of different algorithms. This might be due to the fact that most of the systems are addressing the problem of finding the correct correspondences rather than identifying the optimal solving procedure. In many cases, systems are evaluated against toy examples and purposely-built test cases or adopt strong assumptions where empirical analysis is not adequate to demonstrate the computational complexity and scaling features. We advocate the use of formal aspects such as *soundness*, *completeness*, *consistency*, and practical issues such as *scalability* and *complexity*, in combination with empirical evaluation results.

5.3 Issues

A number of issues remain unclear and could potentially hinder further developments in this fascinating field of research. These are concerned with: (a) the exact and precise meaning of what a mapping is between two concepts, (b) what are the impediments that prevent engineers from producing industry-strength and scalable systems given that there is a long history in trying to tackle this problem (albeit in different domains), and (c) what are the points of convergence and where do works differ in practice.

5.3.1 Semantics of a mapping relationship

The notion of mapping is slippery when it comes to its exact meaning in mathematical terms. There is a variety of informal, semi-formal and formal attempts to characterise it, although most practitioners worry more about mechanisms for finding mappings and less about what it actually means when two elements are mapped. Notions such as *correspondence*, *equivalence*, *same-as*, *similar*, *morphism*, and other variations of natural language are used frequently.

Our focus is on the notion of *equivalence*, which allow us to establish some sort of correspondence between the systems to be mapped (and subsequently their ontologies or database schemata). We look at first-order logic for formally describing equivalence. We also argue for the peculiarities of adopting a first-order logic approach to formalise equivalence. A practical application of this sort of formalisation has been proposed in the Semantic Web realm recently, where we see work regarding the meaning and use of terms that model equivalence between mapped elements.

OWL equivalence semantics

In the W3C recommendation document for the Web Ontology Language (OWL) two different notions of equality between OWL constructs are given.

Identity between Individuals `sameAs` This mechanism is similar to that for classes, but declares two individuals to be identical. An example would be:

$$\begin{aligned}
&\langle Winerdf : ID = "MikesFavoriteWine``\rangle \\
&\quad \langle owl : sameAsrdf : resource = ``\#StGenevieveTexasWhite" / \rangle \\
&\langle / Wine \rangle
\end{aligned}$$

This example does not have great utility. About all we learn from this is that Mike likes an inexpensive local wine. A more typical use of sameAs would be to equate individuals defined in different documents to one another, as part of unifying two ontologies. This brings up an important point. OWL does not have a unique name assumption. Just because two names are different does not mean they refer to different individuals. In the example above, we asserted identity between two distinct names. But it is just as possible for this sort of identity to be inferred. Remember the implications that can be derived from a functional property. Given that hasMaker is functional, the following is not necessarily a conflict.

$$\begin{aligned}&\langle owl : Thingrdf : about = \text{“}\#BancroftChardonnay\text{”}\rangle\\&\quad\langle hasMakerrdf : resource = \text{“}\#Bancroft\text{”}/\rangle\\&\quad\langle hasMakerrdf : resource = \text{“}\#Beringer\text{”}/\rangle\\&\langle /owl : Thing\rangle\end{aligned}$$

Unless this conflicts with other information in our ontology, it simply means that Bancroft = Beringer. Note that using sameAs to equate two classes is not the same as equating them with equivalentClass; instead, it causes the classes to be interpreted as individuals, and is therefore sufficient to categorise an ontology as OWL Full. In OWL Full sameAs may be used to equate anything: a class and an individual, a property and a class, etc., and causes both arguments to be interpreted as individuals.

Equivalence between Classes and Properties: equivalentClass, equivalentProperty

To tie together a set of component ontologies as part of a third it is frequently useful to be able to indicate that a particular class or property in one ontology is equivalent to a class or property in a second ontology. This capability must be used with care. If the combined ontologies are contradictory (all A's are B's vs. all A's are not B's) there will be no extension (no individuals and relations) that satisfies the resulting combination. In the food ontology we want to link wine features in the descriptions of dining courses back to the wine ontology. One way to do this is by defining a class in the food ontology (&food;Wine) and then declaring it equivalent to an existing wine class in the wine ontology.

$$\begin{aligned}&\langle owl : Classrdf : ID = \text{”}Wine\text{”}\rangle\\&\quad\langle owl : equivalentClassrdf : resource = \text{“}\&vin; Wine\text{”}/\rangle\\&\langle /owl : Class\rangle\end{aligned}$$

The property `owl:equivalentClass` is used to indicate that two classes have precisely the same instances. Note that in OWL DL, classes simply denote sets of individuals, and are not individuals themselves. In OWL Full, however, we can use `owl:sameAs` between two classes to indicate that they are identical in every way. Of course the example above is somewhat contrived, since we can always use $\&vin; Wine$ anywhere we would use $\#Wine$ and get the same effect without redefinition. A more likely use would be in a case were we depend on two independently developed ontologies, and note that they use the URI's $O_1 : foo$ and $O_2 : bar$ to reference the same class. owl:equivalentClass could be used to collapse these together so that the entailments from the two ontologies are combined. We have already seen that class expressions can be the targets of `rdfs:subClassOf` constructors. They can also be the target of `owl:equivalentClass`. Again, this avoids the need to contrive names for every class expression and provides a powerful definitional capability based on satisfaction of a property.

$$\begin{array}{l}
\langle owl : Classrdf : ID = \text{“}TexasThings\text{”}\rangle \\
\quad \langle owl : equivalentClass\rangle \\
\quad\quad \langle owl : Restriction\rangle \\
\quad\quad\quad \langle owl : onPropertyrdf : resource = \text{“}\#locatedIn\text{”}/\rangle \\
\quad\quad\quad \langle owl : someValuesFromrdf : resource = \text{“}\#TexasRegion\text{”}/\rangle \\
\quad\quad \langle /owl : Restriction\rangle \\
\quad \langle /owl : equivalentClass\rangle \\
\quad \langle /owl : Class\rangle
\end{array}$$

TexasThings are exactly those things located in the Texas region. The difference betw- een using `owl:equivalentClass` here and using `rdfs:subClassOf` is the difference betw- een a necessary condition and a necessary and sufficient condition. With `subClassOf`, th- ings that are located in Texas are not necessarily TexasThings. But, using `owl:equivalentClass`, if something is located in Texas, then it must be in the class of TexasThings (as shown in Table 5.4).

Relation	Implications
`subClassOf`	TexasThings(x) implies locatedIn(x,y) and TexasRegion(y)
`equivalentClass`	TexasThings(x) implies locatedIn(x,y) and TexasRegion(y)
	locatedIn(x,y) and TexasRegion(y) implies TexasThings(x)

Table 5.4: Difference between `subClassOf` and `equivalentClass`

To tie together properties in a similar fashion, we use `owl:equivalentProperty`.

5.3.2 Engineering mapping systems

Further to the issues of a clear and distinct definition and use of mappings, there are issues related with the practice of mapping systems. We identified drawbacks that prevent engineers from benefiting from such systems. We list them here epigrammatically: (a) the assumptions made when engineer mapping systems are not exposed to the community, (b) mapping systems are often embedded in larger and complex environments where mapping is not their main functionality, (c) most mapping systems are using syntactic clues to determine correspondence, but no semantic information, (d) more formal machinery, like ontological axioms are rarely used in mapping systems, there is also a lack of formal background theory for mapping.

5.3.3 Diversity and convergence

Another observation is that there is an acknowledged diversity in the work on finding mappings but there is also fertile ground for convergence, especially with the advent of technological frameworks where both ontologies and database schemata can coexist, such as the Semantic Web. We summarise some of the key differences and areas for potential convergence.

The key differences are: (a) database schemata do not provide explicit semantics for their data, (b) database schemata are not meant to be sharable or reusable, (c) ontology development is de-centralised as opposed to database schema development, (d) ontology mapping requires handling the formal underpinning that the ontology formalism offers.

The areas of potential convergence are: (a) natural language technologies can benefit both, (b) partial order is a common graph traversal mechanism for both, (c) techniques from database schema matching can be applied to ontology mapping and vice versa, (d) machine learning is applicable to both, (e) the Semantic Web is a common application domain, (f) the notion of Global as View (GAV) and Global-Local as View (GLAV) are associated with top level, global or reference ontologies.

	Pre-Integration Preparation	Similarity Discovery	Similarity Representation	Similarity Execution
Madhavan et. al. system (p.21)			√	
Information Flow Framework (p.24)	√		√	√
OIS framwork (p.26)			√	
DIKE (p.33)		√		
InfoSleuth (p.34)		√		
MOMIS (p.34)		√	√	
SIMS (p.35)				√
TSIMMIS (p.35)	√		√	√
Clio (p.36)		√		√
DELTA (p.37)		√		
TranScm (p.41)				√
Breis and Bejar system (p.42)		√		√
MAFRA (p.47)		√	√	√
OntoMapO (p.48)	√		√	
OntoMorph (p.49)				√
SKAT (p.50)		√	√	√
Automatch (p.35)		√		
Autoplex (p.35)		√		
COMA (p.36)		√		
CUPID (p.36)		√	√	
GLUE and iMAP (p.38)		√		
Holistic Matching (p.39)		√		
OBSERVER (p.39)		√		√
OntoBuilder (p.39)	√	√	√	
SEMINT (p.40)		√		
W3TRANS (p.41)			√	
CAIMAN (p.43)		√		
Chimeara (p.44)		√		
ConcepTool (p.44)	√	√	√	√
FCA-Merge (p.44)		√		
Information Manifold (p.33)			√	√
IF-Map (p.45)		√	√	
ITTalks (p.47)		√		
ONION (p.48)		√	√	
QOM (p.49)	√	√		√
SMART, PROMPT PROMPTDIFF$^{(p.51)}$		√		
S-Match$^{(p.52)}$		√		
OMEN (p.48)		√		√

Table 5.1: Classification of systems into four phases of semantic integration.

	Database Schema				Ontology				Information Integration
	Matching	Integration	Rewriting	Translation	Mapping	Translation	Articulation	Integration	
Madhavan *et.al.* system$^{(p.21)}$					✓				
Information Flow Framework$^{(p.24)}$					✓				
OIS framwork$^{(p.26)}$							✓		
DIKE$^{(p.33)}$		✓							
InfoSleuth$^{(p.??)}$									✓
MOMIS$^{(p.34)}$									✓
SIMS$^{(p.35)}$			✓						
TSIMMIS$^{(p.35)}$			✓						
Clio$^{(p.36)}$			✓						
DELTA$^{(p.37)}$	✓								
TranScm$^{(p.41)}$				✓					
Breis & Bejar system$^{(p.42)}$							✓	✓	
MAFRA$^{(p.47)}$					✓				
OntoMapO$^{(p.48)}$					✓				
OntoMorph$^{(p.49)}$						✓			
SKAT$^{(p.50)}$							✓		
Autoplex$^{(p.35)}$		✓							
Automatch$^{(p.35)}$	✓								
COMA$^{(p.36)}$	✓								
CUPID$^{(p.36)}$	✓								
GLUE (iMAP)$^{(p.38)}$	✓								
Holistic Matching$^{(p.39)}$	✓								
Information Manifold$^{(p.33)}$			✓						✓
OBSERVER$^{(p.39)}$			✓						
OntoBuilder$^{(p.39)}$								✓	
SEMINT$^{(p.40)}$	✓								
W3TRANS$^{(p.41)}$				✓					
CAIMAN$^{(p.43)}$					✓				
Chimeara$^{(p.44)}$					✓			✓	
ConcepTool$^{(p.44)}$							✓		
FCA-Merge$^{(p.44)}$					✓				
IF-Map$^{(p.45)}$					✓				
ITTalks$^{(p.47)}$					✓				
OMEN$^{(p.48)}$					✓				
ONION$^{(p.48)}$							✓		
QOM$^{(p.49)}$					✓				
SMART, PROMPT, PROMPTDIFF$^{(p.51)}$					✓			✓	
S-Match$^{(p.52)}$					✓				

Table 5.2: Major objectives of semantic integration systems.

	Input	Output	Cardinality[1]	Rating
DIKE$^{(p.33)}$	ER	Mapping Table	1:1	yes
MOMIS$^{(p.34)}$	XML/XMLS/RDF/ relational/object	XML	n:1 (related terms)	N/A
SIMS$^{(p.35)}$	KL-ONE (Loom)	Query Rewriting Rules	unspecified	N/A
TSIMMIS$^{(p.35)}$	unspecified	Logic-based Queries	unspecified	N/A
Clio$^{(p.36)}$	Relational	Relational	unspecified	
DELTA$^{(p.37)}$	Text	Correspondence	unspecified	yes
TranScm$^{(p.41)}$	(Semi-)Structured	Rules	unspecified	
Breis and Bejar system$^{(p.42)}$	Text	Text/Tree-structure	unspecified	N/A
MAFRA$^{(p.47)}$	XMLS/Relational	DAML+OIL[2]	1:1/1:n/m:1	
OntoMorph$^{(p.49)}$	KR Languages	Rules	unspecified	no
SKAT$^{(p.50)}$	Structured	Matching Rules	1:1/1:n/m:1	
Autoplex$^{(p.35)}$	Relational	Relational	1:1/1:n	yes
Automatch$^{(p.35)}$	HTML	unspecified	1:1	yes
COMA$^{(p.36)}$	XML/Relational	Matching Pairs	1:1	yes
CUPID$^{(p.36)}$	XML/Relational	Matching Pairs	1:1/n:1	yes
GLUE (iMAP)$^{(p.38)}$	Unspecified (Relational)	unspecified	1:1(complex[3])	yes
Holistic Matching$^{(p.39)}$	Text/HTML	Mapping N-tuples	m:n:k	yes
OBSERVER$^{(p.39)}$	Text/HTML/ Relational/others	Lisp-style Query	unspecified	N/A
OntoBuilder$^{(p.39)}$	HTML Forms[4]	Term dictionary	1:1	
SEMINT$^{(p.40)}$	Relational	Relational	1:1	
W3TRANS$^{(p.41)}$	SGML/HTML/OO/ Structured Text	(R-)Correspondence Translation Rules	complex	no
CAIMAN$^{(p.43)}$	Bookmark Hierarchy	unspecified	unspecified	yes
Chimeara$^{(p.44)}$	OKBC-compliant files	HTML	unspecified	N/A
ConcepTool$^{(p.44)}$	EER	EER	unspecified	
FCA-Merge$^{(p.44)}$	Text documents	Merged Ontology	unspecified	no
Information Manifold$^{(p.33)}$	(Semi-)Structured	unspecified	complex	yes (sources)
IF-Map$^{(p.45)}$	unspecified	RDF	1:1/1:n	
ITTalks$^{(p.47)}$	DAML+OIL	DAML+OIL	semantic	
OMEN$^{(p.48)}$	RDF-like	Mapping Pairs	unspecified	yes
ONION$^{(p.48)}$	XML	Articulation Rules	1:1	yes
QOM$^{(p.49)}$	RDFS	Mapping Pairs	unspecified	
SMART, PROMPT PROMPTDIFF$^{(p.51)}$	Protégé Knowledge Model	(Mis-)Matching Pairs	1:1/1:n/m:1	yes
S-Match$^{(p.52)}$	Graph-like Structure	Mapping Matrix	semantic	yes

[1] *Semantic here refers to cases that instead of giving pair-wise mappings, relationships such as "broader than", "subsume", etc. are used.*
[2] *the output of MAFRA is instances of the Semantic Bridge Ontology in DAML+OIL.*
[3] *complex matching refers to the correspondences between combination of attributes in one data source and combination in the other.*
[4] *Ontobuilder is experimented on HTML forms, but, is argued that it is model independent.*

Table 5.3: Classification of semantic integration systems' criteria.

Bibliography

[1] S. Abiteboul, S. Cluet, and T. Milo. Correspondence and translation for heterogeneous data. *Theoretical Computer Science*, (275):179–213, Mar. 2002.

[2] S. Alagic and P. Bernstein. A model theory for generic schema management. pages 228–246. Springer, 2002.

[3] H. Alani, S. Dasmahapatra, N. Gibbins, H. Glasser, S. Harris, Y. Kalfoglou, K. O'Hara, and N. Shadbolt. Managing reference: Ensuring referential integrity of ontologies for the semantic web. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW'02), Siguenza, Spain*, pages 317–334, Oct. 2002.

[4] J. Ambite and C. Knoblock. Flexible and scalable cost-based query planning in mediators: a transformational approach. *Artificial Intelligence*, 118(1-2):115–161, Apr. 2000.

[5] F. Baader, D. Calvanese, D. McGuiness, D. Nardi, and P. Pater-Schneider. *The Description Logic Handbook.* Cambridge University Press, 2003. ISBN: 05-2178-1760.

[6] J. Barwise and J. Seligman. *Information Flow: the Logic of distributed systems.* Cambridge Tracts in Theoretical Computer Science 44. Cambridge University Press, 1997. ISBN: 0-521-58386-1.

[7] C. Batini, M. Lenzerini, and S. Navathe. A comparative analysis of methodologies for database schema integration. Dec. 1986.

[8] T. Bench-Capon and G. Malcolm. Formalising ontologies and their relations. In *Proceedings of 16th International Conference on Database and Expert Systems Applications (DEXA'99)*, pages 250–259, 1999.

[9] M. Benerecetti and S. Bouquet, P.and Zanobini. Soundness of semantic methods for schema matching. In *Proceedings of the ISWC'04 Meaning, Coordination and Negotiation (MCN'04) Workshop, Hiroshima, Japan*, pages 13–24, Nov. 2004.

[10] S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Record*, 28(1):54–59, 1999.

[11] S. Bergamaschi, F. Guerra, and M. Vincini. A Data Integration Framework for e-Commerce Product Classification. In *Proceedings of the 1st International Conference on the Semantic Web (ISWC'02), Sardinia, Italy*, pages 379–393, June 2002.

[12] J. Berlin and A. Motro. Autoplex: Automated discovery of contents for virtual databases. In *Proceedings of COOPIS-01, 6th IFCIS, International Conference on Cooperative Information Systems, LNCS 2172*, pages 108–122. Springer-Verlag, 2001.

[13] J. Berlin and A. Motro. Database schema matching using machine learning with feature selection. In *Proceedings of CAiSE 2002, Canada*, 2002.

[14] P. Bouquet, B. Magnini, L. Scrafini, and S. Zanobini. A sat-based algorithm for context matching. In *Proceedings of the 4th International and Interdisciplinary Conference on Modeling and Using Context (Context03)*, 2003.

[15] J. Breis and R. Bejar. A cooperative framework for integrating ontologies. *International Journal of Human-Computer Studies*, 56:665–720, 2002.

[16] D. Calvanese, G. deGiacomo, and M. Lenzerini. A framework for ontology integration. In *Proceedings of the 1st Intenational Semantic Web Working Symposium (SWWS), Stanford, CA, USA*, pages 303–317, aug 2001.

[17] D. Calvanese, G. deGiacomo, and M. Lenzerini. Ontology of integration and integration of ontologies. In *Proceedings of International Workshop on Description Logics (DL2001), Stanford, CA, USA*, pages 10–19, aug 2001.

[18] H. Chalupksy. OntoMorph: A Translation System for Symbolic Knowledge. In *Proceedings of the 17th International Conference on Knowledge Representation and Reasoning (KR-2000), Colorado, USA*, Apr. 2000.

[19] C. Clifton, E. Housman, and A. Rosenthal. Experience with a combined approach to attribute-mathing across heterogeneous databases. In *Proceedings of the 7th Conference on Database Semantics (DS-7), Leysin, Switzerland*, pages 428–455, Oct. 1997.

[20] E. Compatangelo and H. Meisel. Intelligent support to knowledge sharing through the articulation of class schemas. In *Proceedings of the 6th International Conference on Knowledge-Based Intelligent Information & Engineering Systems (KES'02), Crema, Italy*, Sept. 2002.

[21] J. De Bruijn and A. Polleres. Towards an ontology mapping specification language for the semantic web. Technical report, 2004. DERI Technical Report 2004-06-30.

[22] R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. imap: discovering complex semantic matches between database schemas. In *Proceedings of the 2004 ACM SIGMOD international conference of Management of data, France*, 2004.

[23] H.-H. Do and E. Rahm. Coma - a system for flexible combination of schema matching approaches. In *Proceedings of the 28th International Conference on Very Large Data Bases (VLDB'02), Hong Kong.*

[24] A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *SIGMOD Conference*, 2001.

[25] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *Proceedings of the 11th International World Wide Web Conference (WWW 2002), Hawaii, USA*, May 2002.

[26] M. Ehrig and S. Staab. Qom - quick ontology mapping. In *Proceedings of the 3rd International Semantic Web Confernece (ISWC'04), LNCS 3298, Hiroshima, Japan*, page 683, 2004.

[27] H. Enderton. *A Mathematical Introduction to Logic.* Academic Press, 2nd edition, Jan. 2001. ISBN: 0122384520.

[28] N. M. P. B. B. B. Fowler, J. Agent-based semantic interoperability in infosleuth. *ACM SIGMOD Record*, 28(1):60–67, 1999.

[29] E. Franconi, G. Stamou, J. Euzenat, L. Serafini, M. Ehrig, P. Bouquet, P. Hitzler, S. Tessaris, , and Y. Sure. Specification of a common framework for characterizing alignment, the D.2.2.1 report of knowledge web. Technical report, 2005.

[30] A. Gal, A. Anaby-Tavor, A. Trombetta, and D. Montesi. A framework for modelling and evaluating automatic semantic reconciliation. *The VLDB Journal*, 2003.

[31] A. Gangemi, G. Steve, and F. Giacomelli. ONIONS: An Ontological Methodology for Taxonomic Knowledge Integration. In P. van der Vet, editor, *Proceedings of the Workshop on Ontological Engineering, ECAI'96, Budapest, Hungary*, Aug. 1996.

[32] B. Ganter and R. Wille. *Formal Concept Analysis: mathematical foundations.* Springer, 1999. ISBN: 3-540-62771-5.

[33] H. Garcia-Molina, Y. Papakonstantinou, D. Quass, A. Rajaraman, Y. Sagiv, J. Ullman, V. Vassalos, and J. Widom. The TSIMMIS Approach to Mediation: Data Models and Languages. *Journal of Intelligent Information Systems*, pages 117–132, Mar. 1997.

[34] W. Grosso, H. Eriksson, R. Fergerson, J. Gennari, S. Tu, and M. Musen. Knowledge Modelling at the Millennium - The design and evolution of Protege2000. In *proceedings of the 12th Knowledge Acquisition, Modelling, and Management(KAW'99), Banff, Canada*, Oct. 1999.

[35] F. Guinchiglia, P. Shvaiko, and M. Yatskevich. S-match: an algorithm and an implementation of semantic mathing. In *Proceedings of 1st European Semantic Web Symposium (ESWS'04), Crete, Greece*, pages 61–75, May 2004.

[36] L. Haas, R. Miller, B. Niswonger, M. Tork Roth, P. Schwartz, and E. Wimmers. Transforming heterogeneous data with database middleware: Beyond integration. Technical report, 1999. IEEE Bulletin of the TC on Data Engineering (22)1: 31-37, March 1999.

[37] B. He and K. Chang. Statistical schema matching across web query interfaces. In *SIGMOD Conference*, pages 217–228, 2003.

[38] B. He and K.-C. Chang. Statistical schema matching across web query interfaces. In *Proceedings of the 2003 ACM SIGMOD Conference, San Diego*, 2003.

[39] D. Hong-Hai and E. Rahm. Coma - a system for flexible combination of schema matching approaches. 2002.

[40] I. Horrocks, P. Patel-Schneider, H. Boley, S. Tabet, B. Grosof, and M. Dean. *SWRL: A Semantic Web Rule Language Combining OWL and RuleML.* W3C, May 2004.

[41] Y. Kalfoglou, B. Hu, and D. Reynolds. On interoperability of ontologies for web-based educational systems. In *Proceedings of the 14th International World Wide Web Conference (WWW 2002) workshop on Interoperability of Web-based Educational Systems (WF05), Chiba, Japan*, May 2005.

[42] Y. Kalfoglou and M. Schorlemmer. Information-flow based ontology mapping. Informatics report no.135, Division of Informatics, University of Edinburgh, June 2002.

[43] Y. Kalfoglou and M. Schorlemmer. If-map: an ontology mapping method based on information flow theory. *Journal on Data Semantics*, 1:98–127, Oct. 2003. LNCS2800, Springer, ISBN: 3-540-20407-5.

[44] Y. Kalfoglou and M. Schorlemmer. Ontology mapping: the state of the art. *The Knowledge Engineering Review*, 18(1):1–31, 2003.

[45] R. Kent. The iff foundation for ontological knowledge organization. *Knowledge Organization and Classification in International Information Retrieval, Cataloging and Classification Quarterly, The Haworth Press Inc.*, 2003.

[46] A. Kiryakov, K. Simov, and M. Dimitrov. OntoMap: Portal for Upper-Level Ontologies. In *Proceedings of the 2nd International Conference on Formal Ontology in Information Systems (FOIS'01), Ogunquit, Maine, USA*, pages 47–58, Oct. 2001.

[47] M. Lacher and G. Groh. Facilitating the exchange of explicit knowledge through ontology mappings. In *Proceedings of the 14th International FLAIRS conference, Key West, FL, USA*, May 2001.

[48] J. Larson, S. Navathe, and R. Elmasri. A Theory of Attribute Equivalence in Databases with Application to Schema Integration. *IEEE Transactions On Software Engineering*, 15(4):449–463, Apr. 1989.

[49] O. Lassila and R. Swick. *Resource Description Framework(RDF) Model and Syntax Specification.* W3C, Feb. 1999.

[50] A. Levy. The information manifold approach to data integration. *IEEE Intelligent Systems*, 13:12–16, 1998.

[51] A. Levy and M. Rousset. Carin: a representation language integrating rules and description logics. In *Proceedings of ECAI'96, Hungary*, 1996.

[52] W.-S. Li and C. Clifton. SEMINT: A tool for identifying correspondences in heterogeneous databases using neural networks. *Data and Knowledge Engineering*, 33(1):49–84, 2000.

[53] P. Lord, S. Bechhofer, M. Wilkinson, G. Schiltz, D. Gessler, D. Hull, C. Goble, and L. Stein. Applying semantic web services to bioinformatics: Experiences gained, lessons learnt. In *Proceedings of the 3rd International Semantic Web Conference (ISWC'04), Hiroshima, Japan*, pages 350–364, 2004.

[54] J. Madhavan, P. Bernstein, P. Domingos, and A. Halevy. Representing and reasoning about mappings between domain models. In *Proceedings of the 18th National Conference on Artificial Intelligence (AAAI'02), Edmonton, Alberta, Canada*, Aug. 2002.

[55] J. Madhavan, P. Bernstein, C. Kuang, A. Halevy, and P. Shenoy. Corpus-based schema matching. In *Proceedings of the IJCAI'03 Workshop on Information Integration on the Web (IIWeb-03), Acapulco, Mexico*, Aug. 2003.

[56] J. Madhavan, P. Bernstein, and E. Rahm. Generic schema matching with cupid. In *Proceedings of the 27th International Conference on Very Large Databases (VLDB'01), Roma, Italy*, Sept. 2001.

[57] A. Maedche and S. Staab. Semi-automatic engineering of ontologies from texts. In *Proceedings of the 12th International Conference on Software Engineering and Knowledge Engineering(SEKE2000), Chicago, IL, USA*, pages 231–239, July 2000.

[58] D. Maynard, G. Stamou, H. Stuckenschmidt, I. Zaihrayeu, J. Barrasa, J. Euzenat, M. Hauswirth, M. Ehrig, M. Jarrar, P. Bouquet, P. Shvaiko, R. Dieng-Kuntz, R. Hernández, S. Tessaris, S. Van Acker, and T. Bach. State of the art on current alignment techniques , the D.2.2.3 report of knowledge web. Technical report, 2005.

[59] D. McGuinness, R. Fikes, J. Rice, and S. Wilder. An Environment for Merging and Testing Large Ontologies. In *Proceedings of the 17th International Conference on Principles of Knowledge Representation and Reasoning (KR-2000), Colorado, USA*, Apr. 2000.

[60] H. Meisel and E. Compatangelo. Conceptool: Intelligent support to knowledge management. In *Proceedings of the 9th Workshop on Automated Reasoning (ARW'02), Imperial College, London, England*, Apr. 2002.

[61] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Porceedings of the 18th International Conference on Data Engineering (ICDE)*, pages 117–128, 2002.

[62] E. Mena, V. Kashyap, A. Illarramendi, and A. Sheth. Domain Specific Ontologies for Semantic Information Brokering on the Global Information Infrastructure. In N. Guarino, editor, *Proceedings of the 1st International Conference on Formal Ontology in Information Systems(FOIS'98), Trento, Italy*, pages 269–283. IOS Press, June 1998.

[63] T. Milo and S. Zohar. Using schema matching to simplify heterogeneous data translation. In *Proceedings of the 24rd International Conference on Very Large Data Bases (VLDB'98), New York, NY, USA, USA*, pages 122–133, Aug. 1998.

[64] P. Mitra, N. Noy, and A. Jaiswal. Omen: A probabilistic ontology-mapping tool. In *Proceedings of the ISWC'04 Workshop on Meaning Coordination and Negotiation (MCN'04), Hiroshima, Japan*, Nov. 2004.

[65] P. Mitra and G. Wiederhold. Resolving terminological heterogeneity in ontologies. In *Proceedings of the ECAI'02 workshop on Ontologies and Semantic Interoperability, Lyon, France*, July 2002.

[66] P. Mitra, G. Wiederhold, and J. Jannink. Semi-automatic Integration of Knowledge Sources. In *Proceedings of the 2nd International Conference on Information Fusion (Fusion'99), Sunnyvale, CA, USA*, July 1999.

[67] F. Noy and M. Musen. SMART: Automated Support for Ontology Merging and Alignment. In *Proceedings of the 12th Workshop on Knowledge Acquisition, Modelling and Management (KAW'99), Banff, Canada*, Oct. 1999.

[68] F. Noy and M. Musen. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *Proceedings of the 17th National Conference on Artificial Intelligence, (AAAI'00), Austin, TX, USA*, July 2000.

[69] F. Noy and M. Musen. PROMPTDIFF: A Fixed-Point Algorithm for Comparing Ontology Versions. In *Proceedings of the 18th National Conference on Artificial Intelligence, (AAAI'02), Edmonton, Alberta, Canada*, pages 744–751, Aug. 2002.

[70] N. Noy. Semantic integration: A survey of ontology-based approaches. *SIGMOD Record*, 33(4):65–70, 2004.

[71] N. Noy and M. Klein. Ontology evolution: Not the same as schema evolution. *Knowledge and Information Systems*, 6(4):428–440, 2004.

[72] J. Pollock. The Wed Services Scandal: How data semantics have been overlooked in integration solutions. *eAI Journal*, pages 20–23, 2002.

[73] S. Prasad, Y. Peng, and T. Finin. A tool for mapping concepts between two ontologies. In *Proceedings of the AAAI'02 workshop on Ontologies and the Semantic Web, Edmonton, Canada*, pages 52–57, 2002.

[74] S. Prasad, Y. Peng, and T. Finin. Using explicit information to map between two ontologies. In *Proceedings of the AAMAS 2002 Wokshop on Ontologies in Agent Systems (OAS'02), Bologna, Italy*, July 2002.

[75] U. Priss. A Triadic Model of Information Flow. In *Proceedings of the 9th International Conference on Conceptual Structures (ICCS'01), Stanford, CA, USA*, pages 159–171, Aug. 2001.

[76] Rahm,E. and Bernstein,P. On Matching Schemas Automatically. Technical report, Microsoft Report, 2001. Technical Report 2001-17.

[77] D. Rosaci, G. Terracina, and D. Ursino. A Semi-automatic Technique for Constructing a Global Representation of Information Sources Having Different Formats and Structure.

[78] D. Rosaci, G. Terracina, and D. Ursino. A semi-automatic technique for constructing a global representation of information sources having different formats and structure. In *Proceedings of 12th International Conference on Database and Expert Systems Applications (DEXA'01), Munich, Germany.*

[79] E. Schulten, H. Akkermans, N. Guarino, G. Botquin, N. Lopes, M. Doerr, and N. Sadeh. The E-Commerce Product Classification Challenge. *IEEE Intelligent Systems*, 16(4):86–88, 2001.

[80] A. Sheth and J. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–230, Sept. 1990.

[81] P. Shvaiko. A classification of schema-based matching approaches. In *Proceedings of ISWC'04 workshop on Meaning, Negotiation and Coordination (MCN'04), Hiroshima, Japan*, Nov. 2004.

[82] J. Sowa. *Knowledge Representation: Logical, Philosophical, and Computational Foundations.* Brooks Cole Publishing Co., Pacific Grove,CA,USA, 2000. ISBN: 0-534-94965-7.

[83] G. Stumme and A. Maedche. Fca-merge: Bottom-up merging of ontologies. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence, Seattle, USA*, pages 225–230, Aug. 2001.

[84] G. Stumme and A. Maedche. Ontology Merging for Federated Ontologies on the Semantic Web. In *Proceedings of the International Workshop for Foundations of Models for Information Integration (FMII-2001), Viterbo, Italy*, Sept. 2001.

[85] M. Uschold and M. Grüninger. Ontologies and semantic integration. In Software Agents for the Warfighter*, the first in a series of reports sponsored by the US Government Information Technology Assessment Consortium (ITAC)*, 2002.

[86] J. Wang, K. Zhang, K. Jeong, and D. Shasha. A system for approximate tree matching. *IEEE Transactions on Knowledge and Data Engineering*, 6(4):559–571, 1994.

[87] F. Wiesman and N. Roos. Domain independent learning of ontology mappings. In *Proceedings of the 3rd International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS'04), New York, NY, USA*, pages 846–853, 2004.

[88] L. Yan, R. Miller, L. Haas, and R. Fagin. Data-driven understanding and refinement of schema mappings. In *SIGMOD*, 2001.