UNIVERSITÀ DEGLI STUDI DI PALERMO

Dottorato in Ingegneria Chimica, Gestionale, Informatica e Meccanica - Indirizzo Ingegneria Informatica

DICGIM - DIPARTIMENTO DI INGEGNERIA CHIMICA GESTIONALE INFORMATICA E MECCANICA

Settore disciplinare: ING-INF/05

PROBABILISTIC TECHNIQUES FOR BRIDGING THE SEMANTIC GAP IN SCHEMA ALIGNMENT

IL DOTTORE
ING. FRANCESCA ANASTASIO

IL COORDINATORE
CH.MO PROF. SALVATORE GAGLIO

IL TUTOR
CH.MO PROF. ROBERTO PIRRONE

XXVI CICLO - ANNO ACCADEMICO 2014-2015

DOTTORATO

To my family

Abstract

Connecting pieces of informations from heterogeneous sources sharing the same domain is an open challenge in Semantic Web, Big Data and business communities.

The main problem in this research area is to bridge the expressiveness gap between relational databases and ontologies. In general, an ontology is more expressive and captures more semantic information behind data than a relational database does. On the other side, databases are the most common used persistent storage system and they grant benefits such as security and data integrity but they need to be managed by expert users.

The problem is quite significant above all when enterprise or corporate ontologies are used to share infomations coming from different databases and where a more efficient data management is auspicable for interoperability purposes.

The main motivations on this thesis are related to the database access via ontology, as in the OBDA (Ontology Based Data Access) scenario, wich provides a formal specification of the domain close to the human's view, while technical details of the database are hidden from end-user, and also the persistent storage of ontologies in databases for facilitating search and retrieval, keeping the benefits of database management systems. In these cases the assertion component(A-Box) is usually stored into a database, and terminological one (T-Box) is mantained in an ontology. So it is more necessary to align schemas than matching instances.

The term *alignment* can be used to define the whole process comprising the mapping process between two existent heterogeneous sources, such as ontology and relational database, and the trasformation process from a representation to the other one, such as ontology-to-database and database-to-ontology.

Defining mappings manually is an hard task expecially for large and complex data representations and existing methodologies fail in loosing some contents and several elements are left unaligned.

In this thesis are discussed various aspects of the alignment in all these senses.

The presented techniques are based on a probabilistic approach that fits well on the uncertain alignment process, where are involved two different representations with a different level of expressiveness. In the methodology ontologies and databases are described in terms of Ontology Web Language (OWL) and

Entity-Relationship Diagram (ERD) lexical descriptions. So, the ontologies are represented by a set of OWL axioms while a properly defined Context-Free Grammar (CFG) is used to represent ERDs (Entity-Relationship Diagrams) as a set of sentences.

Both the $OWL \rightarrow ERD$ transformation and the mapping rely on HMMs (Hidden Markov Models) to estimate the most likely sequence of ERD symbols observing OWL symbols. In the model definition OWL constructs are the observable states, while the ERD symbols are the hidden states.

The tools developed, one for $OWL \rightarrow ERD$ transformation purpose, called OMEGA (Ontology $\rightarrow$ Markov $\rightarrow$ ERD Generator Application) and one for mapping OWL and ERD, called HOwErd(HMM OWL-ERD) own their own GUI interface for showing the alignment results. Finally, HOwErd is compared with the most widespread tools in the reference literature.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Nowadays, several actors (people, enterprises, education institutions, and so on) need to share information with each other on the web. To cope with this need, web systems are focused towards high interoperability, and efficient knowledge management.

Ontologies and relational databases are the most widespread solutions for representing domains and for data storing.

Informations are generally stored in relational databases and managed through DBMS (DataBase Management System). In the last years information management systems use other interfacing methods with digital archives , more expressive than the old ones, realized on semantic technologies, NLP (Natural Language Processing) and ontologies.

Ontologies are domain representations and, according to Schreiber et al. [69] they provide the means for describing explicitly the conceptualization behind the knowledge represented in a knowledge base. An ontology can describe a domain as a set of terms hierarchically structured, so the same ontology can be used for several knowledge bases, which would to share the same skeleton or taxonomy and it is possible to extend it by adding domain-specific subconcepts, or adding upper level concepts covering other areas [37].

The importance of building mappings between relational schemas and ontologies, and approaching the expressiveness gap problem, is growing both in Semantic Web and business communities.

The problem pertains those subjects that have to organize their informations through several management sub-systems in a dynamic organization net scenario, such as enterprises and public administrations that need an efficient corporate knowledge management for granting a good interoperabilty between subagencies, branches or departments. Often, operating in this sense, can arise some problems related to the heterogeneity of informations and the data storage supports.

The motivations of the interest about integrating ontologies and relational databases are mainly related to the ontology storing in existent relational databases process which allows efficient storage and retrieval of very large OWL ontologies thus enabling a high query performance and to the OBDA (Ontology Based Data Access), where representing data in a relational schema using a higher level language with more semantics allows a better interfacing with human users.

In existing OBDA systems the user has to define manually the mapping rules between the two different representations, but this kind of mapping is an expensive and hard task, especially for the case of large and complex databases.

Generally in literature have been presented several approaches, that we can summarize in three kinds of ontology and databases integration tasks:

- *Transformation from database to ontology*: the process that allows to generate a new ontology given a relational database schema by applying some transformation rules;
- *Transformation from ontology to database*. It is the complementary process in respect to the previous, that allows to generate a new database schema given an ontology by applying some transformation rules. Some ontology elements can be lost considering the less expressiveness of database representation;
- *Mapping between ontology and database*, that is the process of defining correspondences between element of both representations, attempting to align all the entity pairs in their Cartesian product. There could be many to many correspondences.

The term *alignment* can be used to define the whole process comprising both the mapping process between the two existent heterogeneous sources and the trasformation processes from a representation to the other one, in double directions.

In figure 1.1 is illustrated the overall alignment scenario. In this example, we assume an ontology expressing a domain; it could be also an enterprise ontology, expressing the whole set of informations useful for a corporation. The concepts in the ontology can be linked with tables in a single database or more, through some mapping rules.

Sometime, it is possible to find databases clearly mapped with any ontology concepts by default, from ontology construction process itself, otherwise a database could contain informations not really relevant for the domain of that single ontology, so they could not be included in the ontology but it could be necessary have a high level representation of them in the same ontology or in another one also for future tasks. On the other side, existing methods for semantical enrichment of the ontology, as in VEBO [67] approach, allow to add new concepts and relations

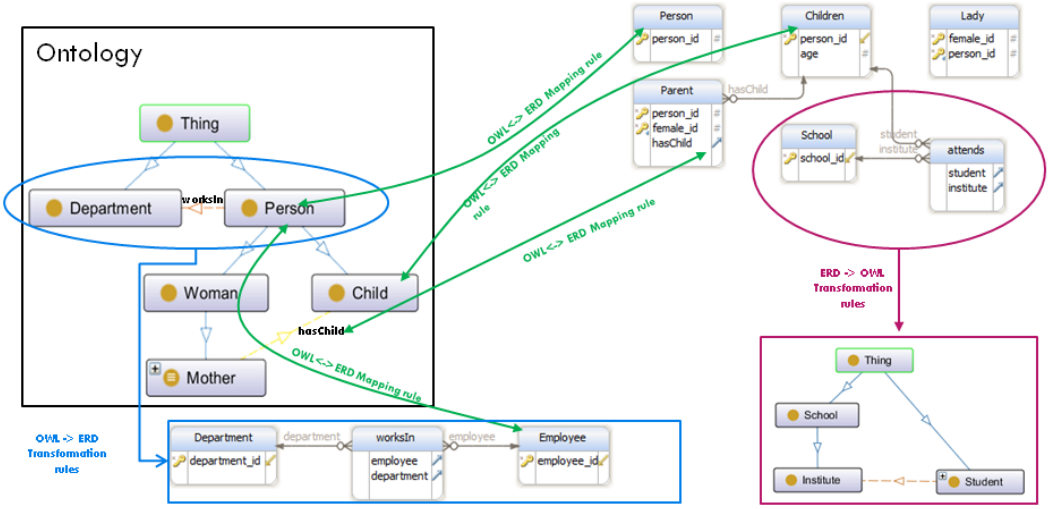


Figure 1.1: *The overall alignment scenario example*

between them; these concepts can be not directly linked with a table in a database, so it could be useful make possible a data retrieval also for this concepts, e.g. for ontology storing tasks.

Furthermore an ontology concept could be referenced by different tables in different databases.

So, it is clear that, under such premises, a full-automatic alignment process is not completely feasible, expecially in mapping or ontology-to-database transformation sense, but it is possible to realize a semi-automatic alignment process aiming to produce suggestions for the user who needs to define mapping rules for knowledge management tasks.

Managing heterogeneous information sources, and the development of a system able to align different representations, modeling also the uncertainty due to the expressiveness gap are the main focus of the reseach work presented in this thesis.

A few approaches in literature define rules for a direct translation of a database schema to an OWL (Ontology Web Language) ontology.

Approaches that consider the alignment between both representations often provide a solution passing through a prior step of translation from a representation to the other one, tipically from database schema description to OWL, which is the more expressive one, and then perform an alignment between homogeneous sources.

The techniques presented in this thesis propose a statistical method, using Hidden Markov Model (HMM) and aim at suggesting the most likely alignment in all three senses.

In the methodology ontologies and databases are described in terms of OWL 2

and Entity-Relationship Diagram (ERD) lexical descriptions. OWL 2 lexical description is the set of functional-style syntax (FSS) [55] of the ontology. There is not an equivalent representation for the conceptual model of the relational database, so has been considered an alternative methodology for implementing this lexical description. The lexical descriptions for relational schemas are obtained using a properly defined Context-Free Grammar (CFG), according to the simbols used in the Barker Notation [16].

The techniques proposed in this thesis can be devised into $ERD \rightarrow OWL$ transformation, $OWL \rightarrow ERD$ transformation and $OWL \leftrightarrow ERD$ mapping, according to the different alignment approaches.

The $ERD \rightarrow OWL$ transformation is a well-defined process, because, due the lesser expressivity of ERD, all OWL constructs cover the ERD ones in the rules that are retrieved in literature, so each ERD construct can be transformed to an equivalent OWL one. So the approach proposed for this task, uses the same direct rules set for that purpose. An extension of these rules was derived empirically, observing a large set of corresponding pairs of OWL ontology and database schema building also a dual correpondences set useful for the other tasks, looking for bridging the structural differences.

In the $OWL \rightarrow ERD$ transformation process and in the $OWL \leftrightarrow ERD$ mapping process, the fundamental step consists in the definition of the HMM, whose observable states are OWL constructs, and the hidden states are the terminal symbols of the ERD grammar. As a result, the system estimates the most likely sequence of ERD symbols in the given schema that corresponds to the OWL axiom of the given ontology.

In the proposed $OWL \rightarrow ERD$ transformation the labels set is empty, because in the transformation task there is no information about the database labels, while in the $OWL \leftrightarrow ERD$ mapping process both the labels of the ontology and the database ones are involved. This consideration entails a different computation of emission and transition probabilities.

In $OWL \rightarrow ERD$ transformation the transition matrix have been set up heuristically by default and emission probabilities are computed starting from the rules in literature.

In $OWL \leftrightarrow ERD$ mapping, a similarity measure has been defined for the emission probabilities computation between two labels. This is because the probability of matching two labels depends on both semantic similarity and syntactic similarity measures. So was used a weighted combination of Jaro-Winkler distance, for the syntactic aspect, and Wu-Palmer distance, for the semantic. For the transition probabilites in the mapping approach, has been proposed a computation based on the definition of *transition trees*.

In both $OWL \rightarrow ERD$ transformation and $OWL \leftrightarrow ERD$ mapping cases, for computing of the most likely sequence, the Viterbi algorithm was used.

In this thesis four GUI versions are described: one for the $OWL \rightarrow ERD$ transformation process, one for development/testing purposes, which show to the user all the mapping process steps, an interactive desktop version and a web version for the end user which allow user to choose a proposed mapping or deny it.

Finally, the mapping approach was compared with the most widespread methods in literature that perform the same task, obtaining interesting results.

1.1 Publications

The work presented in this thesis resulted in the publication of the following research articles:

- Russo, G., Anastasio, F., Pipitone, A., Gentile, A., Pirrone, R. (2012). VEBO: Validation of E-R diagrams through ontologies and WordNet. In Proceedings - IEEE 6th International Conference on Semantic Computing, ICSC 2012 (pp. 342-344).
- Pipitone, A.; Anastasio, F.; Pirrone, R., An Innovative Statistical Tool for Automatic OWL-ERD Alignment. In Proceedings - IEEE 10th International Conference on Semantic Computing, ICSC 2016
- Pipitone, A.; Anastasio, F.; Pirrone, R., A Hidden Markov Model for Aligning Relational Databases to OWL Ontologies In proceedings - Fourth International Workshop on Semantic Computing for Social Networks: from user information to social knowledge, SCSN 2016

1.2 Thesis structure

The thesis is organized as follows. Chapter 2 describes the main issues. Then the enterprise ontologies are introduced, data integration and main OBDA systems and analyzed which are the problems on expressiveness gap betweeen languanges used to represent informations. Then an overview on alignment scenarios, both in the sense of mapping and in the sense of trasformation is outlined.

In Chapter 3 the methodologies in literature treating of mapping or trasformations are described, above all the most widespread tools which were used for a comparative analisys with the proposed system.

Then, in Chapter 4 is reported an excursus that is related to the whole theoretical background needed for developing the system. The methodology is described in Chapter 5. The architecture is described in Chapter 6.

Experiments and discussion of the results are presented in Chapter 7.

Chapter 8 deals with a possible prosecution of research activity in this area.

Chapter 2

Motivation

This section describes the main motivations which prompted this research.

Data integration will be analyzed along with the *OBDA (Ontology Based Data Access)* scenario, and the *Ontology Storing* challenge, considering the *expressiveness gap* problem that is the main issue of the research and the techniques presented in this thesis.

Finally enterprise and corporate ontologies are described. Indeed, the expressiveness gap problem is a relevant issue in enterprise knowledge management.

2.1 Data integration

Data integration is one of the major challenges in Infomation Technology and it is well defined by Lenzerini [46], as the problem of combining data residing at different sources, and providing the user with a unified view of these data.

Data integration is presented in literature as a pervasive challenge faced in applications that need to query across multiple autonomous and heterogeneous data sources; so it is important in large enterprises that own a multitude of datasources, where datasets are being produced independently by multiple users, for better cooperation among agencies, each with their own data sources, and in offering good search quality across the millions of structured data sources on the World-Wide Web [38].

Data integration aims at providing users, who doesn't know how data are stored in repositories, with an access to heterougeneous datasources. The problem is growing due the increasing amount of data which need to be stored and shared. Often these data are stored and managed in different ways. Therefore, the main need is to provide a unified flexible mechanism for accessing the whole set of heterogeneous data.

The main components of a data integration system are the global schema, the sources, and the mapping.
Thus, according to [46, 25] data integration system $\mathcal{I}$ is formalized as a triple $<\mathcal{G}, \mathcal{S}, \mathcal{M}>$ where:

- $\mathcal{G}$ is the global schema. It provides a description of the domain of interest, as a reconciled, integrated, and virtual view of the underlying sources, expressed in a language $\mathcal{L}_\mathcal{G}$ over an alphabet $\mathcal{A}_\mathcal{G}$. The alphabet comprises a symbol for each element of $\mathcal{G}$ (i.e., relation if G is relational, class if $\mathcal{G}$ is object-oriented, etc.);
- $\mathcal{S}$ is the source schema. It describes the structure of the sources, where the real data are, expressed in a language $\mathcal{L}_\mathcal{S}$ over an alphabet $\mathcal{A}_\mathcal{G}$. The alphabet $\mathcal{A}_\mathcal{S}$ includes a symbol for each element of the sources;
- $\mathcal{M}$ is the mapping between $\mathcal{G}$ and $\mathcal{S}$, constituted by a set of assertions of the forms

$$q_\mathcal{S} \rightsquigarrow q_\mathcal{G}$$
$$q_\mathcal{G} \rightsquigarrow q_\mathcal{S}$$

where $q_\mathcal{S}$ and $q_\mathcal{G}$ are two queries of the same arity, respectively over the source schema S $\mathcal{S}$, and over the global schema $\mathcal{G}$. Queries $q_\mathcal{S}$ are expressed in a query language $\mathcal{L}_\mathcal{M}$, $\mathcal{S}$ over the alphabet $\mathcal{A}_\mathcal{S}$, and queries $q_\mathcal{G}$ are expressed in a query language $\mathcal{L}_\mathcal{M}$, $\mathcal{G}$ over the alphabet $\mathcal{A}_\mathcal{G}$. The assertions in the mapping establish the connection between the elements of the global schema and those of the source schema. So, an assertion $q_\mathcal{S} \rightsquigarrow q_\mathcal{G}$ specifies that the concept represented by the query $q_\mathcal{S}$ over the sources corresponds to the concept in the global schema represented by the query $q_\mathcal{G}$ (similarly for an assertion of type $q_\mathcal{G} \rightsquigarrow q_\mathcal{S}$).

2.1.1 Data integration with uncertainty

Dong et al. [34] introduce a new perspective in data integration area, considering the uncertainty of data. This condition occurs in some practical applications where it is not possible to specify exactly the mappings. A data integration system needs to handle uncertainty at three levels.

- *Uncertain schema mappings*: schema mappings can be inaccurate. In many applications it not possible to create and maintain precise mappings between data sources. This can happen because the users are not skilled enough to provide precise mappings, and they do not understand the domain well or

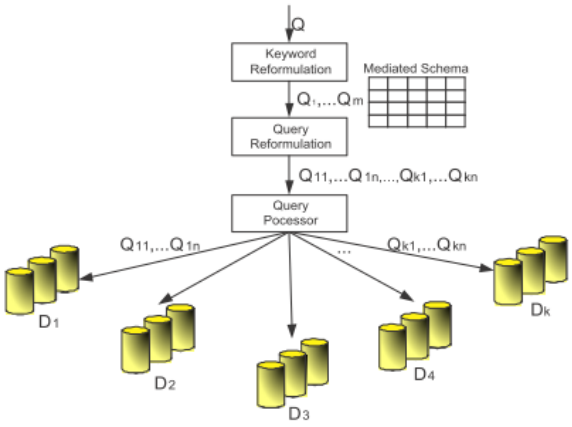


Figure 2.1: *Architecture of a data-integration system that handles uncertainty [34]*

do not even know what correct mappings are, such as in bioinformatics, or due the broadness of datasources.

- *Uncertain data*: date are naturally uncertain; data are often extracted from unstructured or semi-structured sources by automatic methods (e.g., HTML pages, emails, blogs) or they may come from sources that are unreliable or not up to date.

- *Uncertain queries*: queries can be posed as keywords rather than as structured queries against a well defined schema. The system needs to translate these queries into some structured form so they can be reformulated with respect to the data sources. Thus, the system may generate multiple candidate structured queries and have some uncertainty on the real intent of the user.

The architecture of the system proposed by Dong et al. [34] is shown in Figure 2.1. In this case a probabilistic schema mapping describes a probability distribution of a set of possible schema mappings between a source schema and a target schema. The approach presented in that cited work, uses Bayes Nets for representing probability distribution and a top-k algorithm for query answering.

Due the use of probabilistic approach and the effective relevance of uncertainty in data integration, the cited work opens the way on arguing probabilistic approaches also for the domain considered in this thesis.

2.1.2 OBDA - Ontology Based Data Access

In [21] Ontology-based data access (OBDA) is defined as a new paradigm for accessing and integrating data, whose key idea is to resort to a three-level archi-

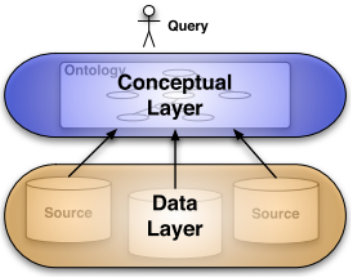


Figure 2.2: *Ontology based data access*

tecture, constituted by the ontology, the data sources, and the mapping between the two. According to the Figure 2.2, OBDA uses a conceptual layer, exported to the client, abstracting away from the data layer, which is is used for data storage. So, the best formalization of a *conceptual layer* is an ontology, that is defined a formal description of the domain of interest, given in terms of concepts, roles, i.e., binary relations between them, and attributes, while RDBMSs are naturally good as *data layer* and constitutes the best management system in term of efficiency also for a huge amount of data.

The OBDA approach is very useful when accessing data is quite difficult, as in the case underlying databases have undergone several manipulations or they are distributed or replicated over the years, or moreover are the results of a merging. In all this cases data could be redudant or missing or inconsistent. Clearly, it is the most common scenario in as Enterprise Application Integration, Data Integration, and the Semantic Web.

The set of mappings specifies the relationships between the conceptual layer and the data sources, as described in section 2.1.

The OBDA scenario fails in complex domains and when data changeg over time. In the first case, the complete specification of mappings becomes expensive. In the second case, changes are desirable in the ontology and/or in the schemata of the data spurces, and consequently in the mappings; new strategies for bootstrapping and maintenance of ontology and mappings are required. The classical OBDA approaches fail to provide support for these two problems.

2.2 Ontology Storing

Storing ontologies in database can be a solution for persistent storage of ontologies, that allows rapid performing of operation such as information search and retrieval, exploiting the benefits of relational databases management systems such as transaction management, security and integrity control.

Figure 2.3: *transformation of relational database to ontology*

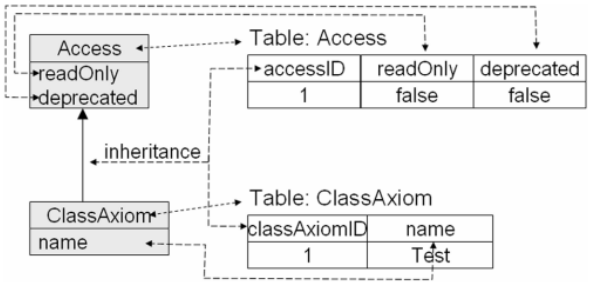


Figure 2.4: *A transformation example*

In literature have been defined two techniques for storing ontologies [39]:

- The first technique is to use file systems for storing ontologies in flat files. The main problem with this technique is that file systems do not provide scalability, sharability, or any query facility.

- The second technique is to use database management systems for storing ontologies in databases. The main problem with this technique is that database management systems require that an ontology should have a fixed structure, which cannot be guaranteed as ontologies are often built in a distributed way.

Furthermore, works in literature distinguish two different processes: the *transformation from ontology to relational databases* and the *mapping between relational databases and ontologies*.

The differences between the two processes is shown respectively in Figure 2.3 and Figure 2.5 and described in the following subsections. The whole process that involves both transformation from ontology to database, trasformation from database to ontology and mapping between ontology and database can be referred as *alignment*.

2.2.1 Ontology transformation to relational schema

The *transformation from ontology to relational database* is a process that involves an ontology aiming to create a corresponding database that have to contain the

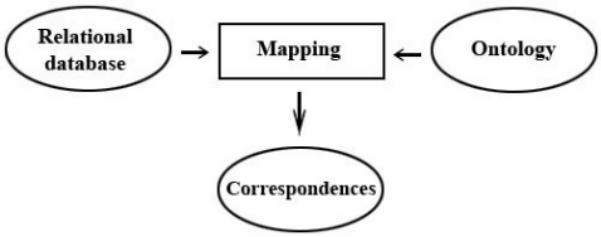


Figure 2.5: *mapping beetween relational database and ontology*

same ontology informations set. Therefore, the main goal is to have a persistent storage perfectly matchable with the input ontology. This is possible by defining a transformation rules making one to one correspondences between the two representations. The concept of transformation process is shown in Figure 2.3.

The Figure 2.4 shows a graphical example of transformation from ontology to database.

The transformation of ontologies to relational databases process handles problems [10] as:

- *Loss of data.* The result of the transformation should adequately describe the original data.
- *Structure loss.* In some cases, the transformation is not really lossless in the sense that not all constructs in an ontology can be mapped to a relational database.
- *Focus on structures.* Besides the mapping of structures, mechanisms should be provided for the mapping of data (i.e. instances).
- Focus on data: Data should be mapped, with incorporation of data types.
- *Applicability.* In some cases, the transformation is not really general in the sense that its application is rather restricted.
- *Correctness.* The transformation should have provable correctness.

2.2.2 Mapping between ontologies and relational schemas

In managing and retrieving tasks both the ontology and the database (or more than one) are known, so they need to be *mapped* to perform an alignment. The alignment process in this sense consists in discovering mappings between relational database schemas and ontologies. As for the transformation, the process entails the creation of mapping rules making one to one correspondences between the two representations. In figure 2.5 is shown the concept of mapping process, while the

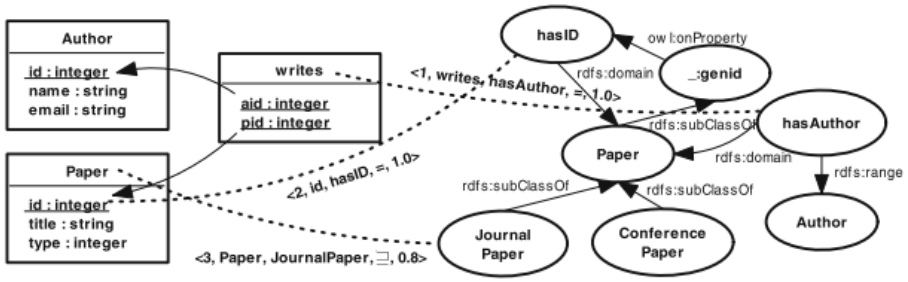


Figure 2.6: *A mapping example*

figure 2.6 shows a practical toy example of a mapping: on the left a database in which are stored information about authors who write papers, on the right is shown an ontology that represent the domain of conference papers. The mapping rules are represented graphically as dotted edges, bridging an ontology concept or property with respectively a table or a field in the database, according to semantic or structural features.

Clearly, also in OBDA systems, mapping process constitutes a crucial step and an effective automatic approach is desiderable. Both mapping rules detecting process, as for in transformation one, clushing with the expressiveness gap problem.

2.3 Expressivity gap issue between different abstraction level languages

In the previous sections were analyzed the main challenges in the area of data storing and data integration. In all these processes, the crossing issue is represented by the expressiveness gap between two abstraction levels.

Ontologies play an important role in interoperability tasks, because of significant characteristics [57, 1] such as:

- Adding rich, machine-readable semantics to data
- Sharing common understanding of the structure of information among people or software agents
- Separating domain knowledge from operational knowledge
- Making explicit domain assumptions
- Enabling reuse of domain knowledge

Obviously, an ontology is clearly more expressive than a database. Relational databases are based on closed-world assumption while ontologies use open-world semantics; there usually exist some approximate correspondences between them but these correspondences may be uncertainty affected.

Relational databases can be formalized by First Order Logic (FOL); while the logical foundation for OWL ontologies is Description Logic (DL), which is a subset of FOL. Thereupon, it is feasible to construct correctly transformation from relational database schemas and ontologies but, in the opposite direction, from OWL ontologies to relational schema representation, it is not always so easy.

2.4 Enterprise and Corporate ontologies

As stated by Alalwan et al. [1] an ontology can be used with a database also to provide a conceptual vision of heterogeneous data sources distributed in a number of databases with an interface built on an ontological model. This kind of scenario is typical in enterprises enviroments.

In [74] an *enterprise ontology* is defined as a collection of terms and definitions relevant to business enterprises. According to this definition they are generally used to define and organize relevant knowledge about enterprise activities, processes, organizations, strategies, marketing, etc..

That infomations can be stored in one or more than one underlying database. O'Leary [58] made a survey on the enterprise ontologies and describes developments in enterprise ontologies and some of the works focused on this area, such as ARIS, REA, Enterprise Ontology, TOVE.

Enterprise ontology could be modeled as a single ontology with different classes, describing any part of the enterprise domain, as in the ARIS's meta business process (Figure 2.7) or a federation of ontologies as in TOVE (Figure 2.8).

If the enterprise is a large corporation, with agencies and departments, e.g. a public administration or a holding group, the evolution of ontology management system is represented by the *corporate ontology* that describes the overall enterprise domain. It is a big ontology useful to be shared by the different local knowledge management systems. This kind of ontology often derives from a federation of some sub-ontologies provided by each agency or department knowledge management system.

In both enterprise or corporate acception, all ontology informations, properties and instances need to be stored in a database, managed and retrived for user query purposes.

Starting from this observation point there are some issues that involve both the enterprise or corporate ontology and the underlying databases, tipically they are more generally defined in data integration and expressivity gap research fields.

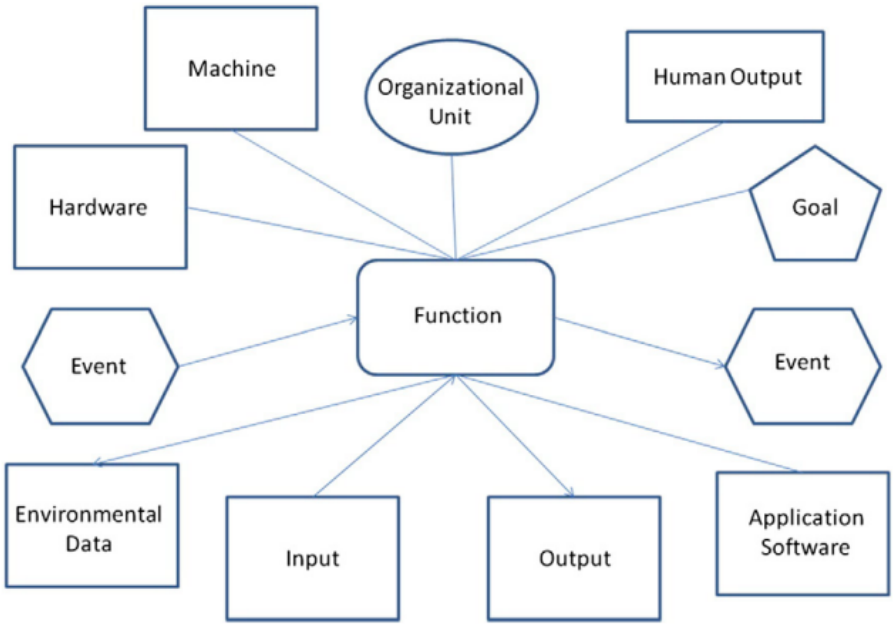


Figure 2.7: *ARIS's meta business process model*

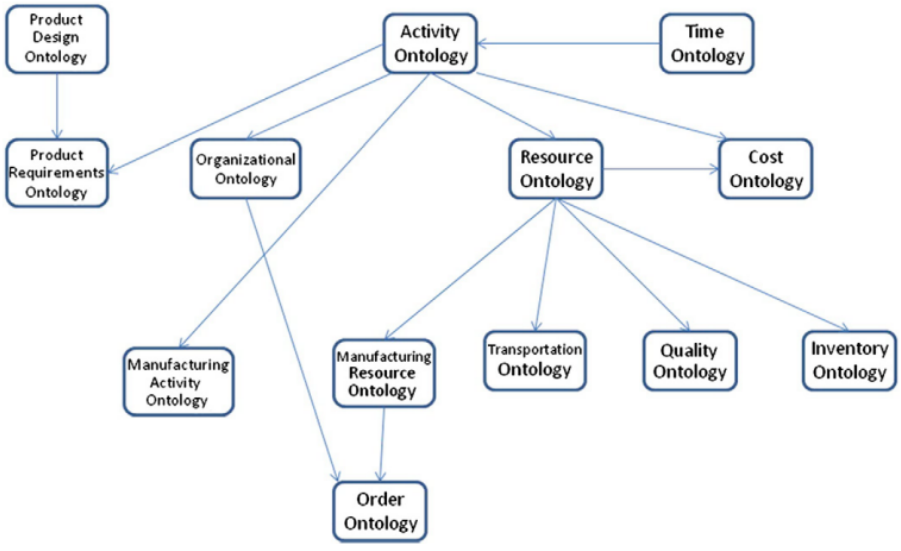


Figure 2.8: *TOVE interlinking ontologies*

Chapter 3

State of the art

Alignment process regards the processes concerning:

- the database-to-ontology transformation (referred as $ERD \rightarrow OWL$), that indicates the process of creating an ontology from an existing database using rules;
- the ontology-to-database transformation (referred as $OWL \rightarrow ERD$), that indicates the process of creating a database conceptual model from an existing ontology using rules;
- the mapping between ontology and database(referred as $OWL \leftrightarrow ERD$), where both ontology and database exist.

The main strategies devised in literature for solving the problem of bridging the expressiveness gap between ontologies and databases need the definition of *mapping rules* to connect an element of a representation to an element of the other one.

Discovering these rules still represents an open challenge in research areas such as Semantic Web, databases and knowledge management and in the latest years it has been approached in different ways.

The most works address the three problems discovering descriprive correspondences between the OWL ontology elements, such as Classes, Object Properties, Data Properties, several kind of Restrictions on data, etc., and relational database conceptual model elements, such as Entities, Relationships, Attributes and so on.

The choise to use conceptual model representation to describe the database derives from the difference between logical and conceptual model. As stated also by Dadjoo and Kheirkhah [31] in conceptual model there is a clear distinction between concepts and relations while in logical model tables are used to represent both concepts and some relationships between concepts cases; conceptual model

expresses inheritance hierarchy and cardinality constraints are explicitly displayed. Conceptual data model is closer to ontology design semantics.

In this Chapter the methodologies described in literature for rules discovering are analyzed and a survey of the mapping techniques is reported. Finally also the OBDA systems are considered, that are used in data integration field.

3.1 Methodologies for rules discovering

Mapping rules estabilish some correspondences between both representations, can be used for transformation tasks. Generally the methods are deterministic and described descriptively.

3.1.1 Transformation from database to ontology ($ERD \rightarrow OWL$)

The database-to-ontology transformation can be defined as a function $f_{eo} : ERD \rightarrow OWL$ whose domain is the set of ERD constructs, while the codomain is the set of OWL constructs; the function associates for each element of the database at least a construct of the ontology.

Due the less expressiveness of ERD with respect to OWL, the fully set of ERD constructs can be represented by an OWL construct or also more than one.

The $ERD \rightarrow OWL$ is a well-defined process in literature and over the years it has been widely approached.

ER2OWL [35] is a framework that aims to help software engineers in translating ERD concepts into OWL ontology, by applying some rules. The purpose is to resolve model conflict in ontology integration task. In [35] the author made a comparison with the previous work ERONTO [73], that extracts ontologies from Extended ERD. According to Fahad [35], ER2OWL addresses the limitations of ERONTO, that lacks some features and does not produce complete mappings, leaving the user to check the incompleteness and write glue code himself and gave inappropriate mappings expecially when Restriction are involved or the functional property is omissed in translating a single valued property, allowing attribute to accept many values and so producing inconsistences.

Xu et al. [78] proposed a method for deep annotation, used for dynamic web page contents extract from the database. The framework developed, called DPAnnotator, performs a rule-based translation from the ER Schema to the OWL ontology and the D2OMapper tool creates automatically mappings based on that rules.

Astrova et al. [9], [7] made a huge work in this field providing some rules for mapping relational databases in owl ontologies. In [7] the author performs a kind of

mapping definition that Dadjoo and Kheirkhah [31] classifies as based on logical model approaches. The method maps constructs of a relational database to an ontology using the names of constructs of the relational database as the names of constructs of the ontology. The choise to use names for translating and also for mapping purposes is quite common in alignment heterogeneous sources.

The approach proposed by Myroshnichenko and Murphy [56] is similar to the previous one. The authors define five mapping rules and related to the correspondences between ER components and OWL ontologies, whose details are presented in and summarized in the table 3.1

Lin et al. [49] proposed an approach and the related tool, called R2OWL, based on DBRE (DataBase Reverse Engineering) aiming at extracting richer and more natural semantics from the database. The conceptual correspondences used in developing their algorithm are summarized in the 3.1. Trinh et al. [72] presented the RDB2ONT tool for generating an OWL ontology from a database, extracting the meta-data yet and considering the structural constraint and applying a properly defined algorithm. One of the purposes is to bridge the semantic gaps between the underlying relational database systems and existing and well-known domain ontologies, aiming to enabling the semantic interoperability between relational database systems in large-scale environments.

As noted by Alalwan et al. [1], generally most approaches, such the ones presented in [47] and [19], fail in differentiating entities having or not an *IS-A* relationship or in not considering it, such in [9].

Some approaches derive rules from examples; it can mislead the results and can affect both the generalization of the rules and the accuracy of the system. Furthermore, some approaches avoid some circumstances or , like in [79], miss to cover some issues such as multi-valued attributes and composite attributes from the relational database side or restriction on the ontology side.

Finally, Albarrak and Sibley [2] made a good survey on methods that operate the transformation in $ERD \rightarrow OWL$. Due to the huge amount of techniques, the work for a more in-depth analysis while the Figures 3.2 and 3.3 show the results of comparison carried out by Albarrak and Sibley [2] between several $ERD \rightarrow OWL$ approaches.

3.1.2 Transformation from ontology to database ($OWL \rightarrow ERD$)

The $OWL \rightarrow ERD$ function $f_{oe} : OWL \rightarrow ERD$ is defined in a domain which is a set of OWL constructs, while the codomain is a set of ERD constructs. As for the $ERD \rightarrow OWL$ case, the function defines s set of mapping rules.

The main issue in $OWL \rightarrow ERD$ transformation is that the original semantics

ER Component	OWL Component
Entity	Class
Strong Entity	Class
Weak Entity	Base class and additional class encapsulating equivalents of the partial key attributes and the key attribute of the owner entity
Attribute	Datatype property
Single-valued Attribute (nullable)	Functional datatype property
Single-valued Attribute (not nullable)	Functional datatype property with min constraint set to one
Multi-valued Attribute (nullable)	Datatype property
Multi-valued Attribute (not nullable)	Datatype property with min constraint set to one
Key Attribute	Functional datatype property with min constraint set to one
Composite Attribute	Class with properties corresponding to components of the composite attribute
Binary Relationship without Attributes	Pair of inverse object properties
Binary Relationship with Attributes	Class with datatype properties corresponding to the relationship's attributes and two pairs of inverse object properties associating the participating entity classes and the relationship class
Ternary Relationship	Class with three pairs of inverse object properties associating the participating entity classes and the relationship class
Participating Entity Role Name	Name of the appropriate object property in the pair of inverse object properties manifesting a relationship without attributes
Min Cardinality Constraint	Min cardinality restriction
Max Cardinality Constraint	Max cardinality restriction

Table 3.1: *Correspondences between ER components and ontologies stated in [56]*

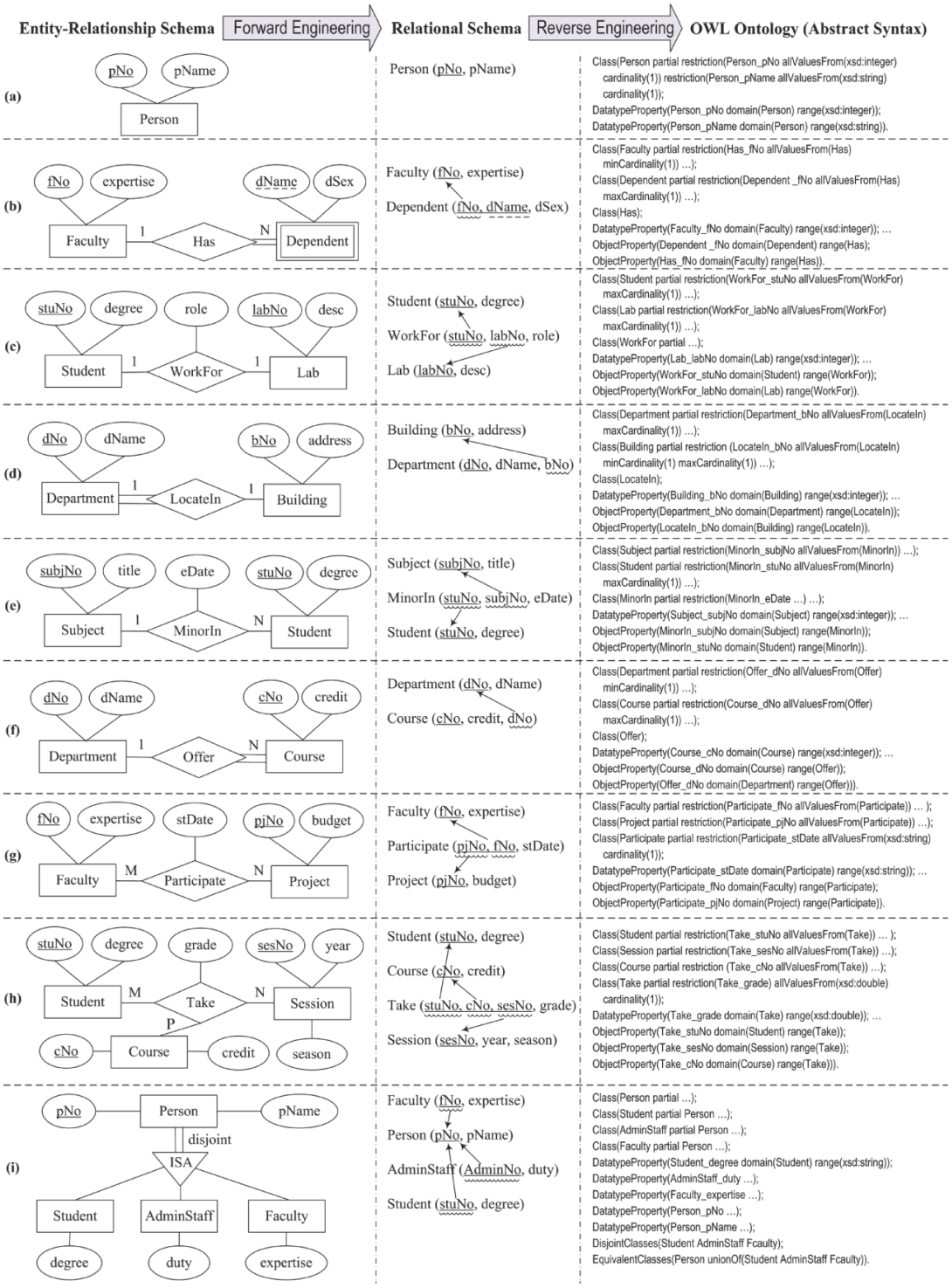


Figure 3.1: *Conceptual correspondendes in database forward engineering and reverse engineering reported by Lin et al. [49]*

	Stand-alone (Yes/No)	Implemented (Yes/No)	Type of Model			Source of information (Metadata, Data, DDL, HTML)
			Source	Target	Uses Meta-model?	
Stojanovic et al	Yes	Yes	RDB	F-Logic	No	DDL
Buccella et al	No	No	RDB	OWL	No	DDL
Astrova – 1	Yes	No	RDB	F-Logic	No	DDL and Data
Astrova – 2	Yes	No	RDB	F-Logic	No	HTML-Forms
Astrova - 3	Yes	No	RDB	OWL	No	DDL
Man Li et al	Yes	Yes	RDB	OWL	No	Metadata & Data
Relational.owl	Yes	Yes	RDB	OWL	Yes	Metadata
RDB2ONT	No	Yes	RDB	OWL	Yes	Metadata
DB2OWL	No	Yes	RDB	OWL	No	Metadata
DataMaster	Yes	Yes	RDB	Protégé-OWL or Protégé-Frames	No[1]	Metadata
Yan/Changrui	Yes	No	RDB	Unknown	Unknown	Unknown
Xu/Li	Yes	No	XML	OWL	Yes	XML data
Automatic Ontology Generator	Yes	Yes	RDB	RDF	No	Unknown
Lubyte/Tessaris	Yes	No	RDB	DLR-DB	No	Metadata
Changjun Hu et al	Yes	Yes	RDB	OWL	No	Metadata
OWLFROMDB	Yes	Yes	RDB	OWL	No	Metadata
DM2OWL	Yes	Yes	RDB & ORDB	OWL	No	Metadata & Data

Figure 3.2: *General properties for the reviewed methods [2]*

	Type of Constructs				
	Table Structure	Primary Key	Foreign Key	Unique	Not-Null
Stojanovic et al	✓	✓	✓	✓	✓
Buccella et al	✓	Partial	✓		✓
Astrova – 1	✓	✓	✓	✓	✓
Astrova – 2	✓				✓
Astrova - 3	✓	✓	✓	✓	✓
Man Li et al	✓	Partial	✓		✓
Relational.owl	✓	✓	✓		
RDB2ONT	✓	✓	✓		✓
DB2OWL	✓		✓		
DataMaster	✓		✓		
Yan-Changrui	✓		✓		
Xu-Li	✓				
Automatic Ontology Generator	✓				
Lubyte/Tessaris	✓		✓	✓	✓
Changjun Hu et al	✓		✓		
OWLFROMDB	✓		✓		
DM2OWL	✓	✓	✓	✓	✓

Figure 3.3: *Type of constructs handled [2]*

OWL constructs	ER/EER construct
`Class`, `rdfs:subclassOf`	Entity
Datatype properties	Simple attributes
Datatype attribute that is: A simple attribute AND with `owl:allValuesFrom` construct AND `minCardinality = 1` AND `maxCardinality = 1` AND all values distinct	Primary key
`Cardinality>1` OR `minCardinality > 1`	Multi-valued attribute
`minCardinality = 0`	Partial participation relationship
`Cardinality = 1`	Full participation relationship
`maxCardinality = 1`	Full participation relationship
`(minCardinality = 0` OR `1)` AND `(maxCardinality = 1` OR `Cardinality = 1)` FOR ObjectProperty AND inverseOf property	Binary 1 : 1 relationship
`(minCardinality = 0` OR `1` AND `(maxCardinality = 1` OR `Cardinality = 1))` FOR ObjectProperty AND (minCardinality = 0 OR 1 AND (maxCardinality > 1 OR Cardinality > 1)) FOR inverseOf property	Binary 1 : N relationship
`(minCardinality = 0` OR `1` AND `(maxCardinality > 1` OR `Cardinality > 1))` FOR ObjectProperty AND inverseOf property	Binary M : N relationship
`owl:subClassOf`	Specialisation class
`owl:disjointWith` OR `owl:complementOf`	Disjoint relationship
`owl:UnionOf` USED WITH subclasses	Overlap relationship
`Owl:UnionOf` OR `owl:intersectionOf` USED WITH regular classes	Union

Table 3.2: *$OWL \rightarrow ERD$ rules defined in [15]*

captured by OWL cannot be fully represented in a less expressive model like ERD.

As in the $ERD \rightarrow OWL$ transformation, different correspondences rules have been defined in the literature, like by Astrova et al. [8] and Bagui [15] (whose rules are summarized in Table 3.2). Bagui [15] reports the correspondences between OWL constructs and EER (Ehnanced Entity-Relationship model) constructs, that allow to describe specializations and a larger set of restriction than standard ER model. This kind of model is very useful in representing complex databases and in expressing more semantics in database schema.

Those rules in literature are not complete, and they do not contemplate each possible transformation between OWL and ERD constructs. So, despite the huge amount of work towards this direction, the problem of the loss of expressiveness hasn't been solved yet.

Vysniauskas and Nemuraite [75] propose an algorithm as solution for the on-

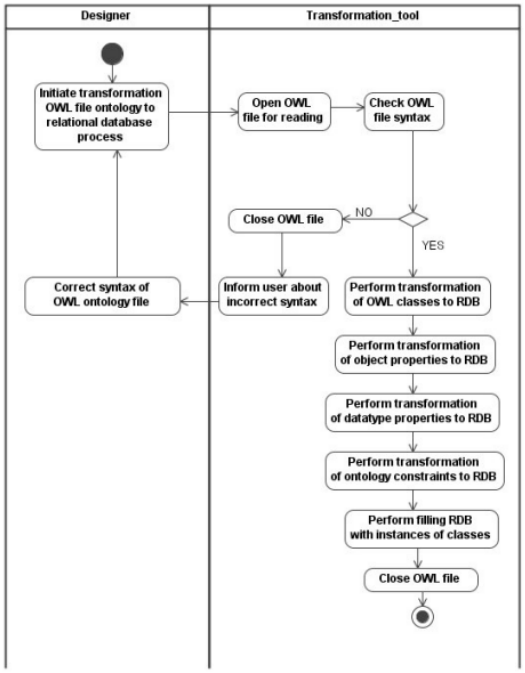


Figure 3.4: *Ontology to database trnsformation algorithm proposed by Vysniauskas and Nemuraite [75]*

tology storing in relational model task. The algorithm (Figure 3.4parses OWL ontology documents and generates DDL scripts containing database descriptions including also OWL constraints.

In [36] the authors provide a solution for a lossless mapping of an OWL ontology to a relational schema and the corresponding instances in order to store the ontology data and execute queries. The system proposed by Gali et al. [36] is composed by three parts:

- *Ontology Modeler* that takes an OWL ontology and creates an ontology model, similarly to DOM[29] for XML document, parses the OWL document, detecting and recording all the constraints.
- *Document Manager* that helps in processing and handling of multiple OWL documents, building the union of imported documents and creating new models the imported OWL documents.
- *Ontology Reasoner* that provides methods for listing, getting and setting the RDF[50] types of a resource.

According to Astrova et al. [9], generally approches based on transformation from OWL ontology to relational database suffer of at least one of the following limitations:

- they ignore value restrictions that capture more semantics;
- they are not implemented or they are semi-automatic;
- they do not address the semantic loss caused by the transformation;
- sometimes, the correspondences provided are not fully accepted by the users, whose ontological structures could mean something different with respect to the output structures.

3.2 Methodologies for mapping heterogeneous sources

The mapping function $m : OWL \cup ERD \rightarrow C$ is a set of correspondences relating constructs in both representations. It assumes, differently from the transformation processes, the existence of both database and ontology, and aims at producing correspondences between elements of both representation, while the transformation processes aim to produce, given one, the other one representation. Among the three processes involved in the alignment scenario, the mapping is the most affected by the expressiveness gap problem, due to its own bidirectionality.

In this section are described some of the main approaches in literature addressed in this way. In particular, Marson [40], Ronto [59] and Coma++ [11] are relevant because they have been used for the comparison with the mapping approach proposed in this thesis.

3.2.1 Similarity Flooding

Similarity Flooding [52] approach is aimed to match different data structures and is based on matching their graph representations. Considering the graph representation producted in a first step from both the inputs, Similarity flooding uses neighbor relations between the elements to find matching correspondences between them.

The idea is that if two elements in heterogeneous schemas are very similar, then their neighboring elements should also be very similar.

The algorithm (Figure 3.5 shows an example) takes the cross-product of all nodes in both ontologies, producing a single node in the *Pairwise Connectivity Graph*, where the edges are produced exclusively if the nodes from the input graphs share an edge.

In the *Propagation Graph* weights are added to the edges. All outbound edges from a given node have equal weight, and sum to one.

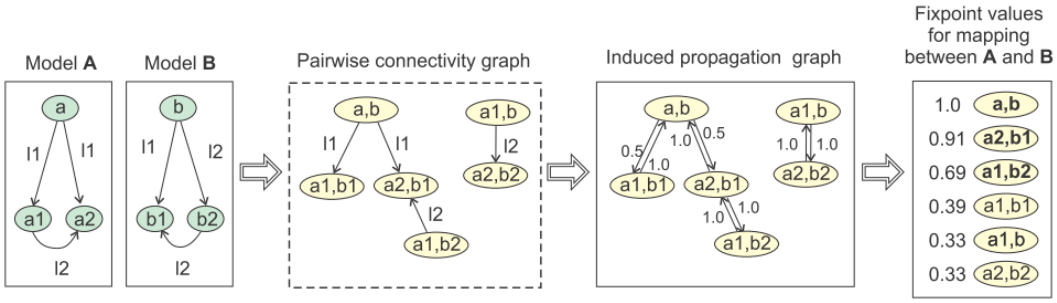


Figure 3.5: *An example reported for Similarity flooding [52]*

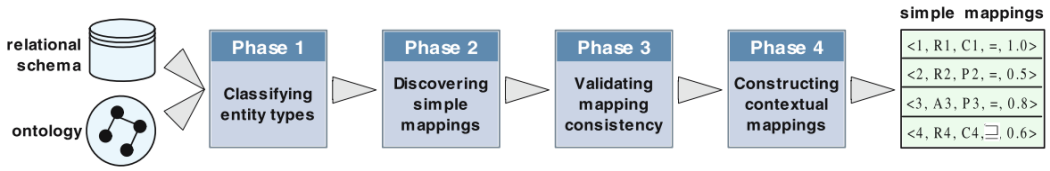


Figure 3.6: *MARSON architecture [40]*

Then an arbitrary initial similarity score is assigned to each node that is refined iteratively. At each iteration the similarity score is increased by the weighted sum of the similarity of all its neighbors in the propagation graph. This computation proceeds until the new similarity converges to a fixed value.

According to Cotterell and Medina [30], one limitation of the Similarity Flooding technique is the necessity that edge labels must be identical, otherwise if the edges of the graph do not have identical labels. Let consider the example where the predicates labeled *"appearsIn"* and *"actsIn"* are used to describe the relationship that a certain actor has been in a movie, such information could be missed by the flooding algorithm.

3.2.2 MARSON (Mapping between Relational Schemas and Ontologies)

Hu and Qu [40] address the problem of espressivity gap between relational databases and ontologies proposing an approach to discover simple mappings between the two representations; a virtual space of documents is built and the consistency of the mappings is validated in the end of the process by considering a relational database as a collection of relation schemas, that consist of names of the relations and names of the attributes with a domain name and a range name for each relation, and a set of integrity constraints; for the input ontology classes($\mathcal{C}$), object properties($\mathcal{P}_{\mathcal{O}}$) and data properties($\mathcal{P}_{\mathcal{D}}$) are classified.

The algorithm (Figure 3.6) follows four stages:

- *Classifying entity types.* The elements of the database are classified into *Strong Entity Relation($\mathcal{SER}$)*, *Weak Entity Relation($\mathcal{WER}$)*,*Regular Relationship Relation($\mathcal{RRR}$)*, *Specific Relationship Relation($\mathcal{SRR}$)*, *Foreign Key Attribute($\mathcal{FKA}$)*, *Non Foreign Key Attribute($\mathcal{NFKA}$)*, according to [27]. Then the elements of database and ontology are heuristically classified into:
  - Group 1: $\{\{\mathcal{SER}\} \bigcup \{\mathcal{WER}\}\} \times \{\mathcal{C}\}$
  - Group 2: $\{\{\mathcal{RRR}\} \bigcup \{\mathcal{SRR}\}\} \times \{\mathcal{P}_{\mathcal{O}}\}$
  - Group 3: $\{\mathcal{FKA}\} \times \{\mathcal{P}_{\mathcal{O}}\}$
  - Group 4: $\{\mathcal{NFKA}\} \times \{\{\mathcal{P}_{\mathcal{D}}\}\} \bigcup \{\mathcal{P}_{\mathcal{O}}\}\}$
- *Discovering Simple mappings.* It is the core phase of the process; the method is inspired by the approach described in [63]: the semantic informations are captured, considering the structures of both database and ontology by constructing virtual documents. A virtual document for an entity represents a collection of weighted tokens derived from the description of the entity itself and its neighbors; the weights indicate the importance and are defined as vectors according to the TF/IDF model [68]. The simple mappings are computed so by calculating the confidence measure between two entities as the cosine value between two TF/IDF vectors.
- *Validating mapping consistency.* The mappings are validated considering the constraints between relations (classes) and attributes (properties). In this phase are also used some inference rules and the compatibility of data type between non foreign key attributes and datatype properties is checked.
- *Constructiong contextual mappings.* It is the last phase of the process and it was developed in data integration application vision, considering sample instances of both the database and ontology and the hierarchies of entities. It construcs so called contextual mappings that could be translated as view-based mappings.

Hu and Qu [40] implemented their approach in Java, calling it MARSON(MApping between Relational Schemas and ONtologies). The experimental results are discussed in the Chapter 7 in comparison with the results of the mapping process proposed in this thesis.

3.2.3 RONTO (Relational to Ontology schema matching)

RONTO [59] (Relational to ONTOlogy schema matching) is an approach for schema matching.

The authors first introduce some intermediate modeling elements such as Candidate Concepts, Candidate Datatype-Property and Candidate Concept Set, then they describe the following six different algorithm that are used to perform the entire matching approach:

- Mapping tables to concepts
- Mapping attributes to Datatype-property
- Mapping foreign-Keys to Object-Properties
- Mapping N:M Relations to Object-Properties
- Mapping Joined Tables to Concepts
- Mapping attributes to Object-Properties

RONTO uses several similarity measures, such as linguistic, semantics and data type compatibility.Those similarity measures are computed like the ones described by Pedersen et al. [60].

The implemented tool is a Protége plug-in. RONTO allows users to follow a step-by-step matching process. The handling database module extracts database meta-data and allows user to view the tree structure of the database, like the structure presented by Protége for the ontological elements. Then, mapping according to the previous algorithms list, RONTO presents to the users the mapping results like lines connecting the elements and labeled with the similarity measure (Figure 3.7).

The experimental results was compared with the MARSON's [40] results.

3.2.4 COMA (Combining Matchers)

COMA (COmbining MAtchers) [11, 33, 12, 51, 65, 3, 4, 5] is the most widespread schema to ontology matching tool and offers a comprehensive infrastructure to solve large real-world match problems. It has been developed at the University of Leipzig and it is still going on.

COMA has been developed in three prototypes: COMA, COMA++ and COMA 3.0. The base concept is to use different matchers evaluating them on the Cartesian product of the elements from the two schemas. COMA project started providing a generic approach for XML and relational schemas. The OWL ontologies were introduced by the COMA++ release; in COMA 3.0 OWL APIs are used to read the OWL ontologies informations that are mapped to a generic model representation based on directed acyclic graph.

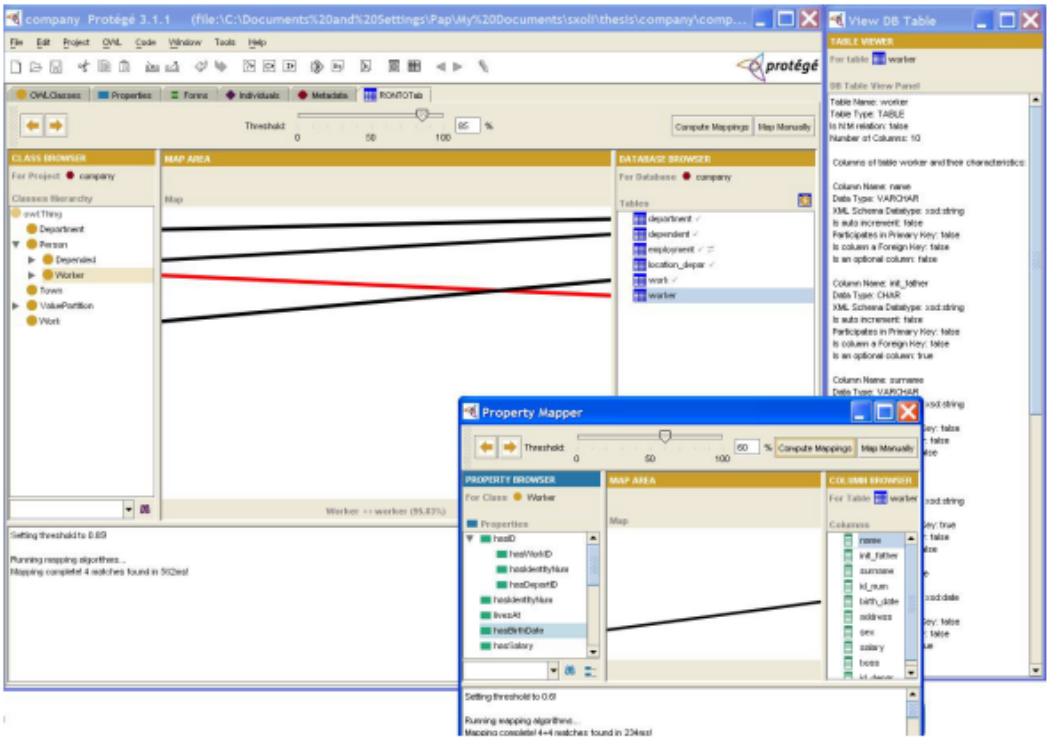


Figure 3.7: *A RONTO screenshot [59]*

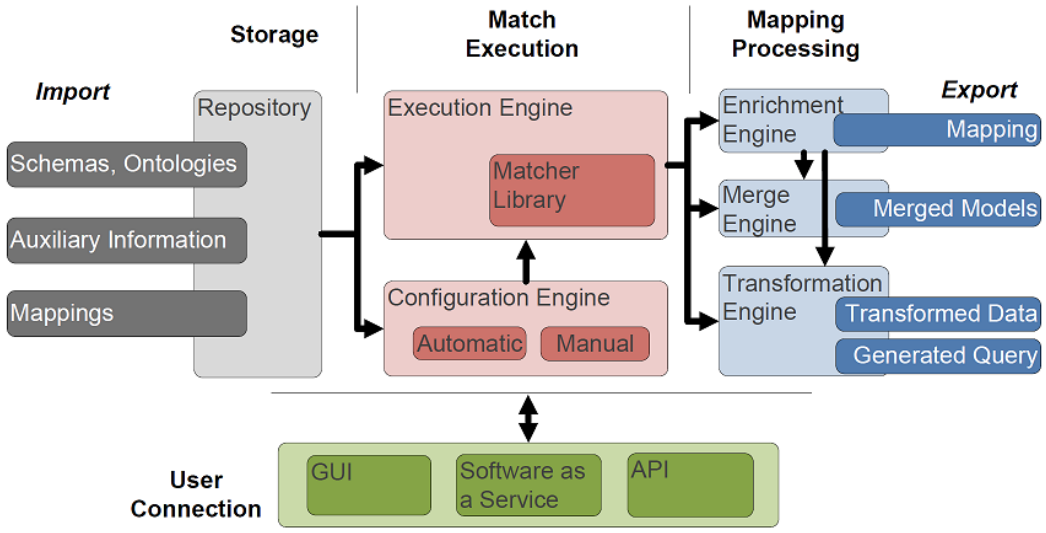


Figure 3.8: *Architecture of COMA 3.0 [51]*

The COMA project evolution is listed in what follows[1]:

- 2002: First release of the schema matcher COMA. COMA stands for combined matching and offers a suite of several matching strategies.
- 2005: Release of COMA++, which now offers ontology matching and fragment-based matching. COMA++ also comes with a GUI to allow a much more comfortable schema matching.
- 2008: Release of the 2008 version of COMA++, which supports instance matching. Also, a web-edition is developed, containing the prime features of COMA++.
- 2011: Release of COMA 3.0, which is a redesigned version of COMA++. It offers ontology merging and an enhanced workflow management.
- 2012: The Community Edition of COMA 3.0 becomes an Open Source Project under AGPL license.

In this thesis we refer to COMA 3.0, whose architecture is depicted in the Figure 3.8.

The system is composed by four modules:

- *Storage*. Allows to store schemas, ontologes, existent mappings that can be reused and other of informations such as instances, dictionaries etc.

[1]http://dbs.uni-leipzig.de/Research/coma.html

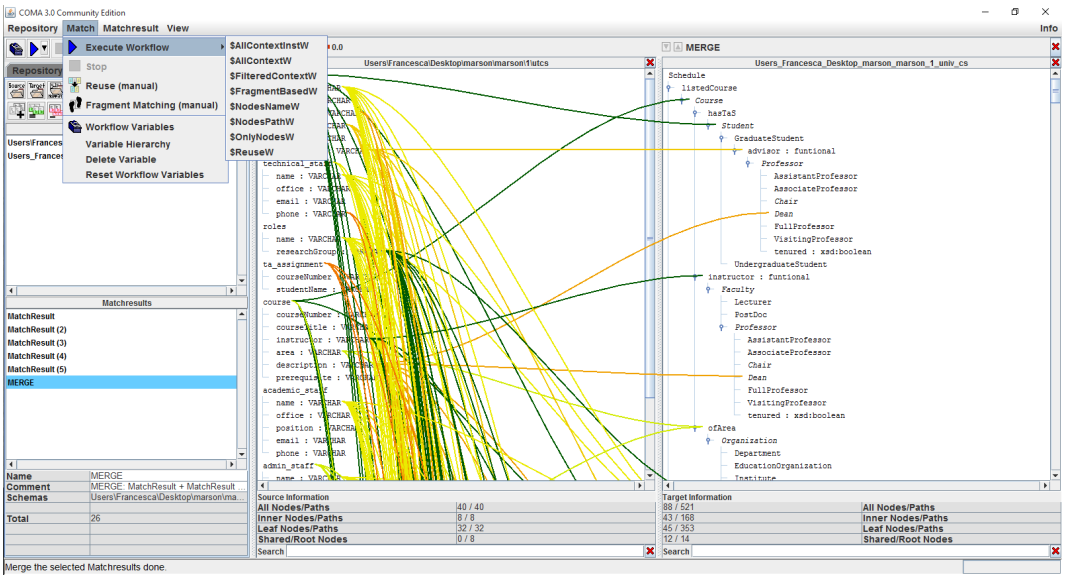


Figure 3.9: *COMA 3.0 user interface*

- *Match Execution.* It is the core module of COMA; the *Configuration engine* allows to choose either a manual or automatic configuraton. The automatic configuration uses the default match strategy as starting point and then uses a set of consideration about the schemas/ontologies to decide about the matchers and how combine them. The *Execution engine* performs the preprocessing, determining schemas components, apply the matching strategies using several matcher libraries, and performs the postprocessing step combining the partial results. The resulting mappings can be used for a new iteration.

- *Mapping Processing.* This module is developed in the Business version; it supports the automatic enrichment of the mappings, through the *Enrichment module*, that allows to enrich the simple correspondences creating more complex mapping expressions. The *Merging Engine* allows to perform ontology merging tasks, whose main approach called ATOM(Automatic Targeg-driven Ontology Merging) is described in [64] and the *Transformation Engine* allows to create queries as executable mappings in data transformation tasks.

- *User Connection.* COMA provides a GUI (Figure 3.9) to improve the interaction between users and the matching system and helps expert user to configure manually the matching process.

COMA was well evaluated in OAEI [2] competition. Like MARSON and RONTO,

[2]http://oaei.ontologymatching.org/

COMA's results are compared with the results of the mapping prosess described in this thesis.

3.2.5 BootOX (Bootstrapper Oxford)

BootOX [43], is an ontology and mappings bootstrapper, developed with the aim of giving more flexibility for specific purposes.

Generally BootOX allows to define different profiles for different applications; the bootstrapped ontology can also be used in OBDA scenarios. Moreover, BootOX allows to import domain ontologies and they can be integrated with the bootstrapped ontology to extend it through alignment methods or otherwise directly mapped in a database. The main ontology alignment system considered for this purpose is LogMap [42, 41], that aligns two ontologies deriving OWL2 equivalence and axioms between entities form both ontology vocabularies using lexical characteristics of terms and the structure of the ontologies, in respect of consistency and conservativity principles [70].

The bootstrapping method in BootOX starts from such considerations as Closed-World (CWA) vs Open-World (OWA) assumptions, that form the basis of the semantic gap problem. By the way, BootOX authors considered a set of translation rules, whose translate in summary:

- each (non-binary) table into OWL Class;
- each attribute not involved in a foreign key into an OWL database property;
- each foreign key into an OWL Object Property.

The rules in Figure 3.10 are used to create the ontological vocabulary. In order to generate mappings from a relational database to an ontological vocabulary, BootOX [43] follows the W3C guidelines and relies on R2RML language to produce direct mappings and it offers a suite of advanced techniques, heavily relying on interaction with the users, for the bootstrapping of complex R2RML mappings that are beyond the direct ones.

R2RML[3] (RDB to RDF Mapping Language) is defined a language for expressing mappings from a relational database to RDF datasets. R2RML mappings are themselves RDF graphs and are written down in Turtle syntax.

Furthermore, the direct mappings can also be extended with metainformations about provenance[4], such as mapping assertions cointaining for instance the source database from wich the information is extracted; the provenance comes into three different granularity levels, whose convenience depend on the intended use of the

[3]http://wwww.w3.org/TR/r2rml/.

[4]Based on the W3C recommendation PROV-O http://www.w3.org/TR/prov-o.

#	RDB feature	Ontology feature	OWL 2 axiom	OWL 2 profile		
				QL	RL	EL
(1)	**Non-binary Relation / Table** T	A class C_T for the non-binary table	Class: C_T	✓	✓	✓
(2)	**Binary Relation or Many-to-Many Table** referencing tables T_1 and T_2	A property P (and its inverse Q) associated to the classes C_{T_1} and C_{T_2} with local and/or global constraints	ObjectProperty: P	✓	✓	✓
			Q InverseOf: P	✓	✓	-
			P Domain: C_{T_1}	✓	✓	✓
			P Range: C_{T_2}	✓	✓	✓
			C_{T_1} SubClassOf: P some C_{T_2}	✓	-	✓
			C_{T_1} SubClassOf: P only C_{T_2}	-	✓	-
(3)	**Data attribute** in table T of (sql) type t	A property R_a associated to the class C_T and datatype d_t with local and/or global constraints.	DataProperty: R_a	✓	✓	✓
			R_a Domain: C_T	✓	✓	✓
			R_a Range: d_t	✓	✓	✓
			C_T SubClassOf: R_a some d_t	✓	-	✓
			C_T SubClassOf: R_a only d_t	-	✓	-
(4)	**Foreign Key** in table T_1, referencing T_2, **no intersection with** or **strict subset of** primary key	A property P_f associated to the classes C_{T_1} and C_{T_2} with local and/or global constraints	ObjectProperty: P_f	✓	✓	✓
			P_f Domain: C_{T_1}	✓	✓	✓
			P_f Range: C_{T_2}	✓	✓	✓
			C_{T_1} SubClassOf: P_f some C_{T_2}	✓	-	✓
			C_{T_1} SubClassOf: P_f only C_{T_2}	-	✓	-
(5)	**Foreign Key is the primary key** in T_1, ref. T_2	Class C_{T_1} is subsumed by class C_{T_2}	C_{T_1} SubClassOf: C_{T_2}	✓	✓	✓
(6)	**Foreign Key** in table T **referencing the same table**	A property P_f associated to class C_T with a self-restriction. The property P_f may also be declared with several characteristics	C_T SubClassOf: P_f some C_T	✓	-	✓
			C_T SubClassOf: P_f some Self	-	-	✓
			P_f Characteristics: Transitive	-	✓	✓
			P_f Characteristics: Symmetric	✓	✓	-
			P_f Characteristics: Reflexive	✓	-	✓
(7)	**Primary Key** or **Unique constraint** in table T_1 on a **foreign key** fk referencing T_2	Key axiom for class C_{T_1} and property P_f. P_f is associated to (local and/or global) cardinality constraints	C_{T_1} HasKey: P_f	-	✓	✓
			P_f Characteristics: Functional	-	✓	-
			P_f Characteristics: InverseFunctional	-	✓	-
			C_{T_1} SubClassOf: P_f exactly 1 C_{T_2}	-	-	-
			C_{T_1} SubClassOf: P_f max 1 C_{T_2}	-	✓	-
			C_{T_1} SubClassOf: P_f some C_{T_2}	✓	-	✓
(8)	**Primary Key** or **Unique constraint** on a **data attribute** a of (sql) type t in table T	Key axiom for class C_T and data property R_a. R_a is associated to (local and/or global) cardinality constraints.	C_T HasKey: R_a	-	✓	✓
			R_a Characteristics: Functional	-	✓	✓
			C_T SubClassOf: R_a exactly 1 d_t	-	-	-
			C_T SubClassOf: R_a max 1 d_t	-	✓	-
			C_T SubClassOf: R_a some d_t	✓	-	✓
(9)	**Composed Primary Key** in table T	Key axiom for the class C_T and the data and object properties involved in primary key	C_T HasKey: $R_1 \dots R_n\ P_1 \dots P_n$	-	✓	✓
(10)	**Not Null Constraint** on a **data attribute** in T, type t	Existential or cardinality restriction over R_a in C_T	C_T SubClassOf: R_a min 1 d_t	-	-	-
			C_T SubClassOf: R_a some d_t	✓	-	✓
(11)	**Not Null Constraint** on a **foreign key** in T_1, ref. T_2	Existential or cardinality restriction over P_f in C_T	C_{T_1} SubClassOf: P_f min 1 C_{T_2}	-	-	-
			C_{T_1} SubClassOf: P_f some C_{T_2}	✓	-	✓
(12)	**Check Constraint** on data attribute a of type t in table T **listing posible values** $v_1 \dots v_n$ (n ≥ 1)	Enumeration of literals: in a restriction in class C_T and/or as a range of Ra. Alternatively, one could create subclasses for each of the values	C_T SubClassOf: R_a some $\{v_1 \dots\}$	-	-	✓*
			C_T SubClassOf: R_a only $\{v_1 \dots\}$	-	-	-
			C_T SubClassOf: R_a value v_1	-	-	✓
			R_a Range: $\{v_1 \dots\}$	-	-	✓*
			C_{Tv_i} SubClassOf: C_T	✓	✓	✓
(13)	**Check Constraint** on attrib. a in table T **restricting numerical range** of t	Datatype restriction: in a class restriction in C_T and/or as a range of Ra	C_T SubClassOf: R_a some $d_t[> x]$	-	-	-
			C_T SubClassOf: R_a only $d_t[> x]$	-	-	-
			R_a Range: $d_t[> x]$	-	-	-
(14)	**Several data attributes** $a_1 \dots a_n$ **in different tables** $T_1 \dots T_n$ **with the same name and type** t	Group properties $R_1 \dots R_n$ under new superproperty R_a or merge $R_1 \dots R_n$ into new property R_a	R_i SubPropertyOf: R_a	✓	✓	✓
			R_a Domain: C_{T_1} *or* $\dots$ *or* C_{T_n}	-	-	-
			C_{T_i} SubClassOf: R_a some d_t	✓	-	✓
			C_{T_i} SubClassOf: R_a only d_t	-	✓	-
(15)	T_1 **and** T_2 **not involved in a inheritance relationship**	Class C_{T_1} is disjoint with class C_{T_2}	C_{T_1} DisjointWith: C_{T_2}	✓	✓	✓

Figure 3.10: *Rules used in BootOx [43] for ontological volcabulary creation. When not stated the contrary, a class C_T represent a Table T, an object property P_f a data*

information: URI level, Triple level, Graph level. These metainformations can be used for many purposes such as providing a richer query answering interface, identifying faulty ontology axioms or mappings and so on.

3.3 Probabilistic approaches to ontology alignment

In ontology alignment field, there are many approaches, based on probabilistic considerations.

An example is in [54]. Mitra et al. [54] introduce OMEN (Ontology Mapping ENhancer) that is a probabilistic method for enhancing mappings by using a Bayes Net to represent the influences between concepts across the ontologies and a set of meta-rules based on the semantics of the ontology relations. For instance, one of the meta-rules asserts that if two nodes, that tipically represent concepts in the ontology graph, match and they have two arrows (that is a kind of property in ontological terms) coming out of these nodes, then the probability that the other nodes at the other end of the arrows match is greater.

Although they are focused on ontology-to-ontology alignment it could be interesting to analyze them thinking how to use a similar approach where heterogeneous sources are involved in alignment task. As it has been reported in subsection 3.2.5, using a bootstrapping teqniques, they can be used for aligning heterogeneous sources yet despite a possible increasing of inconsistencies and computational costs.

In the following subsections are described two probabilistic approaches for the ontology alignment task.

3.3.1 PARIS (Probabilistic Alignment of Relations, Instances and Schema)

PARIS [71] (Probabilistic Alignment of Relations, Instances and Schema) is an algorithm for the automatic alignment of RDFS ontologies. It is based on a probabilistic framework taking advantage of schema alignment and instance matching, thus providing a holistic solution to the ontology alignment problem.

The authors of PARIS transform the equations of their equality model into probability assessments assuming mutual independence of all distinct elements of models, assuming that an ontology has not to contain equivalent resources. Such an assumption is not true in practice but it allows them to approximate efficiently the probability of the consequence of their alignment rules.

The algorithm implementation is composed by three steps: the equivalence probabilities of the instances are computed first, then the probabilities for sub-

relationships. In a third step the equivalences between classes are computed.

PARIS do not use any kind of heuristics on relation names and cannot deal with structural heterogeneity. So it is not able to find matches when an ontology models an event by a relation while the other one with an entity. The same when one ontology is more fine-grained then the other one.

3.3.2 Cotterell and Medina's approach for Ontology Alignment using Markov Model

The approach proposed by Cotterell and Medina [30] is quite interesting, although it is focused on ontology alignment, because it uses a Markov Model for the alignment. The method is thinked as an improvement of Similarity Flooding.

The authors proposed the use of a lexical measure to estimate the similarity between predicate levels, using an arbitrary threshold; when the similarity between two predicates exceeds the threshold they mean the same thing. The algorithm computes the edge confidence, that is defined as the similarity score between two labels of the two edges, being that similarity score is greater than the threshold, then an NPMC (Normalized Pairwise Markov Chain) with a row-stochastic transition matrix is created, creating a directed graph between pairs of concepts.

The aim is to have a connections net of pairs of ontological terms with other pairs of ontological terms, weighted proportionally to similarity of edges in the input ontologies.

3.4 OBDA systems

In this section two well-known solutions in the OBDA research field are described. MASTRO and ONTOP are two projects developed by italian reseach groups and are still going on.

MASTRO[26] is a tool for the Ontology Based Data Access (OBDA) developed at the University of Rome "La Sapienza" and at the Free University of Bozen-Bolzano. So, the main scope of MASTRO is to allow users to manually design and manage systems in which an ontology is connected to external data sources through mappings. Indeed, the mappings can be used to specify the semantic correspondences between a unified view of the domain (called global schema in data integrationterminology) and the data stored at the sources.

The theoretical background of MASTRO is strong (see for details [20, 22, 23, 62, 24]).

The ontologies managed by MASTRO are specified in $DL - Lite_{A,id}$, a logic of the *DL-Lite family* of tractable *Description Logics (DLs)*, secifically tailored for ontology querying and managing. $DL - Lite_{A,id}$ captures the main modeling

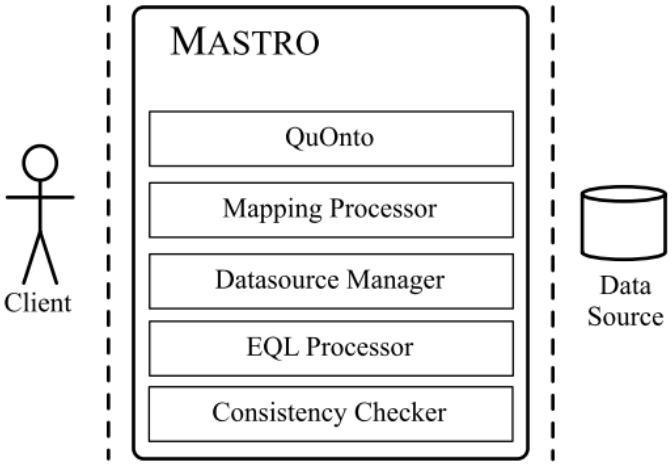


Figure 3.11: *Architecture of MASTRO [26]*

features of a variety of representation languages, such as basic ontology languages and conceptual data models. The use of EQL constraints allows to specify integrity constraints and ensures the tractability of reasoning. The mapping adopted by MASTRO allows for solving the so called impedance mismatch problem, arising from the fact that data sources store values, while the ontology contains objects (the instances of concepts).

Taking account that user queries over the DL ontology $< \mathcal{T}, \mathcal{A} >$, where $\mathcal{T}$ is the TBox (the intensional knowledge) and $\mathcal{A}$ is the ABox (the extensional knowledge), are defined as *conjunctive queries (UCQs)* and that generally an OBDA system can be defined as $< \mathcal{T}, \mathcal{M}, \mathcal{D} >$, where $\mathcal{T}$ is a TBox, $\mathcal{M}$ is a set of mapping rules typically defined manually by the user, and $\mathcal{D}$ is the database, the architecture of MASTRO is depicted in the Figure 3.11 and it is composed by five steps:

- *QuOnto module.* QuOnto is a reasoner that provides intensional reasoning service, reformulating an UCQ with respect to $\mathcal{T}$. The reformulation engine uses the `PerfectRef` algorithm presented in [23] and further reducing as is possible the number of generated queries.

- *Mapping Processor.* Provided that MASTRO does not manage directly the ABoxes, the queries are not evaluated over an ABox. They need to be reformulated, according to the mappings, thus obtaining a query that can be evaluated over the data sources. The new reformnulation step is called *query unfolding* and it is the main scope of the Mapping processor module.

- *Datasource Manager.* This module is responsible for maintaining connection and managing the database resources.

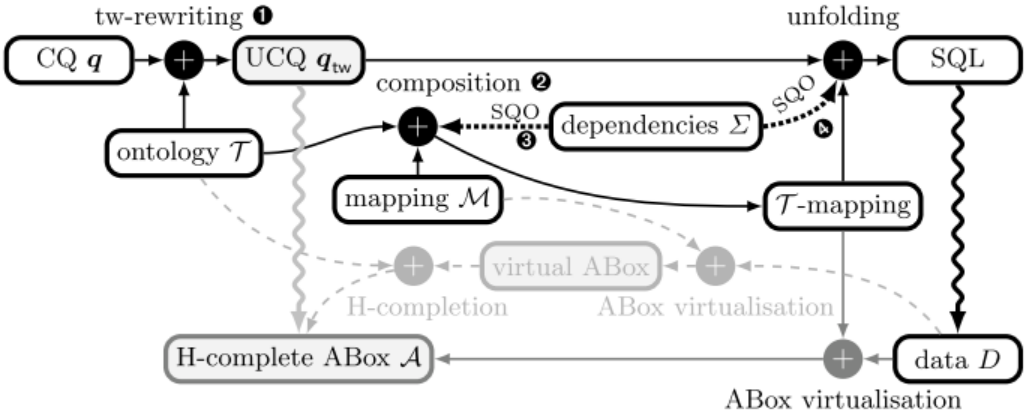


Figure 3.12: *Architecture of ONTOP [66]*

- *EQL Processor*. This module allows to specify and execute EQL queries, they allows a major semantic expressiveness than SQL ones. The goal of this module is to rewrite each query in SQL query over the sources.
- *Consistency Checker*. The final step task is to verify the satisfability of the EQL constraints and localize data that violate TBox assertions over the ontology.

MASTRO is released in three-level interfaces: Java API, an OWLAPI compatible interface and a plugin for Protégé ontology editor.

ONTOP[14, 66] is an open-source OBDA framework developed at the Free University of Bozen-Bolzano. It is actually considered the first system supporting all the W3C recommendations about query languages, and also it supports the major commercial and free databases.

The core of the ONTOP process, is on using the SPARQL engine QUEST that allows to rewrire SPARQL queries to SQL queries. The process is similar to MASTRO one, based on chained rewriting steps. The architecture is showed in the Figure 3.12.

ONTOP provides a plugin for Protégé, where the user can manually specify her mappings in R2RML documents and control the results; it can also be used as Java library and a SPARQL end-point. It makes the bootstrap of the database as BootOX.

Chapter 4

Theoretical background

This chapter has the goal to introduce the theoretical arguments useful for the comprehension of the proposed approach. So the HMMs (Hidden Markov Models), the CFG (Context-Free Grammar) and the similarity measures will be explained.

4.1 Hidden Markov Model

The Hidden Markov Model (HMM) [17] is a well known mathematical representation for stochastic processes where the system being modeled is assumed to be a Markov process whose states are hidden, so they are not directly visible, while only a sequence of output values are visible; outputs are called *observations*.

In other words, a sequence of *emissions* is observed, and it gives some information about the sequence of *hidden states*. Analyses of HMMs try to recover this hidden sequence from the observed data.

A Markov process is characterized by the *Markov property* that is the probability of being in a state x_t at a given time step t depends only on the previous state of the system x_{t-1}. The Markov property can be stated in terms of the conditional probability of a given state x_t at the time step t on each possible state sequence X that is $P(x_t|X) = p(x_t|x_{t-1}) = a_{t-1,t}$.

HMMs are a widespread technique in pattern recognition when temporal processes are to be modeled such as handwriting, speech, and gesture recognition.

In particular, several studies have been carried out in the past decades, which were concerned in Part Of Speech (POS) tagging using HMM [32, 28]. Formally, a HMM is a 5-tuple $< H, O, A, B, \pi >$ where:

- $H = h_0, h_1, ..., h_I$ is the *hidden states space*, and I is its dimension. Each element $h_i \in H$ represents a possible hidden state. The hidden states set is

the real estimated sequence of hidden states in correspondence of a thread of observation;

- $O = o_0, o_1, ..., o_J$ is the *observable states space*, and J is its dimension. Each observable state $o_j \in O$ represents a possible observation. The observable states set is the real sequence of emissions in a thread of observation;
- $A = [a_{ij}]$ is the $I \times I$ matrix of *transition probabilities* describing the likeliness of moving from a hidden state h_i to a state h_j, so $a_i j = p(h_j|h_i)$;
- $B = [b_{ij}]$ is the $I \times J$ matrix of *emission probabilities* describing the likeliness of observing b_j when the hidden state is h_i , so $b_{ij} = p(b_j|h_i)$;
- π is the set of *initial probabilities*, which describe the inherent likeliness of being in a certain state, and are the starting point of the Markov process.

Given a set of observed sequences and the state space,a HMM can be used for learning both A and B distributions. Otherwise, one can be interested in devising the *most likely explanation* of a given sequence of observations $Y = y_1, ..., y_T$ that is the most probable hidden states sequence $X = x_1, ..., x_T$, which generated Y. In our work we're concerned in the second scenario so we want to compute:

$$\max_X P(X|Y) \tag{4.1}$$

Using the *Bayes rule*, and neglecting the evidence P(Y) our goal is:

$$\max_X P(Y|X)P(X) \tag{4.2}$$

The most likely explanation task in HMM is addressed using the *Viterbi algorithm* (Table 4.1), which is based on the concept of Viterbi path. A *Viterbi path* $V_{t,k}$ is the probability of the most probable hidden states sequence, which is responsible for the first t observations, and has x_k as its final state. The algorithm devises an iterative scheme to compute the Viterbi path:

$$V_{1,k} = P(y_1|x_k).\, p(x_k) \equiv b_{1,k}.\, p_k \tag{4.3a}$$

$$V_{t,k} = b_{t,k}.\, \max_{x \in S}(a_{x,k} V t - 1, x), t = 2, ..., T \tag{4.3b}$$

The state sequence can be obtained using a backward procedure where each state is retrieved using a proper function $F(x_t, t)$ as the one that maximized equation 4.3 at each step in the forward Viterbi path:

$$x_T = \arg\max_{x \in S}(V_{T,x}) \tag{4.4a}$$

$$x_{t-1} = F(x_t, t) := \begin{cases} \arg\max_{x \in S}(V_{t,k}) & t > 1 \\ k & t = 1 \end{cases} \tag{4.4b}$$

for states e from 1 **to** N **do**
initialize $vb[e,1] \leftarrow a_{0,e} * b_{e,1}$
initialize $bp[e,1] \leftarrow 0(startstate)$
end for
for construct o from 2 **to** T **do**
for state e from 1 **to** N **do**
$vb[e,o] \leftarrow \max_{e'=1}^{N} vb[e',o-1] * a_{e',e} * b_{e_o}$
{score=the last max value * transition * emission}
$bp[e,o] \leftarrow \arg\max_{e'=1}^{N} vb[e',o-1] * a_{e',e}$
back-pointer = the last max value
end for
end for
$vb[e_F,T] \leftarrow \max_{e=1}^{N} vb[e,T] * a_{e,e_F}$
$bp[e_F,T] \leftarrow \arg\max_{e=1}^{N} vb[e,T] * a_{e,e_F}$
return (bestpath)
{derived following the back-pointers from (e_F,T) to e_0}

Table 4.1: *The Viterbi algotithm*

4.2 CFG (Context-free grammar)

A Context Free Grammar (CFG) G is a 4-tuple $G =< N,T,\mathfrak{R},S >$, where:

- N is a finite set of non-terminal symbols so that each element in N is a variable that can be replaced by other symbols (either a terminal or a non-terminal one);
- T is a finite set of terminal symbols, so that $T \cap N = \emptyset$. Such symbols are the words of a sentence, and T is usually considered the vocabulary of the language defined by G;
- $\mathfrak{R}$ is the set of derivation rules $\mathfrak{R} : N \rightarrow (N \cup T)$ that specify a possible substitution of the non-terminal in the left-hand side of each rule with the expression in the right-hand side. A derivation rule defines the parse tree for a sentence of the language. In the CFGs there is a parse tree for each sentence. Usually rules in $\mathfrak{R}$ are expressed in the Backus-Naur Form (BNF) [13];
- S is the start symbol of the grammar and represents the root of each parse tree.

CFGs are used to represent a knowledge domain in order to provide support for its lexical description. A lexical description is the set of all the sentences that can be expressed applying the derivation rules.

4.3 OWL (Web Ontology Language)

The Web Ontology Language (OWL) 2 [55] is the semantic language designed to represent the web taxonomies of things with properties and relations between them. The main components of an OWL 2 ontology are the axioms, which are statements defining true facts of the represented domain.

Axioms in OWL 2 can be stated about declarations, classes, object or data properties, datatype definitions, keys, assertions and annotations, and they are expressed by means of the FSS syntax. However, the primary syntax that all the OWL compliant tools must support is RDF/XML [18]; this syntax provides an XML representation of the RDF Graph of the taxonomy. The specification in [18] therefore provides a bidirectional mapping from the OWL FSS to RDF Graphs.

4.4 Word similarity measures

In the mapping model proposed in this thesis labels comparison is necessary for estimating to what extent one can match two lexical descriptions of different representations. The similarity measure among labels addresses the matching. Two labels can represent the same thing and have the same meaning even if they are syntactically dissimilar. Also, considering that the involved representations are related to the same domain, syntactic comparison is as much important. In view of these considerations, a procedure for evaluating labels similarity must be implemented and involves either syntactic and semantic aspects.

In the described approach we refer to two well-known distances, the Jaro-Winkler distance [76] for the syntactic similarity metric, and the Wu-Palmer distance [77] for the semantic similarity metric.

4.4.1 Syntactic similarity measure: the Jaro-Winkler distance

The Jaro-Winkler distance allows computing the syntactic similarity between a couple of strings: in particular, given two labels l_1 and l_2 , being $|l|$ the number of chars of label l, the Jaro-Winkler approach defines:

- a match window w so that:

$$w = \lfloor (\frac{\max(|l_1|, |l_2|)}{2} \rfloor - 1 \tag{4.5}$$

  that represents a maximum number of chars to be searched for to match the next character in l_1 to the ones in the l_2;

- the number m of *matching chars* inside the match window;

- the number t of matching chars (but in different sequence order) divided by 2.

The Jaro-Winkler distance $d_{jw}(l_1, l_2)$ between labels l_1 and l_2 is:

$$d_j w(l_1, l_2) = \begin{cases} 0 & m = 0 \\ \frac{1}{3}(\frac{m}{|l_1|} + \frac{m}{|l_2|} + \frac{m-t}{m}) & otherwise \end{cases} \tag{4.6}$$

The higher the Jaro-Winkler distance is, the more similar the labels are.

The Jaro-Winkler distance is best suited for short strings such as person names, and consequently it is a good choice for the labels in an OWL ontology.

The measure is usually normalized such that 0 means no similarity and 1 is an exact match.

4.4.2 Semantic similarity measure: the Wu-Palmer distance

The Wu-Palmer distance computes the similarity between two concepts c_1 and c_2 by looking at the path length between them in the WordNet [53] taxonomy. The Wu-Palmer approach defines:

- $len(c_i, c_j)$ as the function that computes the length of the shortest paths from c_i to c_j measured by counting the edges between nodes;

- $lcs(c_i, c_j)$ as the function that computes the common subsumer (i.e. the most specific common super-class in the taxonomy) such that $len(c_i, c_j)$ is minimal;

- $depth(c)$ as the depth from the root of the taxonomy to the concept c.

The Wu-Palmer distance d_{wp} , which evaluates the path length between two concepts c_1 and c_2 in WordNet, is:

$$d_{wp}(c_1, c_2) = \frac{2X}{Y + Z + 2X} \tag{4.7}$$

where

$$X = depth(lcs(c_1, c_2))$$
$$Y = len(c_1, lcs(c_1, c_2))$$
$$Z = len(c_2, lcs(c_1, c_2))$$

Such as the Jaro-Winkler one, the Wu-Palmer distance is normalized among 0 and 1; the score can never be zero because the depth of the root of a taxonomy is one. The score is 1 if the two input concepts are the same. Lin [48] demonstrate that the Wu-Palmer measure can be computed simply and has very good performances, while remaining as expressive as the others.

Chapter 5

Methodology

This chapter describes the methodology proposed face to the problem introduced in the previous chapters and the main issue of this thesis: bridging the semantic gap in schema alignment.

As stated, the problem consists in addressing three tasks both in transformation or mapping sense.

The methodology is outlined in the same way, sketching a solution for any sigle problem one by one. The three parts can be used indipendently to address any singular problem or can be used togehter in a more complete alignment process and can be summarized as follows:

- $ERD \rightarrow OWL$ *transformation and correspondences set definition.* The main goal of this part is to translate each construct of the Entity-Relationship Diagram (ERD) with a specific OWL ontology construct. Many works in literature address this problem providing specific procedures; so it is possible to use those procedures to perform the transformation. In addition, it is useful to define some $ERD \leftrightarrow OWL$ correspondences to perform a linguistical validation task; such procedures can be used also in the converse sense of transfomation. Details are discussed in the Section 5.1.

- $OWL \rightarrow ERD$ *transformation.* In this part each construct of the ontological description level needs to be translated to a specific ERD construct. Operating in this sense, a translation that uses a set of simple transformation rules or a deterministic procedure, as it is done in the complementary task, can be quite inaccurate, so the methodology considers the uncertainty of informations and aims at producing automatically the *most likehood sequence* that approximates each OWL ontology input. The idea is to consider only the structural features of the input ontology, a generalized grammar is used to represent any possible relational schema and a Hidden Markov

Model (HMM) is used for modeling the system. Details of the approach are described in the Section 5.2.

- *$OWL \leftrightarrow ERD$ mapping.* The method aims at helping user when she defines the mapping rules, i.e. for OBDA purposes. The core component is an Hidden Markov Model like for the $OWL \rightarrow ERD$ transformation, but the labels and the specific ERD constructs are considered as states of the model. The main characteristics of this approach are: a *customized ERD grammar (C-ERD)* defined starting from the input ERD; emission probabilities taking into account both structural and semantic informations; computing transition probabilities are computed using a method based on the definition of the so called *transition tree*. Section 5.3 is due to clarify the details of this approach.

5.1 $ERD \rightarrow OWL$ transformation and correspondences set definition

The $ERD \rightarrow OWL$ transformation is a well known process. The works in literature, reported in Chapter 3 propose some procedures and devise rules for realizing the transformation. The methodology used for the purposes of this thesis applies a reverse engineering process in order to extract the conceptual model of a database like the one described into [6] and considers the approaches described by Fahad [35], Myroshnichenko and Murphy [56], from which is possible to extract the following set of rules:

- Each `Entity` in the ERD representation is translated into an OWL `Class`;
- In general an attribute can be transformed into a `DatatypeProperty`; three kinds of attribute can be found in an ERD:
  - A simple attribute is translated into a datatype tagged as *functional*, cause in OWL a simple datatype can have more values;
  - It is possible to map any single attribute that composes the composite attribute as single simple attributes, o with a set of `FunctionalProperty`. A second way is to map the attribute with a `DatatypeProperty` and the attributes that compose the attribute as sub-properties of the datatype;
  - A multivalue attributes is transformed into a `DatatypeProperty`;
- A primary key is also an attribute so it can be transformed together as a `Datatype` tagged as functional and a `InverseFunctionalProperty`. In this way the transformation grants the univocity of the key;

- A specialization, so an `IsA` relationship is transformed into a `SubClassOf` (a specialization of the superclass Thing is an ERD `Entity`);
- The relationships between entities in ERD typically are transformed into object properties between classes in OWL. Relationships can be devised into many subcategories:
  - Bidirectional relationship are transformed into two object properties, each one being the inverse of the other one, because an ObjectProperty is defined unidirectional. For each `ObjectProperty` a `Domain` and a `Range` need to be defined, that to the involved classes;
  - The translation of an 1-N recursive unary relationship is composed by an `ObjectProperty` that refers to the same `Class` and a `Restriction` on the values of the `Class` involved;
  - M-N unary relationship is transformed into an `ObjectProperty` referencing to a `Class` and another `Class` containing a `Restriction` on the values of the first one.
  - A relationship with an attribute is transformed into a `DatatypeProperty` on an `ObjectProperty`.
  - A one-to-many relationship (or in the opposite direction a many-to-one relationship) is transformed into an ObjectProperty and another `ObjectProperty` defined as inverse of the first one. Each property has as `Domain` and `Range` two classes with resrictions on values. The cardinality is defined into the `Restriction`;
  - Many-to-many relationships have not a direct transformaion, so they are traslated splitting each one into a one-to-many relationship and a many-to-one relationship and applying the previous rule.

It may be useful to define a method that allows to reconstruct the original input ERD. A process of this kind is implemented in the system described in section 5.1.1.

As seen in Chapter 3 many other authors try to define deterministical rules for both transformation and mapping but there is not a sole approach convering all the $OWL \leftrightarrow ERD$ correspondences; so for the purposes of this thesis the rules in literature have been collected and summarized in an unique suitable set of possible correspondences, reported in Table 5.1.

All the works in the literature have been considered allow to derive a possible correspondence between the two representations, in the sense of both transformation and mapping.

The process of defining those rules may be anyhow ambiguos for several reasons: the authors of works in literature describe a procedure through a qualitative description of the translation rules so it is necessary to interpret it; for some constructs can exist many possible translations or they need a set of complex operations on the database and the specific semantic context can influence the translation. The set of rules can be continuosly reorganized and expanded: rules that involve construct that are not commonly considered in the works in literature, have been extract directly on observing the behaviour of mapping systems or in manually creating mapping rules between corresponding OWL-RDB couples.

As example, we can consider the axiom `Class (a:Person)`: all the works in literature report the correspondence between the OWL construct `Class` and the ERD construct `Entity` or with a RDB table that applying a *Reverse Engineering Process* is logically ascribable to an Entity; on the other side, if we consider the axiom

```
EquivalentClasses(
                  a:Employee , ObjectIntersectionOf(
                                    a:Person ,
                                    ObjectSomeValuesFrom(a:worksIn , a:Department)
))
```

that indicates that a class *a:Employee* is an equivalent concept of a *a:Person* on some values retrived in the relation called *a:worksIn* that involves the concept *a:Department*; in this case so an `EquivalentClasses` can defines a `Class`, a specification (so a `Relation` or an `IsA` construct in relational model), the `IntersectionOf` and `ObjectSomeValuesFrom` do not have a specific possible translation but they need a more complex relational structure to be satisfied. In the same way a `Class` can contain the information that in a relational schema can be expressed by a `Relation` between two Entities or a Restriction and so on.

In Table 5.1 are presented the rules without redundances, but in the probabilistic methods presented in Sections 5.2 and 5.3 the redundances are particularly useful informations for the probabilistic methods development.

5.1.1 VEBO (Validate ERD by Ontology)

The tool presented in [67] aims to provide a system for ERD linguistical validation through a semi-automatic creation and enrichment of the corresponding ontology.

The approach can be devised into three steps. The first step uses a reverse engineering process that is able to produce the ontology corresponding to the database structure. In the second step the ontology is analyzed to extract any information about its structure and to derive relational objects; in this step the user has the chance to enhance ontology adding some semantic elements and so

ERD Costruct	OWL Construct	
Attribute	AllValuesFrom; AnnotationProperty; DataProperty; DataPropertyRange; DataSomeValuesFrom; InverseObjectProperty; Entity; MaxCardinalityRestriction; Relationship; ValueRestriction; Datatype; MultiValuedDatatype;	AnnotationAssertion; Class; DataPropertyDomain; DataRange; InverseFunctionalProperty; Declaration; EnumeratedDatatype; MinCardinalityRestriction; SingleValuedDatatype; FunctionalDataProperty; DataProperty; WeakEntity;
BidirectionalRelation	Class; Entity; MinCardinalityRestriction	Declaration; MaxCardinalityRestriction;
CompositeAttribute	Class; Entity; MinCardinalityRestriction	Declaration; MaxCardinalityRestriction;
Domain	ObjectPropertyDomain	
Entity	Class; Datatype; DifferentIndividuals; Entity; MultiValuedObjectProperty; Relationship; SubClassOf; SingleValuedDatatype; Instance;	ClassAssertion; Declaration; DisjointClasses; MultiValuedDatatype; OptionalSingleValuedObjectProperty; SingleValuedInverseOfObjectProperty; WeakEntity; EquivalentClasses;
Identifier	Class; Entity; ObjectProperty; MultiValuedObjectProperty;	Declaration; KeyFor; SingleValuedObjectProperty; WeakEntity
IsA	ClassAssertion; Declaration; InverseFunctionalProperty; PropertyAssertion; SingleValuedInverseOfObjectProperty; ObjectProperty; SubClassOf; MultiValuedDatatype;	Datatype; EquivalentClasses; InverseOfObjectProperty; Relationship; MultiValuedObjectProperty; SingleValuesObjecetProperty; SubObjectPropertyOf OptionalSingleValuedObjectProperty;
MultivalueAttribute	AllValuesFrom; Entity; inCardinalityRestriction	Declaration; MaxCardinalityRestriction;
Primarykey	AllValuesFrom; Datatype; Entity; FunctionalDatatypeProperty; MultiValuedObjectProperty; SingleValuedObjectProperty; FunctionalDataProperty; ObjectProperty;	Class; Declaration; FunctionalDatatype; MultiValuedDatatype; OptionalSingleValuedObjectProperty; WeakEntity InverseOfObjectProperty; SingleValuedInverseOfObjectProperty;
Range	DataRange;	ObjectPropertyRange
Relation	AllValuesFrom; Declaration; InverseObjectProperties; InverseOfObjectProperty; MultiValuedObjectProperty; ObjectProperty; ObjectPropertyRange; TransitiveObjectProperty InverseObjectProperty; ObjectMinCardinalityRestriction;	DataPropertyRange; InverseFunctionalObjectProperty; InverseOf; MaxCardinalityRestriction; ObjectMinCardinality; ObjectPropertyDomain; OptionalSingleValuedObjectProperty; FunctionalObjectProperty; MinCardinalityRestriction;

Table 5.1: *Some $ERD \leftrightarrow OWL$ constructs correspondences derived from the literature*

a SQL file containing SQL statements is created. In the last step, changes are applied definitively, by executing SQL statements derived from the mapping rules. The process may be repeated endlessly.

In the second step, to represent the ontological elements as relational objects the method uses some rules, that are the dual rules used into the first step, such as:

- All classes in the ontological representation identify entities in the relational model, entities are then mapped to the database as tables, where the name of each table is the identification name of the entity that is the class name;
- Functional data properties are represented by attributes and so translated into columns of tables corresponding to the domain of the property;
- Object Properties define relationships between entities and in mapping rules are handled as constraints;

5.2 A Hidden Markov Model for the $OWL \rightarrow ERD$ transformation

The approach proposed for the $OWL \rightarrow ERD$ transformation considers the less expressivity of the ERD in respect to the OWL and so it considers the uncertainty of the transformation process. This problem is discussed in the work presented by Pipitone and Pirrone [61] whose implementation is part of the work presented in this thesis and is described in Chapter 6. In [61] the $OWL \rightarrow ERD$ the transformation process is modeled using an *Hidden Markov Model (HMM)* that estimates the most likely sequence of ERD symbols that corresponds to the OWL constructs in an axiom of the given ontology by means of the *Viterbi algorithm*. The input OWL is represented by a set of axioms in functional-style syntax (FSS) [55]. The OWL constructs are the elements contained in any single axiom.

The ERD in output is represented by a set of sentences according to a grammar, described in section 5.2.1 each corresponding to an OWL axiom. The ERD constructs are the symbols of the grammar.

The technique aims to estimate the most likely composition of ERD constructs that correspond to a given sequence of OWL constructs using an Hidden Markov Model (HMM) where the OWL inputs are the observable states, while ERD structures are the hidden states.

The *observable states space* $O = \{o_1, o_2, ..., o_J\}$ contains the OWL construct contained into the input ontology.

The *hidden states space* $E = \{e_1, e_2, ..., e_I\}$ contains a subset of the ERD constructs of the ERD grammar composed by: `Entity`, `IsA`, `Relation`,

`BidirectionalRelation`, `Attribute`, `CompositeAttribute`, `MultivalueAttribute`, `Identifier`, `PrimaryKey`.

The *transition matrix* A of the HMM contains the probabilities to move from one ERD symbol to another one in constructing the output sentence, and has been set up considering the sequential structural representation through the ERD grammar mentioned above.

The *emission matrix* B contains the probability values of observing an ERD symbol is the hidden state while an OWL construct is observed, and it is computed considering the correspondences rules set defined in the Section 5.1.

The labels of the estimated ERD constructs are the same of the observed OWL axioms and they are fitted in the end of the estimation process for concluding the transformation process.

Summarily, given a sequence of OWL constructs observations $Ob = ob_1, ob_2, ..., ob_T$, that is a sequence of the elements contained into a sigle axiom without labels, the probabilistic transformation finds the most probable sequence of ERD constructs $He = h_1, h_2, ..., h_T$ that composes the translation of the input OWL axiom in an ERD sentence.

For further details not included in this thesis, it can be possible to refer to the work in [61]. The set of *initial probabilities* π describes the inherent probability of being in the initial state; hidden state space E is discrete so the probabilities in π are generated from a categorical distribution that is $\pi = \{p_i = p(e = e_i), \sum_i p_i = 1\}$. Cause it is unknow in advance if a particular ERD construct has to be considered more probable than the others because it depends on the observed OWL context. As a consequence, the p_i values are all equal $\forall i$: $p_i = \frac{1}{|T|}$.

5.2.1 The ERD grammar

The ERD grammar is a context-free grammar written using the classical *Backus-Naur Form (BNF)* [45] that has been defined to describe how all ERD constructs can be lexical combined in a sequence.

It can be expressed as a set of non-terminal symbols NT, a set of terminal symbols T, and a set of derivation rules (Table 5.4). The set T of terminal symbols, as shown in Table 5.3, can be divided into two subsets: ET contains the terminal symbols related to the ERD constructs, while DT contains the descriptive terminal symbols in plain English that represent the labels of the ERD constructs, as the name of an entity or the cardinality of a relation, so $T = ET \bigcup DT$.

Table 5.2 reports the notation used in writing the BNF for the ERD grammar, while the actual grammar is reported in tables 5.3 and 5.4 respectively. For more clarity, the reader can see the convention used in the grammatical model of the OWL W3C report [55], which was referred to for devising the OWL grammar that has been used to write the observation sequences and for building this grammar.

Construct	Syntax	Example
terminal symbols	sequence of alphabetic chars	`Entity`
a set of terminal symbols described in English	italic	*a sequence of alphanumeric chars*
nonterminal symbols	enclosed in angle brackets	`<AttributeAxiom>`
zero or more	curly braces	`{AttributeAxiom}`
one or more	square brackets	`[<AttributeAxiom>]`
alternative	vertical bar	`Entity <id>| Entity <id><key>`

Table 5.2: *The BNF Notation [61]*

```
ET = {Entity, Attribute, SimpleAttribute, CompositeAttribute,
       MultiValueAttribute, Relation, Is-A, Bi-DirectionalRelationship,
       Domain, Range, Identifier, Key, Primary-key}

DT = {plain regular expression, string of alphabetic chars,
       string of alphanumeric chars, sequence of digits, 1,
       an integer value}

NT = {<ERDStatement>,<EntityAxiom>, <Entity>, <id>, <name>, <value>,
       <regex>, <PrimaryKey>, <key>, <EntityDefinition>, <AttributeAxiom>,
       <RelationAxiom>, <SimpleAttributeAxiom>, <CompositeAttributeAxiom>,
       <MultiValueAttributeAxiom>, <Relation>, <DomainAxiom>, <RangeAxiom>,
       <1:NRelationAxiom>, <M:NRelationAxiom>, <Is-AAxiom>,
       <Bi-DirectionalAxiom>, <id>, <regex>,<name>,<value>,<key>,
       <unary>,<nary>}

Start symbol = <Diagram>
```

Table 5.3: *The ERD grammar BNF - Terminal And Non-Terminal Symbols*

```
<id> ::= Identifier <regex>
<regex> ::= plain regular expression
<name> ::= string of alphabetic chars
<value> ::= string of alphanumeric chars
<key> ::= sequence of digits
<unary> ::= 1
<nary> ::= an integer value

<Diagram> ::= {<ERDStatement>}

<ERDStatement> ::= <EntityAxiom>|<AttributeAxiom>|<RelationAxiom>

<EntityAxiom>::= <Entity><EntityDefinition>|<Entity><id><EntityDefinition>
                |<Entity><PrimaryKey><EntityDefinition>
                |<Entity><id><PrimaryKey><EntityDefinition>

<EntityDefinition> ::= <key> AttributeAxiom|<key> <AttributeAxiom>
                     | <name> AttributeAxiom|<name><AttributeAxiom>

<PrimaryKey> ::= Primary-key <key>

<Entity> ::= Entity

<AttributeAxiom> ::= <SimpleAttributeAxiom>| <CompositeAttributeAxiom>
                   | <MultiValueAttributeAxiom>

<SimpleAttributeAxiom> ::= Attribute <name>| Attribute<name><value>

<CompositeAttributeAxiom> ::= CompositeAttribute <name> [<SimpleAttributeAxiom>]

<MultiValueAttributeAxiom> :: = MultiValueAttribute <name> [<value>]

<RelationAxiom> ::= <DomainAxiom> <Relation> <RangeAxiom>

<Relation> ::= <1:NRelationAxiom>| <M:NRelationAxiom>
             | <Is-AAxiom>| <Bi-DirectionalAxiom>

<1:NRelationAxiom> := <unary> Relation <nary>

<M:NRelationAxiom> ::= <nary> Relation <nary>

<Is-AAxiom> ::= Is-A

<Bi-DirectionalAxiom> ::= Bi-DirectionalRelationship

<DomainAxiom> ::= Domain <EntityAxiom>

<RangeAxiom> ::= Range <EntityAxiom>
```

Table 5.4: *The ERD grammar BNF - Derivation rules [61]*

5.2.2 Computing emission probabilities in transformation process

In the $OWL \rightarrow ERD$ transformation, the emission probabilities are computed starting from the heuristics set $H_S = (o_l, e_m) : o_l \in O, e_m \in S$s that is the set of all the ordered OWL-ERD couples derived from the transformation rules we have seen in section 5.1. Given the observable states space O and the hidden states space E, the single element $b_{i,j} = p(o_j|e_i)$ of the emission matrix is computed as the *likelihood function* $l : O \times E \rightarrow [0, 1]$

$$b(i,j) \equiv l(o_j, e_i) = \begin{cases} \frac{v_{o_j}}{v_{e_i}} & \text{if } v_{o_j} > 0 \\ 0 & otherwise \end{cases} \tag{5.1}$$

where v_{o_j} is the number of couples containing the OWL construct o_j while v_{e_i} is the number of couples containing both the OWL construct o_j and the ERD construct e_i. As supposed in the section 5.1 the set of correspondences has to contain any redundances. A redundance means that many authors of the works in literature consider that correspondence as valid.

5.2.3 Computing transition probabilities in transformation process

The *transition probabilities* indicate the probabilities for moving from an ERD construct to another one in making up the output ERD sequence.

The *transition probability* computation in transformation process is bounded to the definition of *structural probabilities* and of the so called *contextual lexical probabilities* that model the consecutive expansion of two terminals t_i and t_j. These probabilities are computed considering the derivation path that is the minimum path that joins the symbols in the derivation tree. The contextual lexical probability cl_{t_i,t_j} is computed as the number of different derivation paths divided by the number of all derivation paths that contain both the symbols.

$$cl_{t_i,t_j} = \begin{cases} \frac{|DDP_{ij}|}{S_{ERD}} & \text{if } S_{ERD} \neq 0 \\ 0 & otherwise \end{cases} \tag{5.2}$$

According to the Barker notation [16], there are three basic *ERD concepts* in an ERD: *Entity*, *Attribute* and *Relation*.

Based on the structural constraints,was defined a set of axioms that describe the ERD sequential composition that is a method for representing an ER Diagram as sequence of ERD concepts. Such axioms establish that in a ERD sequential composition:

	Entity	Attribute	Relation
Entity	0.2	0.4	0.4
Attribute	0.33	0.33	0.33
Relation	0.6	0.2	0.2

Table 5.5: *The structural probabilities matrix S_M [61]*

- an entity is followed either by the relation whose domain is the entity itself or by one of its attributes. It cannot be followed by another entity because it becomes an isolated entity with no relations or attributes;
- a relation is followed by the entity that is its range;
- an attribute is followed by an entity in a new entity declaration, or by a relation if it is an attribute of the domain, or by another attribute of the same entity in a new attribute declaration;
- an entity follows either the relation whose range is the entity itself or an attribute in a new entity declaration;
- a relation follows the entity that is its domain or an attribute of this entity;
- an attribute follows an entity or another attribute of the same entity.

Given the set $C = \{Entity, Attribute, Relation\}$, is defined the structural matrix S_M whose elements $s_{ij}, i, j \in C$ represent the probability that the concept j follows concept i based on the structural axioms stated above.

Defining F_i as the set of all the possible ERD concepts occurring after i using the axioms, while $NF_i = C - F_i$ is the complement of C with respect to F_i , which is never empty according to the axioms.

The probabilities are shown in Table 5.5 and are computed as

$$s_{ij} = \begin{cases} 0.2 & \text{if } j \notin F_i \\ \frac{1-0.2\cdot|NF_i|}{|F_i|} & otherwise \end{cases} \tag{5.3}$$

The value 0.2 is chosen according to heuristic considerations; it is set a priori as the lowest probability value. Considering that the total number of events is 3 the probability is equal to $1/3 = 0.33$ if they are equiprobable, then 0.2 was arbitrarily chose as the value of the less probable event.

S_M is a stochastic matrix but it cannot be used directly as transition matrix because it does not include all the possible ERD constructs involved in the mapping task.

Considering that terminal symbols in ET are specializations of the ERD concepts and so defining the function $c : ET \rightarrow C$ that maps the e_i construct to the corresponding ERD concept and the set $S_M^{h,k} = \{cl_{e_i,e_j} : c(e_i) = h, c(e_j) = k$,where $h, k \in C\}$ of all the contextual lexical probabilities between terminal symbols in ET mapping to the couple of ERD concepts $c(e_i)$, $c(e_j)$. The normalization factors $x_{i,j}^{h,k}$ for the couple of terminal symbols e_i and e_j are computed as

$$x_{i,j}^{h,k} = \frac{s_{h,k}}{\sum_{|S_M^{h,k}|} cl_{e_i,e_j}} \tag{5.4}$$

being $s_{h,k}$ the $(h,k)^{th}$ element of the S_M matrix, and the summation being intended over all the elements in $S_M^{h,k}$.

5.3 An Hidden Markov Model for the $OWL \leftrightarrow ERD$ mapping

The $OWL \leftrightarrow ERD$ mapping approach proposed in this thesis constitutes an evolution of the method presented for the $OWL \rightarrow ERD$ transformation.

The main scope of this probabilistic technique consists in finding the most probable sequence of ERD symbols, choosing them from the closed vocabulary of the ERD grammar derived from the specific input database, given a sequence of OWL symbols including the labels contained into each sigle axiom of the input OWL ontology and in proposing the obtained correspondences to the user, that is responsible of choosing the most suitable for its own purposes. As explained in Chapter 2 the main applications of mappings bootstrap are in helping users for OBDA purposes.

The theoretical background is the same of the transformation approach: an Hidden Markov Model is used but being the inputs quite different a customized grammar is used and a different computation of probabilities.

So the *observable states* are the OWL FSS constructs contained in the input ontology axioms including the names of classes, properties and so on.

The *hidden states* are in the same way the ERD symbols and the name of tables, relations and so on. An ERD is described through a set of sentences written using the generalized ERD grammar described for the transformation process. Starting from these sentences a *customized ERD grammar* for the specific input is defined. Details on this customized grammar are described into section 5.3.1.

The *emission matrix* B of the HMM for the mapping process is built considering semantics between strings and structural; details are described into the section 5.3.2.

Instead, the *transition matrix A* is built using a method based on that considering the context of each terminal defines the so called *transition tree*. Details are in section 5.3.3.

5.3.1 The customized ERD grammar (C-ERD)

Given a database it is possible to represent it as a set of ERD sentences, written using the ERD grammar previously described for the transformation technique.

So, given a specific database, and so its own ERD, all the lexical information (such as labels, relations, attribute names, and attribute values, etc...) can instantiate the regular expressions in the DT set, thus defining the set $DT_C \in DT$; the ET constructs that are actually in the given diagram form the subset ET_C.

The symbols in $T_C = DT_C \bigcup ET_C$ are the terminal symbols of a new specific grammar describing the given ERD.

The NT_C set of this grammar differs from NT, because it is inferred considering the production rules that describe the specific combinations of the T_C terminal symbols in the diagram.

In fact, given a diagram, its entities, relations, and their labels can be combined according to the specific order they appear in the diagram; the production rules are written so that the correct lexical sentences representing the given ERD contents can be produced. The result is the *C-ERD grammar (Customized-ERD grammar)* which is tailored on the specific diagram.

For example, given the diagram in Figure 5.1, the correspondent $C - ERD$ grammar is specified in the tables 5.6 and 5.7, which allow to generate the following comma-separated lexical sentences describing the toy diagram:

```
Entity Children IsA Person Primarykey person_id Attribute age
   Relation hasChild Domain Parent  Relation attends Range School
   Attribute student Attribute institute ,
Entity Lady IsA Person Primarykey person_id Primarykey female_id ,
Entity Parent IsA Lady Primarykey person_id Primarykey female_id
   Relation hasChild Range Children ,
Entity Person Primarykey person_id Attribute person_id ,
Entity School Primarykey school_id Attribute school_id Relation attends
   Domain Children Attribute student Attribute institute ,
Relation hasChild Domain Parent Range Children ,
Relation attends Domain Children Range School Attribute student
   Attribute institute.
```

```
ETc = {Entity, IsA, Primarykey, Attribute,Relation, Domain, Range}
DTc = {Children, Person, Parent, School, Lady, person_id, female_id, school_id,
       hasChild, attends, age, institute, student}
Start symbol = <Diagram>
```

Table 5.6: *The customized ERD grammar BNF - Terminal and non-terminal symbols for the toy example*

```
<Diagram> ::= <ERDStatement>
<ERDStatement> ::= <EntityAx> | <RelationAx>
<EntityAx> ::= <s0> | <s1> | <s2> | <s3> | <s4>
<RelationAx> ::= <s5> | <s6>
<s0> ::= <p0> <p1> <p2> <p3> <p4> <p5> <p6> <p7> <p8> <p9>
<s1> ::= <p10> <p1> <p2> <p11> <p8> <p9>
<s2> ::= <p12> <p13> <p2> <p11> <p4> <p14> <p8> <p9>
<s3> ::= <p15> <p2> <p8> <p9>
<s4> ::= <p16> <p17> <p6> <p18> <p8> <p9>
<s5> ::= <p4> <p5> <p14>
<s6> ::= <p6> <p18> <p7> <p8> <p9>
<p0> ::= Entity Children
<p1> ::= IsA Person
<p2> ::= Primarykey person_ id
<p3> ::= Attribute age
<p4> ::= Relation hasChild
<p5> ::= Domain Parent
<p6> ::= Relation attends
<p7> ::= Range School
<p8> ::= Attribute student
<p9> ::= Attribute institute
<p10> ::= Entity Lady
<p11> ::= Primarykey female_ id
<p12> ::= Entity Parent
<p13> ::= IsA Lady
<p14> ::= Range Children
<p15> ::= Entity Person
<p16> ::= Entity School
<p17> ::= Primarykey school_ id
<p18> ::= Domain Children
```

Table 5.7: *The C-ERD grammar BNF - derivation rules for the toy example*

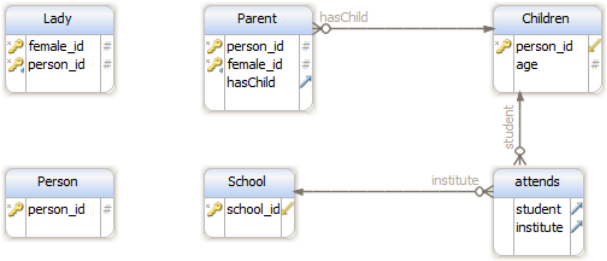


Figure 5.1: *A database schema toy example*

DT_O = {*Employee*, *Person*, *worksIn*, *Department*}
ST_O = {EquivalentClasses, ObjectIntersectionOf, ObjectSomeValuesFrom}
T_O = {EquivalentClasses, Employee, ObjectIntersectionOf, Person,
ObjectSomeValuesFrom, worksIn, Department}

Table 5.8: *Ontological symbols set: a simple example*

5.3.2 Computing emission probabilities in mapping process

Emission probabilities $b_{ij} = p(o_j|e_i)$ in the proposed mapping process are computed in two different ways depending on whether the ontological element o_j is a descriptive terminal of the OWL grammar or not.

Descriptive and not descriptive terminals sets are the equivalent of the respectively DT_C and ET_C in the customized ERD grammar. In the same way we can divide the set of symbols OT of the OWL grammar into DT_O and ST_O sets, such as $T_O = DT_O \bigcup ST_O$.

Considering as simple example the axiom:

```
EquivalentClasses(
                a:Employee , ObjectIntersectionOf(
                                a:Person ,
                                ObjectSomeValuesFrom(a:worksIn , a:Department)
))
```

the sets are those in the table 5.8 A *descriptive terminal symbol* is identifiable with a label, such as the names of classes, properties and so on. A *non-descriptive symbol* is a typical construct of the OWL FSS syntax. Logically a descriptive symbol needs to be mapped with an ERD label ($e_i \in DT_C$), as the name of entities or relation an so on, while a non descriptive terminal has to be mapped with a structural symbol of the ERD grammar, so $e_i \in ET_C$.

The probability of mapping two labels, so an $o_j \in DT_O$ with a $e_i \in DT_C$, is bounded to compunting a symilarity measure and it depends on both semantic

and syntactic evaluations. In effect, two strings can represent the same thing and have the same meaning even if they are syntactically dissimilar or be very similar (let's think about the acronyms) but with different or none meanings.

In view of these considerations, emission probabilities related to labels have been set up as a weighted sum of two measures: the *syntactic similarity sy* and the *semantic similarity sm*.

The syntactic similarity measure is a function $sy : DT_O \times DT_C \rightarrow \mathbb{R}$ that associates to each couple of labels their Jaro-Winkler[76] distance $sy(o_j, e_i) = d_{jw}(o_j, e_i)$.

Similarly to the syntactic case, given two labels o_j and e_i, the semantic similarity measure between them is a function $sm : DT_O \times DT_C \rightarrow \mathbb{R}$ that associates to each couple of labels their Wu-Palmer [77] distance: $sm(o_j, r_i) = d_{wp}(o_j, e_i)$.

Once defined the singular measures has to be defined the weighted measure. It is the function $s : DT_O \times DT_C$ defined as follow:

$$s(o_j, e_i) = \alpha \cdot sy(o_j, e_i) + \beta \cdot sm(o_j, e_i) \tag{5.5}$$

The parameters α and β were introduced for weighting differently the measures. The performances have been evaluated varying these weights. Actually the better performances have been obtained using weights that ampliphy the greater value. So the syntactic and the semantic features are avaluated and is assigned to the greater the 80 percent of the total measure. Summarly:

$$s(o_j, e_i) = \begin{cases} 0.8 \cdot sy(o_j, e_i) + 0.2 \cdot sm(o_j, e_i) & \text{if } sy(o_j, e_i) \geq sm(o_j, e_i) \\ 0.2 \cdot sy(o_j, e_i) + 0.8 \cdot sm(o_j, e_i) & otherwise \end{cases} \tag{5.6}$$

Whether a state $o_j \in ST_O$ is observed, the corresponding hidden state is an $e_i \in ET_C$. In this case the probability is computed according to the method used for the transformation process. If the set of $OWL \leftrightarrow ERD$ correspondences does not contain any rule with the OWL construct o_j the emission probability is considered the same for each element $e_i \in DT_C$; the same happens when an $o_j \in DT_O$ has all the similarity measures equal to 0. These consideration allows to avoid an interruption in the Vitebi path constructing.

Finally the emmission probabilities computation can be summarized as follow:

$$b_{i,j} \equiv \begin{cases} l(o_j, e_i) & \text{if } o_j \in ST_O \wedge e_i \in ET_C \\ s(o_j, e_i) & \text{if } o_j \in DT_O \wedge e_i \in DT_C \\ \frac{1}{|ET|} & \text{if } o_j \in ST_O \wedge e_i \in ET_C \wedge \sum_I l(o_j, e_i) = 0 \\ \frac{1}{|DT|} & \text{if } o_j \in DT_O \wedge e_i \in DT_C \wedge \sum_I s(o_j, e_i) = 0 \end{cases} \tag{5.7}$$

5.3.3 Computing transition probabilities in mapping process

Given the input database let consider its purposely defined C-ERD distinguishing the terminal set of symbols T_C (further divised into structurals ET_C, descriptive DT_C and the non-terminal symbols NT_C).

The *transition matrix* A is a $T_C \times T_C$ matrix whose elements a_{ij} represent the probability to move in the *right context* from the hidden state e_i to the hidden state e_j with $e_i, e_j \in T_C$.

The right context is implied by the typical left-to-right direction in the sequence of states building adopted in the HMM. At each time instant, a new terminal symbol has to be estimated depending on both the previously estimated one and the observation.

Keeping in account the right context of each symbol e_i has been depicted the method for computing the transition probabilities using the notion of *transition tree* t_i.

A tree t_i contains all the possible paths that link the root to the symbols findable moving in right direction from e_i in a sentence containing both symbols. The root of the tree is the symbol e_i itself; at the first level, the root symbol is linked to the ones that are placed immediately after the root itself in the occurrence set. It is not the typical well known parse tree, which represents the derivation of the sentence in the customized ERD grammar based on the production rules.

First, in development of the technique were considered all the sentences containing e_i as they can be directly derived using the customized ERD grammar, as a sequence of symbols and was considered as right context the substring of any sequence starting with e_i from we moving out and ends with the symbol e_j, computing the transition probabilities as the number of symbols separating e_i and e_j. So for example assuming we want to compute the transition probability of moving out from e_i =*attends* to e_j =*School* let select the sentences containing both e_i and e_j:

1. `Entity` ***Children*** `IsA` ***Person*** `Primarykey` ***person_id*** `Attribute` ***age*** `Relation` ***hasChild*** `Domain` ***Parent*** `Relation` ***attends*** `Range` ***School*** `Attribute` ***student*** `Attribute` ***institute***: only one symbol separates the couple so the transition from *attends* to *School* has the highter value among the ones computed to other symbols contained in DT_C.

2. `Entity` ***School*** `Primarykey` ***school_id*** `Attribute` ***school_id*** `Relation` ***attends*** `Domain` ***Children*** `Attribute` ***student*** `Attribute` ***institute***: the symbol *School* is in the left context of the symbol *attends*, so the transition is considerable null.

3. `Relation` *`attends`* `Domain` *`Children`* `Range` *`School`* `Attribute` *`student`* `Attribute` *`institute`*. As for the first case *attends* is in the right context of *School*, but the transition value is less hight.

It is clear that computing the transition probabilities in this way can imply an inaccurate detection of the ERD symbols sequence, as example in evaluating the most likely sequence corresponding to `ObjectPropertyRange(`*`a:attending a:Institute`*`)`.

So it was useful to introduce a rooted bracketed notation that allows avoiding structural redundances and addressing a different transition computing. To explain it let define a set D_{REL} containing the sentences written in the C-ERD grammar using a rooted notation. For the example used in the section 5.3.1 the set D_{REL} is the one shown in the Table 5.9.

Let now define the *occurrence set* S_i with $S_i \in D_{REL}$, which represents the set of sentences in D_{REL} containing the symbol e_i. The transition tree t_i for the symbol e_i is an ordered, rooted, and weighted tree that contains the branches from the root symbol e_i to the ones in its right context inside S_i.

Considering that a sentence is expressed as a list notation, such symbols are the ones placed at the same parenthesis insertion level. Iteratively, at each level of the tree, each node is connected to the symbols that are placed immediately after it, in the same way of the root. For estimating the transition probability values between symbols, the transition trees were weighted: the main principle is that the more the symbol is far from the root in a sentence, the less is the probability to move from the root to the symbol itself. So, the higher the weight is, the higher the probability to move from e_i to e_j even if some other symbols along the sentence are bypassed.

Formally, the transition tree of the symbol e_i is a set of ordered couples $t_i = (w_{ij}, e_j)|e_j \in S_i, j \neq i$, where w_{ij} is the weight of the branch from e_i to e_j in the tree.

As example, let consider the symbol *Children* of the toy example, numbering it with 0. The occurrence set is in Table 5.10:

It is worth to say that experiments were carried out under different considerations about the transition tree building methodolgy; so have been defined different trees. The transition tree t_0 builded under the first set of considerations is shown in Figure 5.2.

Considering the Figure 5.2 the weights are computed according to the following rules:

- if $e_i \in ET_C$ the probability to move to a label is higher than the probability to move to another symbol in ET_C. There are three steps that are iteratively applied:

D_{REL}= {
(Entity *Children*
(IsA *Person*)
(Primarykey *person_id*)
(Attribute *age*)
(Relation *hasChild* (Domain *Parent*)
attends (Range *School*) (Attribute *student institute*))),
(Entity *Lady*
(IsA *Person*)
(Primarykey *person_id female_id*)),
(Entity *Parent*
(IsA *Lady*)
(Primarykey *person_id female_id*)
(Relation *hasChild* Range *Children*)),
(Entity *Person*
(Primarykey *person_id*)
(Attribute *person_id*)),
(Entity *School*
(Primarykey *school_id*)
(Attribute *school_id*)
(Relation *attends*(Domain *Children*) (Attribute *student institute*))),
(Relation *hasChild*
(Domain *Parent*)
(Range *Children*)),
(Relation *attends*
(Domain *Children*)
(Range *School*)
(Attribute *student institute*))
}

Table 5.9: *The sentences set represented in the rooted bracketed notation*

S_i= {
```
 (Entity Children
   (IsA Person)
   (Primarykey person_id)
   (Attribute age)
   (Relation hasChild (Domain Parent)
             attends (Range School) (Attribute student institute))),
(Entity Parent
   (IsA Lady)
   (Primarykey person_id female_id)
   (Relation hasChild Range Children)),
(Entity School
   (Primarykey school_id)
   (Attribute school_id)
   (Relation attends(Domain Children) (Attribute student institute))),
(Relation hasChild
   (Domain Parent)
   (Range Children)),
(Relation attends
   (Domain Children)
   (Range School)
   (Attribute student institute))
}
```

Table 5.10: *The occurrence set for the simbol Children useful in transition tree defining*

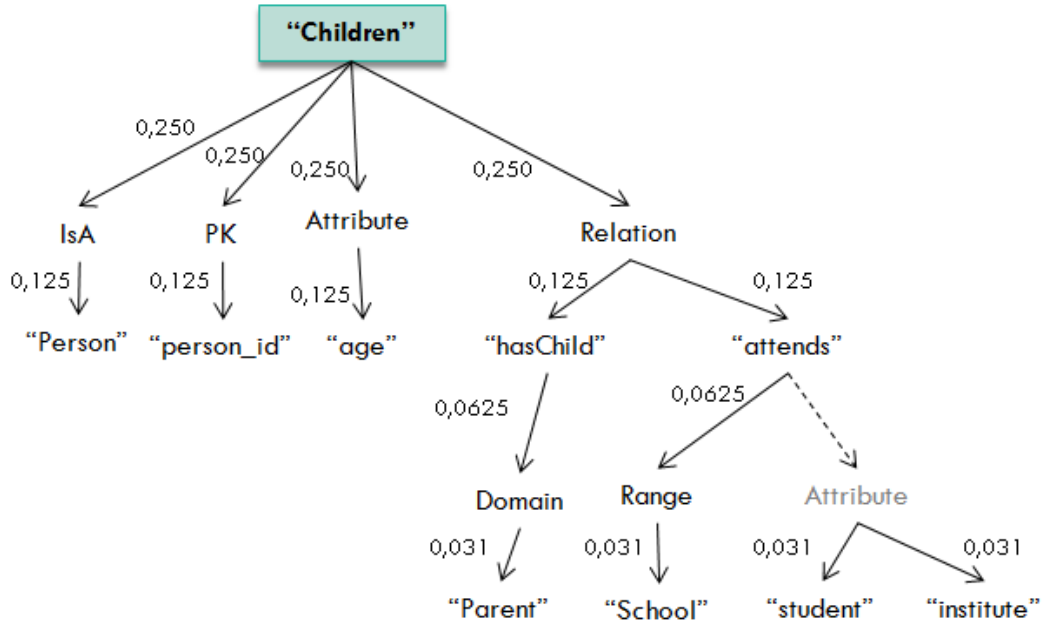


Figure 5.2: *The transition tree t_0 designed with the first approach for the symbol Children of the toy example*

1. links to the labels in DT_C from e_i have the same one-normalized weights (their sum is equal to 1);
2. links to the symbols in ET_C have the same one half of the symbols in DT_C ;
3. the other links to depth symbols in the tree have iteratively half-normalized weights with respect to their parent (i.e. their sum is equal to the half of the link to the parent).

- if $e_i \in DT_C$: the edges from the root to all the other symbols have equally one-normalized weights. In fact, according to the grammar, a label could be followed by another label in ET_C or by another symbols in DT_C indifferently. When traversing the tree in depth, the weights of other branches are normalized to the half of the weight along the edge to their parent.

When a node is replicated in the tree, the highest weight is preferred, and it is set for all involved branches to the replicated node. This is the case of the node Attribute in the example; such a node is related to both the attribute *age* of the root and the attributes *student* and *institute* of the relation *attends*.

However, the real weights are used in the next computations, and not the highest one. In the toy example, the symbol *Children* $\in DT_C$. As result the *Children* $\rightarrow$ `IsA`, *Children* $\rightarrow$ `Attribute`, *Children* $\rightarrow$ `Relation` and *Children* $\rightarrow$

`PrimaryKey` edges are weighted with $w_{0,1} = w_{0,2} = w_{0,3} = w_{0,4} = 1/4 = 0.25$ because they are at first level of the transition tree, and their weights are one-normalized. In each next level, nodes are weighted with the half of the weight of the link to their parent; for example `IsA` $\rightarrow$ *Person* and `PrimaryKey` $\rightarrow$ *person_id* are weighted with $w_{0,5} = w_{0,6} = 0.25/2 = 0.125$, and so on.

The branch *attends*$\rightarrow$`Attribute` it is not weighted because the node `Attribute` is yet considered in an highter level, while `Attribute` $\rightarrow$ *institute* and `Attribute` $\rightarrow$ *student* edges are weighted with $w_{0,12} = w_{0,13} = 0.0625/2 = 0.03125$ because the real weight is used in this computation.

Once all branches in the transition trees are weighted according to the previous rules, all of them are normalized by a X_i normalization factor, so that their sum is equal to one and can be used as probability. In general:

$$X_i = \frac{1}{\sum_j w_{ij}} \tag{5.8}$$

X_i is associated to each transition tree t_i . Formally, we defined the transition function $f : T \times T \in [0, 1]$ that associates each couple of symbols (e_i, e_j) to the probability that the hidden states sequence evolves from e_i to e_j according to the transition tree weights. In particular, given t_i and the correspondent X_i factor, the transition matrix A is built so that

$$a_{ij} = p(e_i|e_j) = f(e_i, e_j) = \begin{cases} w_{ij} \cdot X_i & \text{if } w_{ij} \in t_i \\ 0 & otherwise \end{cases} \tag{5.9}$$

An example is in Figure 5.3 where a little different transition tree is considered. The tree in this case is designed under the same considerations even if the root symbol belongs to ET_C or DT_C. In this case the transition weights assigned to any single symbol are simply obtained halving the weight assigned to the upper level. Also the symbols in the left context and non-contextual symbols (the symbols that are not included into the sentences containing the root).

Anymore, the best performances have been obtained considering a *packed transition tree*. The packed transition tree pt_0 was derived from the S_i set, considering anyway the rooted notation. The difference with the previuos method is that there is not distinction of structural and desscriptive terms. This is stated on the fact that the emission matrix yet gives the information about the kind of terminal; it allows to reduce the depth of the graph.

The weights used have been derived using an exponential value. Let consider $k = 0, 1, ..., K$ where K is the depth of pt_0 and a starting weight p (in the experiments described in Chapter 7 was used a $p = 10$). Distinguishing the set T_rc of terminals in the right context, a set T_lc of terminals in the left context and T_nc

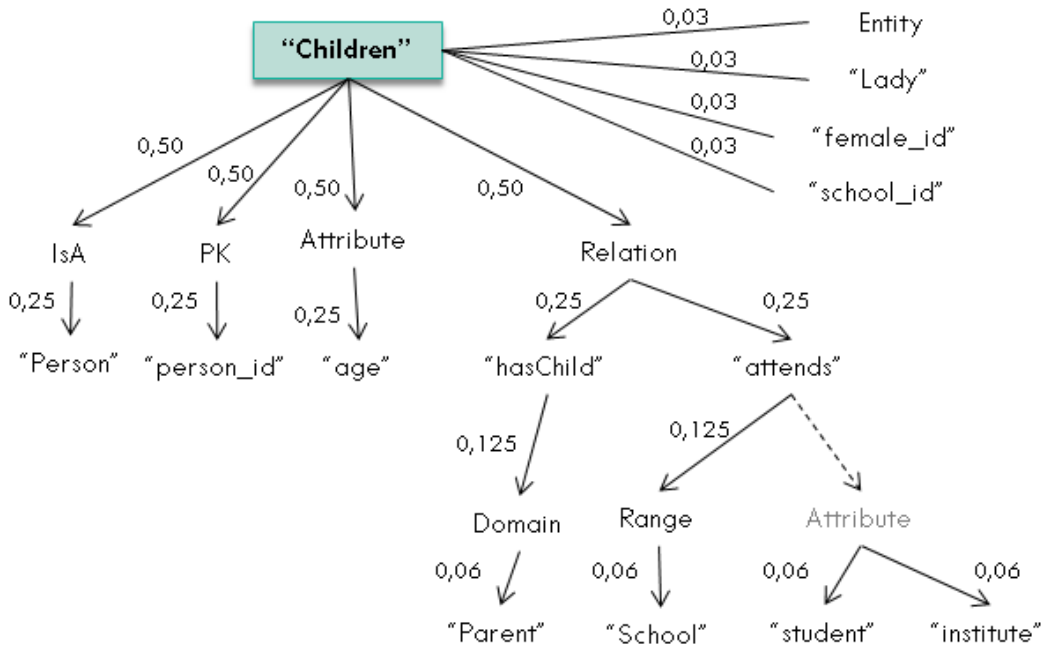


Figure 5.3: *The transition tree t_0 designed with the second approach for the symbol Children of the toy example*

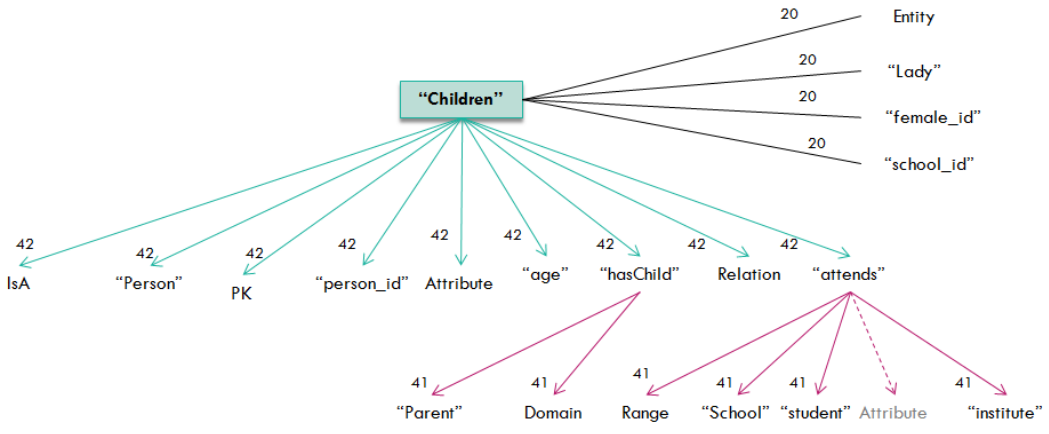


Figure 5.4: *The packed transition tree for the symbol Children of the toy example*

the set of terminals that are not in the context of the symbol e_i, the weights are computed as:

$$w_{ij} = \begin{cases} w_{nc} = p & \text{if } e_j \in T_n c \\ w_{lc} = 2p & \text{if } e_j \in T_l c \\ w_{rc} = 4p + 2^k \ \forall k \in 0, 1..., K-1 & \text{if } e_j \in T_r c \end{cases} \tag{5.10}$$

This weights allow to prevent that the transition weight is too reduced, there is a sharp difference between weights referring to the different sets, but at the same time the transition weights vary slowly in the right context. This corresponds to give greater relevance to emissions in the Viterbi path computation in moving to a left contextual or a non-contextual symbols instead of a right contextual symbol. Increasing the value of p proportionally increases the gap of weights, so the emission probability has to be very hight then others in order the Viterbi algorithm choose a left contextual or a non-contextual terminal.

As for the weights in Figure 5.2, also the weights in Figure 5.3 and Figure 5.4 need a suitable normalization in the end to ensure the stochasticity of the resulting transition matrix.

Chapter 6

System architecture

The system proposed has been implemented according to the architecture depited in Figure 6.1.

Any part of the architecture is constructed according to the methodology described in Chapter 5; in effect the architecture is thinked to provide a whole system for granting any alignment task. The inputs of the system are so choosed by user, according to the kind of aligment he want to perform for his own puropsed. The user, using a control GUI can so load his SQL database or his OWL file.

The *ERD Generator module* produces the C-ERD representation of the diagram from its SQL description. From the SQL database description we obtained the logical schema, that is the meta description of the database structure, which contains information about tables, fields, and relationships. RDBRE methodologies (Relational DataBase Reverse Engineering) for building the ERD from the logical schema.

In the transformation from a logical schema to the diagram, the main problem arises in the relationship identification. To cope with this problem, we applied RDBRE techniques to the well-known conceptual-logical schema transformation. The following rules hold:

1. If a table has not any foreign key in the logical schema, it is an entity in the ER diagram. The fields in this table become attributes of the entity, and the primary keys of the table become the primary keys of the entity;

2. If a table has one (or more) foreign key that is not the primary key in the logical schema, it is an entity in the ER diagram. Unlike the previous case, the foreign key field is not translated into an attribute of the entity, but it becomes a relationship between this entity and the entity that contains the primary key referenced by the foreign key. Such a relationship has a one-to-many cardinality where one is on the side of the primary key, and many is

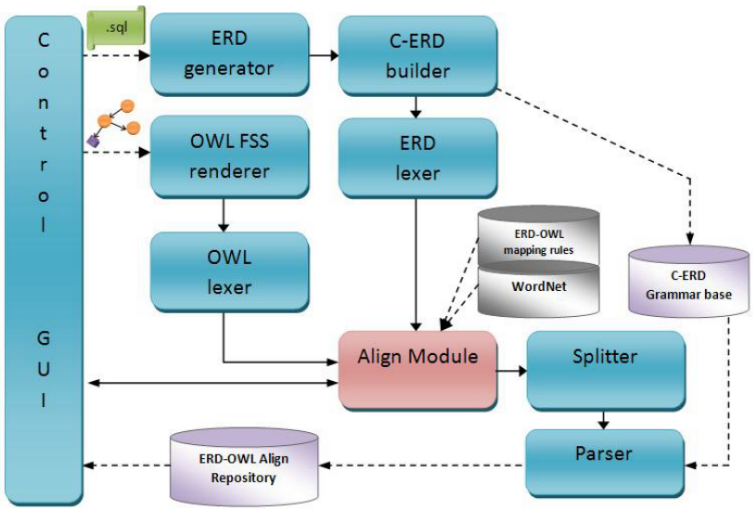


Figure 6.1: *Overall proposed alignment system architecture*

on the side of the foreign key.

3. The table in which all primary keys are foreign keys, and the foreign keys refer to different tables, is a relationship between the entities referenced by foreign keys. Such a relationship as a many-to-many cardinality. Finally, any additional fields in the table that are not keys, become attributes of the relationship.

4. The table in which a foreign key refers to a primary key of the same table, is an entity in the ER diagram. The foreign key becomes a recursive relationship that links an entity to itself, and has a one-to-many cardinality.

The ERD Generator builds the model producing a variety of entities, relationships, and their attributes. In the implementation of the module an iterative incremental approach has been adopted; the module starts assuming that all tables are entities. Then the tables owning foreign keys are considered, and the module evaluates one of the two following options:

1. If all primary keys are also foreign keys, the system transforms the entity in a many-to-many relationship;

2. If not all keys are primary keys, the system transforms the entity in a one-to-many relationship.

Once the module has identified the cardinality of the relationship, it investigates if it is a unary or binary relationship. The choice depends on the foreign key

reference. If the foreign key points to a primary key in this same table, a unary relationship is selected otherwise the relationship is a binary one.

The results are arranged as three vectors V_E, V_A and V_R containing the entities, the attributes and the relationships respectively. Additional information is stored for each element in the vectors elements, such as the attributes of each entity, and the relationships cardinality. Such vectors are next used by the C-ERD grammar builder for the ERD grammatical description.

If the user wants to transform an ERD to OWL, he loads a SQL database and the output will be the OWL obtained by applying the correspondences rules already stated.

If the user wants to transform an OWL ontology to ERD, he loads the corresponding OWL file, and the output will be an ERD diagram estimated by the HMM defined in [61].

If he wants to map an OWL ontology to a particular ERD, he loads both representations, and the output in this case will be a set of mapping couples from the two representations. In all cases, the output is stored in the ERD-OWL Align Repository.

The OWL FSS Renderer is used to obtain the axiomatic representation of the OWL/RDF ontology singletons; in this way, the OWL FSS representation is written out, and its elements (i.e. labels and constructs) can be used as the observable states of the HMM.

The *ERD lexer* and the *OWL lexer* scan the ERD and OWL sentences respectively to split them into lexical tokens. The core of the architecture is the *Align module*; it computes either the transformation of the input representation or the ERD-OWL mapping. The *Align module* interfaces with the *ERD-OWL mapping rules* base that contains all the correspondence rules between ERD and OWL language constructs. Moreover, the module is connected to the *WordNet* linguistic source [53] that is used for computing the similarity values between the labels when the HMM is performing its estimations. The sentences estimated by the *Align module* are next split by the *Splitter* to consider their basic components, which are useful for integrating the mapping process with the transformation one, as it will be more clear in next subsections. The *Parser* infers if an estimated sentence or sentence fragment obeys to the specific grammar (C-ERD grammar for the ERD, W3C BNF in [55] for OWL) that is the sentence or the sentence fragment represents coherent structures in the involved repressentations.

The *Control GUI* manages the whole alignment process and the interaction with the user.

In the following subsections are shown two sub-architectures used for the tools development referring to the two probabilistic approaches; in section 6.1. The first tool called OMEGA (Ontology → Markov → ERD Generator Application) is buided on the architecture described in [61] and references to the $OWL \rightarrow ERD$

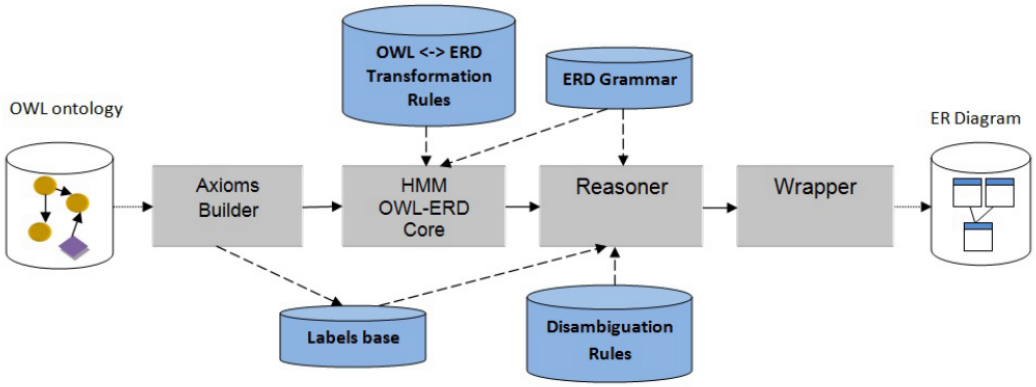


Figure 6.2: *The OMEGA architecture for the $OWL \rightarrow ERD$ transformation [61]*

.

transformation. The second one called HOWERD (HMM OWL-ERD) allows user to interact with the mapping process. HOWERD was developed in three ways, as a development/testing desktop tool that was used for the experiments and allows to follow al phases of the process,a user desktop tool and a web tool that allow to input the sources and managing the output couples of axiom - sentence correspondences.

6.1 The transformation model architecure and OMEGA development

The architectute shown in Figure 6.2 refers to the $OWL \rightarrow ERD$ transformation. This architecture is described by Pipitone and Pirrone [61].

The system is composed by four main steps: the *Axioms Builder*, the *HMM $OWL \rightarrow ERD$ Core*, the *Reasoner* and the *Wrapper*.

The Axioms Builder receives an OWL ontology as its input, and rewrites it as the set of axioms according to W3C specification. In this module the declaration of names, labels and so on are removed from each axiom and getted in the *Labels base* maintaining the information about the position in the axiom. The result is a sequence of OWL constructs and it is fed into the Core module.

The *HMM $OWL \rightarrow ERD$ Core module* is responsible of defining the HMM using the constructs obtained by the previous step as observable states and computing the most likely sequence of corresponding ERD constructs. The Core uses so the Viterbi algorithm and generates the sequence of hidden ERD states corresponding to the sequence of OWL constructs for each axiom, and puts them as input to the Reasoner that builds the correct derivation tree.

The Reasoner put into the resulting sequence of ERD constructs the labels extracting them back from the label space. The Reasones uses a disambiguation Rules to produce the correct sentence of the ERD, making also some adjustments about the position of constructs and labels. Let take as example the `ObjectPropertyDomain` or `ObjectPropertyRange`

```
ObjectPropertyDomain(a:hasChild a:Mother)
```

where the first label refers to the `ObjectProperty` *hasChild*, the `ObjectPropertyDomain` is typically transformed into an ERD construct `Domain`, so the Reasoner, after getting in the labels as

`Domain` ***`hasChild Mother`***

uses the disambiguation rules to build the correct sentence

`Relation` ***`hasChild`*** `Domain` ***`Mother`***

according to the ERD grammar. It is worth to note that the sentences in this kind of transformation are builded as a sequence of constructs exactlly alternating a construct with a label.

The *Wrapper* has the tasks to merge the constructs labeled by the same DT terminal symbols that correspond to the same ERD construct. Moreover it associates the attributes to the proper entities, and defines relations.

OMEGA (Ontology → Markov → ERD Generator Application) is the developed Java tool that allow user to perform the transformation described above. It uses the GUI in the Figure 6.3 (The ontology used in the Figure is the UnivCS of the first dataset used in the experimental setup in the Chapter 7.

In the main window the user can load his own ontology. Starting processing the implemented system extracts the informations about the input ontology elements using OWL APIs. The advantage of using OWL APIs istead of Jena APIs consists in treat the ontology as a set of axiom, instead of a set of triples that are typical of the RDF language. The ontology axioms aconstructs (only structural) The emission matrix and the transition matrix are computed. Since the transition probabilities are computed according to the general ERD grammar, tre transition matrix is always the same (Figure 6.4) for all iterations.

The box *Correspondences* contains all the OWL constructs; each construct is associated to a number that will be used as observation, so each axiom is converted into a sequence of discrete observations.

The button *Most likely sequence* call the method for performing the Viterbi algorithm, so for each observation sequence is computed the most likely sequence of ERD constructs.

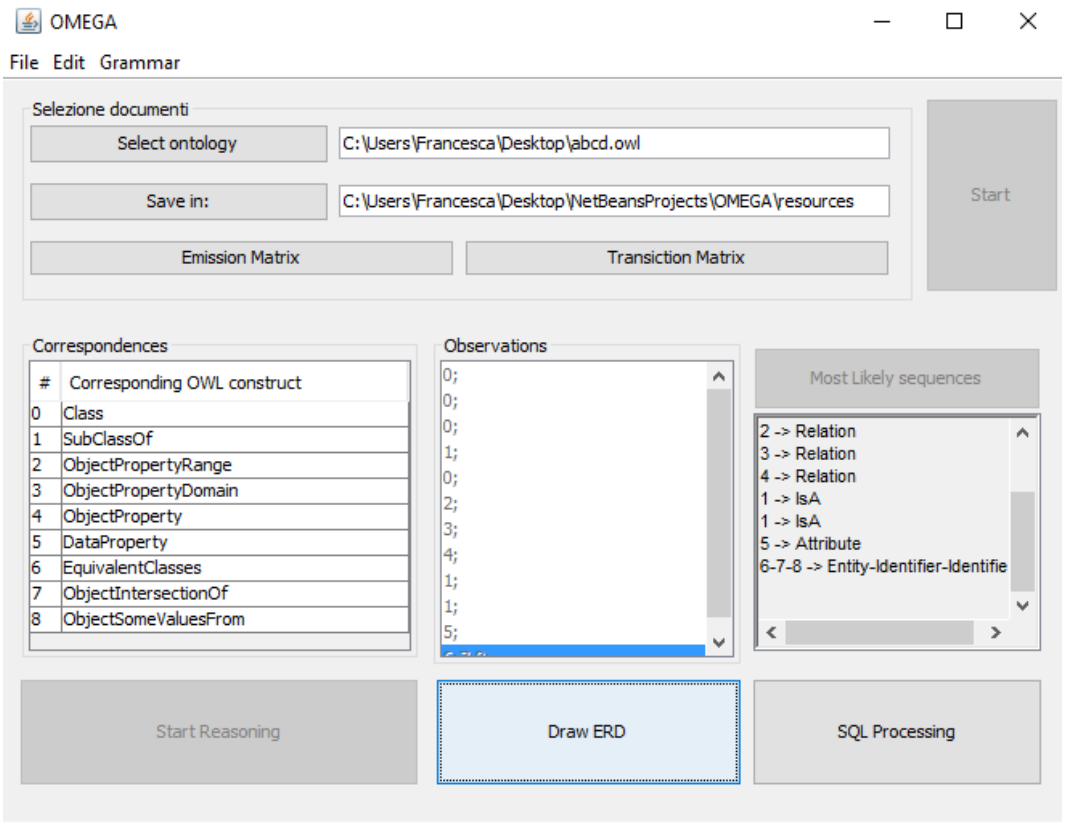
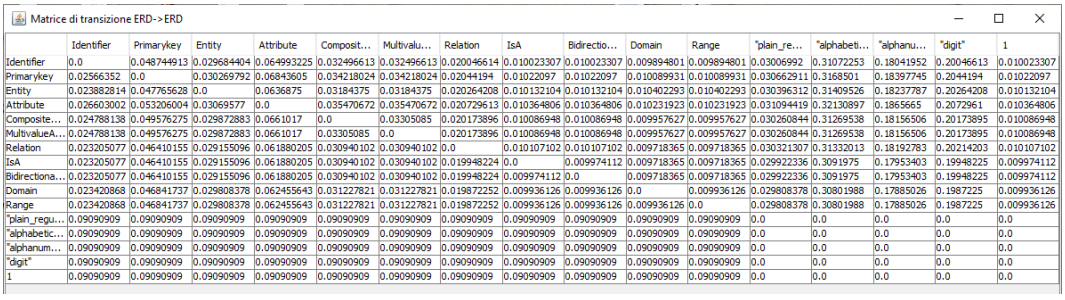
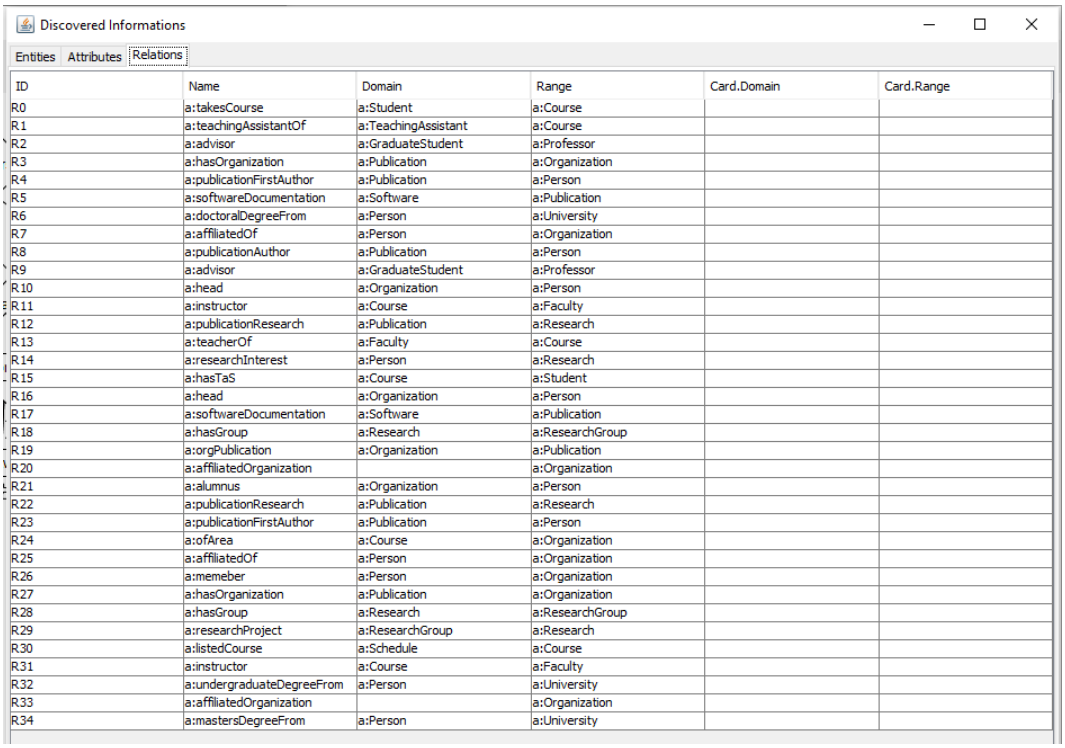


Figure 6.3: *The OMEGA GUI: the main window*



	Identifier	Primarykey	Entity	Attribute	Composit...	Multivalu...	Relation	IsA	Bidirectio...	Domain	Range	"plain_re...	"alphabeti...	"alphanu...	"digit"	1
Identifier	0.0	0.048744913	0.029684404	0.064993225	0.032496613	0.032496613	0.020046614	0.010023307	0.010023307	0.009894801	0.009894801	0.03006992	0.31072253	0.18041952	0.20046613	0.010023307
Primarykey	0.02566352	0.0	0.030269792	0.06843605	0.034218024	0.034218024	0.02044194	0.01022097	0.01022097	0.010089931	0.010089931	0.030662911	0.3168501	0.18397745	0.2044194	0.01022097
Entity	0.023882814	0.047765628	0.0	0.0636875	0.03184375	0.03184375	0.020264208	0.010132104	0.010132104	0.010402293	0.010402293	0.030396312	0.31409526	0.18237787	0.20264208	0.010132104
Attribute	0.026603002	0.053206004	0.03069577	0.0	0.035470672	0.035470672	0.020729613	0.010364806	0.010364806	0.010231923	0.010231923	0.031094419	0.32130897	0.1865665	0.2072961	0.010364806
Composite...	0.024788138	0.049576275	0.029872883	0.0661017	0.0	0.03305085	0.020173896	0.010086948	0.010086948	0.009957627	0.009957627	0.030260844	0.31269538	0.18156506	0.20173895	0.010086948
MultivalueA...	0.024788138	0.049576275	0.029872883	0.0661017	0.03305085	0.0	0.020173896	0.010086948	0.010086948	0.009957627	0.009957627	0.030260844	0.31269538	0.18156506	0.20173895	0.010086948
Relation	0.023205077	0.046410155	0.029155096	0.061880205	0.030940102	0.030940102	0.0	0.010107102	0.010107102	0.009718365	0.009718365	0.030321307	0.31332013	0.18192783	0.20214203	0.010107102
IsA	0.023205077	0.046410155	0.029155096	0.061880205	0.030940102	0.030940102	0.019948224	0.0	0.009974112	0.009718365	0.009718365	0.029922336	0.3091975	0.17953403	0.19948225	0.009974112
Bidirectiona...	0.023205077	0.046410155	0.029155096	0.061880205	0.030940102	0.030940102	0.019948224	0.009974112	0.0	0.009718365	0.009718365	0.029922336	0.3091975	0.17953403	0.19948225	0.009974112
Domain	0.023420868	0.046841737	0.029808378	0.062455643	0.031227821	0.031227821	0.019872252	0.009936126	0.009936126	0.0	0.009936126	0.029808378	0.30801988	0.17885026	0.1987225	0.009936126
Range	0.023420868	0.046841737	0.029808378	0.062455643	0.031227821	0.031227821	0.019872252	0.009936126	0.009936126	0.009936126	0.0	0.029808378	0.30801988	0.17885026	0.1987225	0.009936126
"plain_regu...	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.0	0.0	0.0	0.0	0.0
"alphabetic...	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.0	0.0	0.0	0.0	0.0
"alphanum...	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.0	0.0	0.0	0.0	0.0
"digit"	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.0	0.0	0.0	0.0	0.0
1	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.09090909	0.0	0.0	0.0	0.0	0.0

Figure 6.4: *The transition matrix for $OWL \rightarrow ERD$ transfomation*



ID	Name	Domain	Range	Card.Domain	Card.Range
R0	a:takesCourse	a:Student	a:Course		
R1	a:teachingAssistantOf	a:TeachingAssistant	a:Course		
R2	a:advisor	a:GraduateStudent	a:Professor		
R3	a:hasOrganization	a:Publication	a:Organization		
R4	a:publicationFirstAuthor	a:Publication	a:Person		
R5	a:softwareDocumentation	a:Software	a:Publication		
R6	a:doctoralDegreeFrom	a:Person	a:University		
R7	a:affiliatedOf	a:Person	a:Organization		
R8	a:publicationAuthor	a:Publication	a:Person		
R9	a:advisor	a:GraduateStudent	a:Professor		
R10	a:head	a:Organization	a:Person		
R11	a:instructor	a:Course	a:Faculty		
R12	a:publicationResearch	a:Publication	a:Research		
R13	a:teacherOf	a:Faculty	a:Course		
R14	a:researchInterest	a:Person	a:Research		
R15	a:hasTaS	a:Course	a:Student		
R16	a:head	a:Organization	a:Person		
R17	a:softwareDocumentation	a:Software	a:Publication		
R18	a:hasGroup	a:Research	a:ResearchGroup		
R19	a:orgPublication	a:Organization	a:Publication		
R20	a:affiliatedOrganization		a:Organization		
R21	a:alumnus	a:Organization	a:Person		
R22	a:publicationResearch	a:Publication	a:Research		
R23	a:publicationFirstAuthor	a:Publication	a:Person		
R24	a:ofArea	a:Course	a:Organization		
R25	a:affiliatedOf	a:Person	a:Organization		
R26	a:memeber	a:Person	a:Organization		
R27	a:hasOrganization	a:Publication	a:Organization		
R28	a:hasGroup	a:Research	a:ResearchGroup		
R29	a:researchProject	a:ResearchGroup	a:Research		
R30	a:listedCourse	a:Schedule	a:Course		
R31	a:instructor	a:Course	a:Faculty		
R32	a:undergraduateDegreeFrom	a:Person	a:University		
R33	a:affiliatedOrganization		a:Organization		
R34	a:mastersDegreeFrom	a:Person	a:University		

Figure 6.5: *The OMEGA GUI: the ERD constructs after the reasoning (only the Relation tab is shown)*

Starting Reasoning is shown in the window in figure 6.5; according to the Barker Notation, the mai three kind of ERD elements are so labeled and other information are derived from the ERD construct found. At this time, using these elements it is possible to represent graphically the ER Diagram (Figure 6.6).

Finally can be producted a sql file and a database schema in a common DBMS (PostgreSQL has been used) is created.

6.2 The mapping model architecture and HOWERD development

The Figure 6.7 shows the architecture of the mapping system.

In a first phase the inputs are described as a set of sentences according to the general ERD grammar and a set of OWL axioms, extracted using the OWL APIs.

In the next step, the customized ERD grammar is builded starting from the ERD sentences. The ERD terminals of the customized ERD grammar are so used as hidden states into the *HMM OWL-ERD Core*.

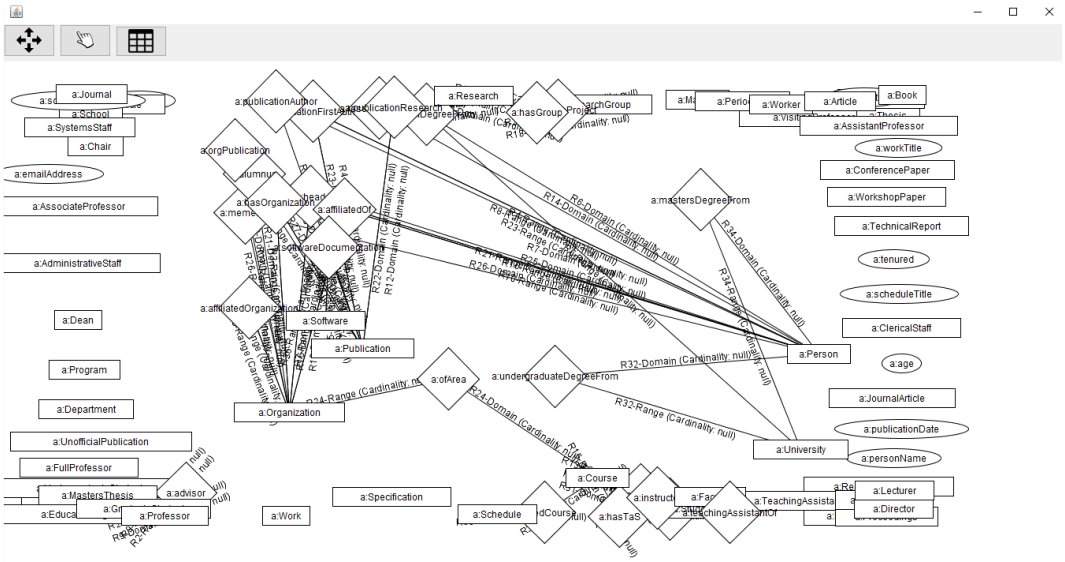


Figure 6.6: *The OMEGA GUI: the ERD diagram*

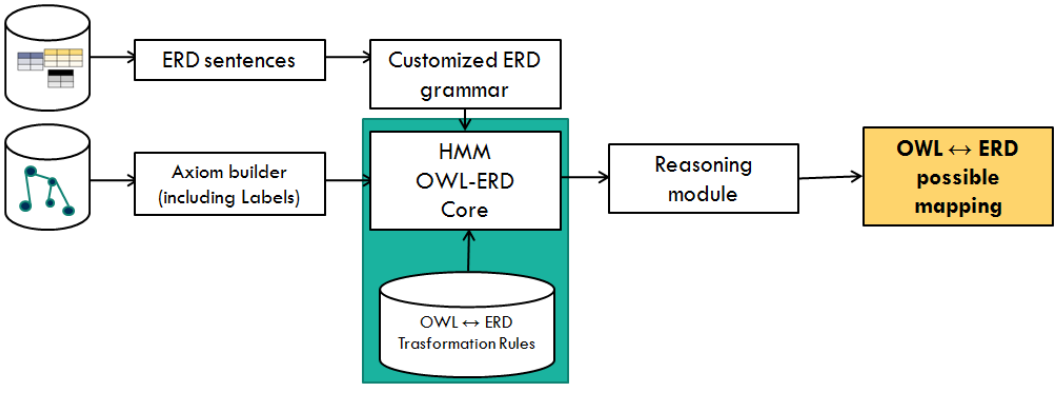


Figure 6.7: *The HOWERD architecture for the* $OWL \leftrightarrow ERD$ *mapping*

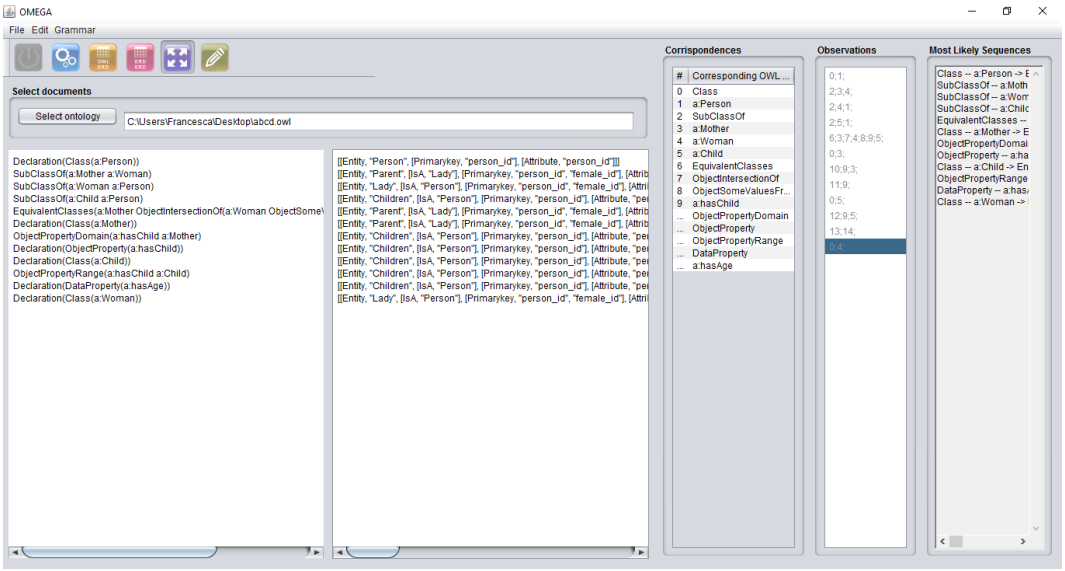


Figure 6.8: *The HOWERD GUI: the main window*

The *HMM OWL-ERD Core* is the main module; in this phase is defined the HMM as described into the Chapter 5 and the most likely sequence for each axiom is computed.

The *Reasoner* has as input the most likely sequence computed by Viterbi algorithm for any single axiom and is due to choose from the set of the input ERD sentences the sentence containing all the symbols into the sequence.

As for the implemented transformation tool, a stand-alone tool, called HOWERD (HMM OWL-ERD), has been developed for performing the $OWL \leftrightarrow ERD$ mapping. The tool has three versions:

- A development/testing desktop tool used in experimental running. The main panel of the GUI is shown in the Figure 6.9;
- A user-level desktop tool that performs the alignment without showing user the intermediate step as in the development tool. The GUI is showed in Figure 6.12;
- A web tool stuctured as the user desktop tool but developed as web service. The GUI is showed in Figure 6.15.

In order to select the inputs, it is possible to choose the ontology directly on the main window, while the ERD has to be used the panel in Figure 6.8. Typing the information about the database schema the user can automatically generate the ERD sentences and then he can view the ERD grammar in BNF. On the top of the main panel (6.9) there are some buttons that allow to show or start

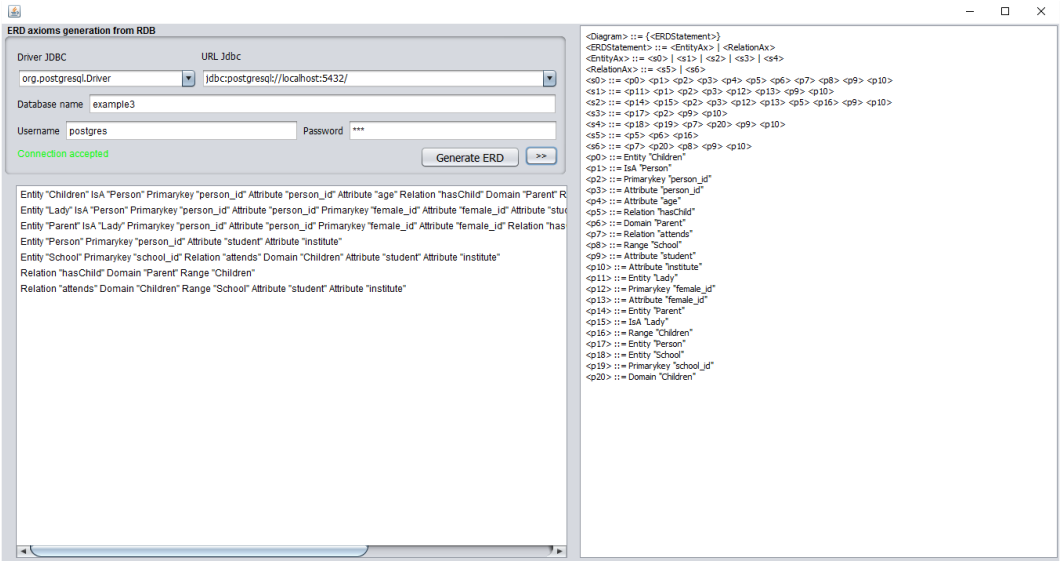


Figure 6.9: *HOWERD tool: the window for ERD management*

Matrice di emissione OWL->ERD

	Class	a:Person	SubClass...	a:Mother	a:Woman	a:Child	Equivalen...	ObjectInte...	ObjectSo...	a:hasChild	ObjectPro...	ObjectPro...	ObjectPro...	DataProp...	a:hasAge
Entity	0.596658...	0.0	0.497512...	0.0	0.0	0.0	0.5	0.142857...	0.142857...	0.0	9.950248...	0.001491...	9.950248...	9.940357...	0.0
IsA	0.001392...	0.0	0.497512...	0.0	0.0	0.0	0.5	0.142857...	0.142857...	0.0	9.950248...	0.248508...	9.950248...	9.940357...	0.0
Primarykey	0.198886...	0.0	9.950248...	0.0	0.0	0.0	0.0	0.142857...	0.142857...	0.0	9.950248...	0.248508...	9.950248...	9.940357...	0.0
Attribute	0.198886...	0.0	9.950248...	0.0	0.0	0.0	0.0	0.142857...	0.142857...	0.0	9.950248...	0.001491...	9.950248...	0.994035...	0.0
Relation	0.001392...	0.0	9.950248...	0.0	0.0	0.0	0.0	0.142857...	0.142857...	0.0	0.497512...	0.497017...	0.497512...	9.940357...	0.0
Domain	0.001392...	0.0	9.950248...	0.0	0.0	0.0	0.0	0.142857...	0.142857...	0.0	0.497512...	0.001491...	9.950248...	9.940357...	0.0
Range	0.001392...	0.0	9.950248...	0.0	0.0	0.0	0.0	0.142857...	0.142857...	0.0	9.950248...	0.001491...	0.497512...	9.940357...	0.0
"Children"	0.0	0.089744...	0.0	0.074113...	0.084740...	0.129777...	0.0	0.0	0.0	0.110289...	0.0	0.0	0.0	0.0	0.066667...
"Person"	0.0	0.124759...	0.0	0.065983...	0.094207...	0.090116...	0.0	0.0	0.0	0.076584...	0.0	0.0	0.0	0.0	0.069767...
"person_id"	0.0	0.113418...	0.0	0.073314...	0.094207...	0.090116...	0.0	0.0	0.0	0.068074...	0.0	0.0	0.0	0.0	0.072093...
"age"	0.0	0.049903...	0.0	0.059984...	0.024233...	0.0	0.0	0.0	0.0	0.029782...	0.0	0.0	0.0	0.0	0.139535...
"hasChild"	0.0	0.077772...	0.0	0.072509...	0.072699...	0.131419...	0.0	0.0	0.0	0.234538...	0.0	0.0	0.0	0.0	0.069767...
"Parent"	0.0	0.091364...	0.0	0.114860...	0.072920...	0.052567...	0.0	0.0	0.0	0.051546...	0.0	0.0	0.0	0.0	0.074419...
"attends"	0.0	0.048967...	0.0	0.088152...	0.048812...	0.047060...	0.0	0.0	0.0	0.043964...	0.0	0.0	0.0	0.0	0.072647...
"School"	0.0	0.064522...	0.0	0.073533...	0.080575...	0.088614...	0.0	0.0	0.0	0.075307...	0.0	0.0	0.0	0.0	0.089302...
"student"	0.0	0.080472...	0.0	0.073096...	0.077633...	0.081855...	0.0	0.0	0.0	0.069563...	0.0	0.0	0.0	0.0	0.068217...
"institute"	0.0	0.046207...	0.0	0.051069...	0.057572...	0.045948...	0.0	0.0	0.0	0.046210...	0.0	0.0	0.0	0.0	0.057733...
"Lady"	0.0	0.060489...	0.0	0.084594...	0.099136...	0.063081...	0.0	0.0	0.0	0.058324...	0.0	0.0	0.0	0.0	0.072868...
"female_id"	0.0	0.087854...	0.0	0.095253...	0.112684...	0.090825...	0.0	0.0	0.0	0.068376...	0.0	0.0	0.0	0.0	0.065116...
"school_id"	0.0	0.064522...	0.0	0.073533...	0.080575...	0.088614...	0.0	0.0	0.0	0.067436...	0.0	0.0	0.0	0.0	0.081861...

Figure 6.10: *The Emission Matrix of the toy example showed in HOWERD*

Matrice di transizione ERD->ERD

	Entity	IsA	Primarykey	Attribute	Relation	Domain	Range	"Children"	"Person"	"person_id"	"age"	"hasChild"	"Parent"	"attends"	"School"	"student"	"institute"
Entity	0.0	0.051886...	0.051886...	0.051886...	0.051886...	0.049528...	0.049528...	0.056603...	0.056603...	0.051886...	0.051886...	0.051886...	0.056603...	0.051886...	0.056603...	0.049528...	0.049528...
IsA	0.058171...	0.0	0.058171...	0.058171...	0.058171...	0.058171...	0.058171...	0.058171...	0.027700...	0.058171...	0.058171...	0.058171...	0.058171...	0.058171...	0.058171...	0.058171...	0.058171...
Primarykey	0.044843...	0.044843...	0.0	0.044843...	0.044843...	0.044843...	0.044843...	0.044843...	0.044843...	0.094170...	0.044843...	0.044843...	0.044843...	0.044843...	0.044843...	0.044843...	0.044843...
Attribute	0.0390625	0.0390625	0.0390625	0.0	0.0390625	0.0390625	0.0390625	0.0390625	0.0390625	0.082031...	0.082031...	0.0390625	0.0390625	0.0390625	0.0390625	0.082031...	0.082031...
Relation	0.033112...	0.033112...	0.033112...	0.069536...	0.0	0.069536...	0.069536...	0.069536...	0.033112...	0.033112...	0.033112...	0.072847...	0.069536...	0.072847...	0.069536...	0.069536...	0.069536...
Domain	0.060869...	0.060869...	0.060869...	0.060869...	0.060869...	0.0	0.060869...	0.028985...	0.060869...	0.060869...	0.060869...	0.060869...	0.028985...	0.060869...	0.060869...	0.060869...	0.060869...
Range	0.058171...	0.058171...	0.058171...	0.058171...	0.058171...	0.058171...	0.0	0.027700...	0.058171...	0.058171...	0.058171...	0.058171...	0.058171...	0.058171...	0.027700...	0.058171...	0.058171...
"Children"	0.027472...	0.060439...	0.060439...	0.060439...	0.060439...	0.057692...	0.057692...	0.0	0.060439...	0.060439...	0.060439...	0.060439...	0.057692...	0.060439...	0.057692...	0.057692...	0.057692...
"Person"	0.045871...	0.045871...	0.096330...	0.096330...	0.045871...	0.045871...	0.045871...	0.045871...	0.0	0.096330...	0.045871...	0.045871...	0.045871...	0.045871...	0.045871...	0.045871...	0.045871...
"person_id"	0.054054...	0.054054...	0.054054...	0.054054...	0.054054...	0.054054...	0.054054...	0.054054...	0.054054...	0.0	0.054054...	0.054054...	0.054054...	0.054054...	0.054054...	0.054054...	0.054054...
"age"	0.057142...	0.057142...	0.057142...	0.057142...	0.057142...	0.057142...	0.057142...	0.057142...	0.057142...	0.057142...	0.0	0.057142...	0.057142...	0.057142...	0.057142...	0.057142...	0.057142...
"hasChild"	0.043668...	0.043668...	0.043668...	0.043668...	0.043668...	0.091703...	0.091703...	0.091703...	0.043668...	0.043668...	0.043668...	0.0	0.091703...	0.043668...	0.043668...	0.043668...	0.043668...
"Parent"	0.033003...	0.072607...	0.072607...	0.072607...	0.072607...	0.033003...	0.069306...	0.069306...	0.033003...	0.072607...	0.033003...	0.072607...	0.0	0.033003...	0.033003...	0.033003...	0.033003...
"attends"	0.038910...	0.038910...	0.038910...	0.081712...	0.038910...	0.081712...	0.081712...	0.081712...	0.038910...	0.038910...	0.038910...	0.038910...	0.038910...	0.0	0.081712...	0.081712...	0.081712...
"School"	0.035211...	0.035211...	0.077464...	0.077464...	0.077464...	0.073943...	0.035211...	0.073943...	0.035211...	0.035211...	0.035211...	0.035211...	0.035211...	0.077464...	0.0	0.073943...	0.073943...
"student"	0.055555...	0.055555...	0.055555...	0.055555...	0.055555...	0.055555...	0.055555...	0.055555...	0.055555...	0.055555...	0.055555...	0.055555...	0.055555...	0.055555...	0.055555...	0.0	0.055555...
"institute"	0.055555...	0.055555...	0.055555...	0.055555...	0.055555...	0.055555...	0.055555...	0.055555...	0.055555...	0.055555...	0.055555...	0.055555...	0.055555...	0.055555...	0.055555...	0.055555...	0.0
"Lady"	0.045248...	0.095022...	0.095022...	0.095022...	0.045248...	0.022624...	0.045248...	0.045248...	0.095022...	0.095022...	0.022624...	0.045248...	0.045248...	0.022624...	0.022624...	0.022624...	0.022624...
"female_id"	0.064516...	0.064516...	0.064516...	0.064516...	0.064516...	0.032258...	0.064516...	0.064516...	0.064516...	0.064516...	0.032258...	0.064516...	0.064516...	0.032258...	0.032258...	0.032258...	0.032258...
"school_id"	0.068965...	0.034482...	0.068965...	0.068965...	0.068965...	0.068965...	0.034482...	0.068965...	0.034482...	0.034482...	0.034482...	0.034482...	0.034482...	0.068965...	0.068965...	0.068965...	0.068965...

Figure 6.11: *The Transition Matrix of the toy example showed in HOWERD*

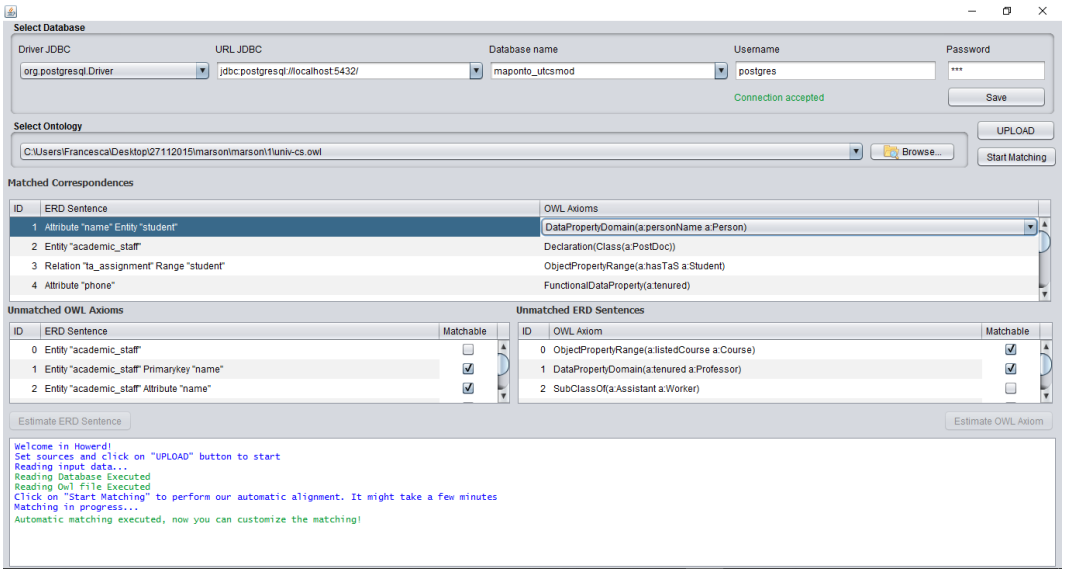


Figure 6.12: *The user-level HOWERD desktop tool*

any single part of the process. In particular from left to right there are: the main starting button, the reasoning button to switch on the reasoning process, the emission matrix viewer (Figure 6.10), the transition matrix viewer (Figure 6.11), a button to show the right panel that shows the most likely sequences as in the transformation process, the editing button for editing the correspondences rules whetever new ones are detected by an expert user.

The white areas in the bottom show the input axioms and the corresponding list of possible mappings. These lists in the other two version are showed as combo-boxes, alllowing user to choose the suitable mapping.

The Figure 6.12 shows the user desktop tool window. On the top of the window a form alows user to set up his own input database, selecting it and saving the connection informations as an XML file (Figure 6.14) such to facilitate a following reuse of the same connection. In the same way the user can browse in his filesystem the OWL ontology. Uploading the inputs the system extracts the OWL axioms and the ERD sentences, then it is possible to start the matching that fills the matched correspondences table. User can interact by the GUI, selecting the preferred matched correspondence in the combo-box (as in Figure 6.13). The OWL axioms and the ERD sentences not involved in a matching correspondence can be transformed clicking on the two estimation buttons. The white area in the bottom guides the user in using the application. The Figure 6.15 shows the single-page web application; the page sections and the behavior is the same described above for the desktop tool.

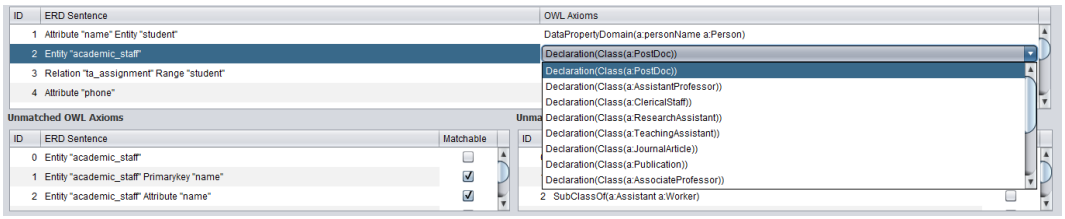


Figure 6.13: *A particular of the HOWERD desktop tool: selecting the preferred matched correspondence*

```
<?xml version="1.0" encoding="UTF-8"?>
<SavedDBList>
  <Database>
    <Driver>org.postgresql.Driver</Driver>
    <JDBCUrl>-- Unselected JDBC url --</JDBCUrl>
    <DBName>-- Unselected DB name --</DBName>
    <DBUserName>-- Username --</DBUserName>
    <DBPassword />
  </Database>
  <Database>
    <Driver>org.postgresql.Driver</Driver>
    <JDBCUrl>jdbc:postgresql://localhost:5432/</JDBCUrl>
    <DBName>maponto_utcsmod</DBName>
    <DBUserName>postgres</DBUserName>
    <DBPassword>123</DBPassword>
  </Database>
  <Database>
    <Driver>org.postgresql.Driver</Driver>
    <JDBCUrl>jdbc:postgresql://localhost:5432/</JDBCUrl>
    <DBName>example3</DBName>
    <DBUserName>pguser</DBUserName>
    <DBPassword>abc</DBPassword>
  </Database>
</SavedDBList>
```

Figure 6.14: *The XML file of saved database configurations*

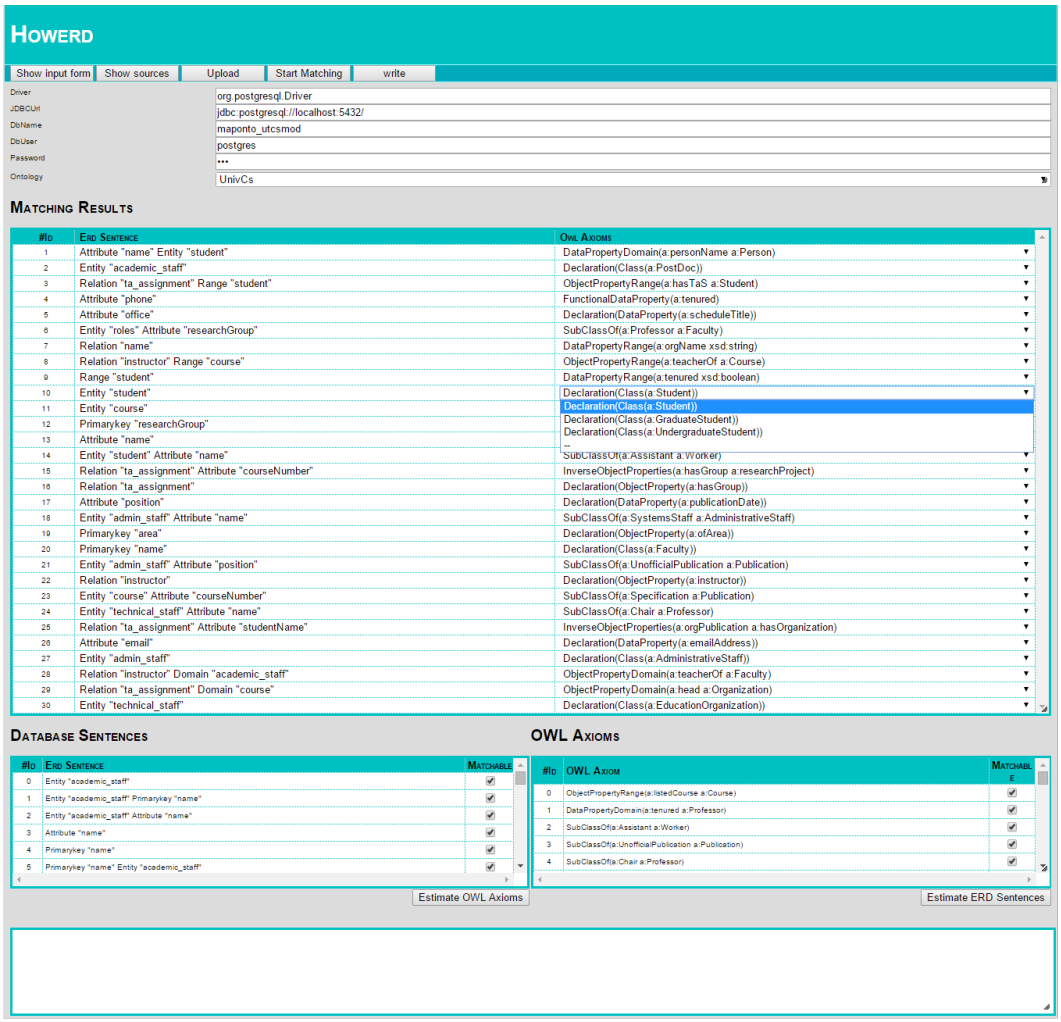


Figure 6.15: *The HOWERD Web interface*

Chapter 7

Experimental discussion

The quality of the alignment approach proposed in this thesis has been evaluated on a comparative analysis of the Java implementation of the model versus some tools reported in the literature aiming at the same goals.

A suitable group of data sets was used in the analysis, which are detailed in section 7.1.

The experimental results was compared with the ones obtained in by the two most known methods in the literature described respectively by Hu and Qu [40] and Aumueller et al. [11].

The experiments are set up to assessing the proposed alignment as regards both mapping and transformation processes.

In both $ERD \rightarrow OWL$ and $OWL \rightarrow ERD$ transformation processes, the experiments were aimed at analyzing the loss of data caused by the transformations, and the correctness of the generated new constructs was to be evaluated. For addressing these issues, each transformed representation was transformed back to its original form. Correctness was measured considering new constructs, which were not present in the original representation. The proposed system has been tested using one hundred of ERD and OWL fragments belonging to five different ontologies and SQL database available in the public domain.

Given so the start representation R_s and the retransformed one R_r , being $|R|$ the numbers of constructs in the sets, the following measures has been used:

- *similarity rate* defined as $sr = \frac{|R_s \bigcap R_r|}{|R_s|}$;
- *loss of data rate* defined as $lr = \frac{|R_s - R_r|}{|R_s|}$;
- *error rate* defined as $er = \frac{|R_r - R_s|}{|R_s|}$;

The average values obtained for the $OWL \rightarrow ERD$ and $ERD \rightarrow OWL$ transformations are reported in Table 7.1, and they are generally very satisfactory.

	$ERD \rightarrow OWL$	$OWL \rightarrow ERD$
sr	0.86	0.72
lr	0.09	0.12
er	0.05	0.16

Table 7.1: $ERD \rightarrow OWL$ *and* $OWL \rightarrow ERD$ *transformation evaluation*

For the mapping approach, more complex experiments was set up. They are described in section 7.2.

7.1 Experimental setup: datasets and gold standards

For evaluating the quality of the $OWL \leftrightarrow ERD$ mapping process has been used some datasets, each composed by an OWL ontology and a SQL database, related to the same domain along with a *gold standard*, that lists the correct correspondences among the constructs of the representations in the dataset.

The datasets are modeled on a variety of real world domains, and in all cases the related database schema and the ontology have been developed independently.

The chosen datasets are the same used by Hu and Qu [40] in MARSON approach evaluation; they include the following databases:

- the Department of Computer Science database in University of Toronto (UTCS);
- the VLDB Conference database (*VLDB*);
- the DBLP computer science bibliography database (*DBLP*);
- the test schemas in the OBSERVER project (*Observer*);
- the COUNTRY database appearing in [44] (*Country*).

All ontologies involved in the experiments are described in OWL (Web Ontology Language) and are the following:

- the Academic Department Ontology in the DAML library (*UnivCS*);
- the Academic Conference Ontology from the SchemaWeb ontology repository (*Conf*);
- the Bibliography ontology in the library of the Stanford's Ontolingua server (*Bibliog*);

Task	Database	Tables	Fields	Ontology	Classes	Object Prop.	Data Prop.
1	UTCS	8	32	UnivCS	55	25	10
2	VLDB	9	38	Conf	18	18	11
3	DBLP	5	27	Bibliog	66	18	63
4	Observer	8	112	Bibliog	66	18	63
5	Country	7	22	Fact	43	42	169

Table 7.2: *The datasets characteristics (T=number of Tables, C=number of Columns, Cl=number of Classes, OP=number of Object Properties, DP=number of Data Properties)*

- and the CIA Factbook Ontology (*Fact*).

All these datasets are freely available at the site of the datasets provider [1], and the characteristics are summarized in Table 7.2: for each database it indicates the number of tables and fields, while for each ontology the number of classes, object properties and data properties.

Hu and Qu [40] provide also a gold standard, builded according to the mappings manually defined by volunteers. When running the comparative analysis this gold standard resulted lacking of many possible real mappings. So, the gold standard has been extended with other possible mappings.

Table 7.3 shows the gold standard provided by Marson's authors and its extended version for the UTCS/UnivCS dataset, the first experimental task; i.e. in *Marson Gold Standard (MGS)* the Class `Faculty` is mapped with the Entity `academic_staff`, but the Class `Professor` that is ontologically defined as a subclass of `Faculty` is unmapped, although the correspondence between `Professor` and `Faculty` may be valid so that correspondence is added in the *Extended Gold Standard (EGS)*

7.2 Mapping results comparison

As states in describing the datasets, in the evaluation of the proposed $OWL \leftrightarrow ERD$ alignment technique, whose Java implementation is called HOWERD, a data set has to be composed by an OWL ontology, a SQL and the correspondences has to be evaluated in respect to the gold standard.

Due the fact that the gold standard was extended (see for details the section 7.1 and that it implies an ambiguous trend of the results, the experimental setup for the alignment process need to consider two threads, one for each gold standard.

Table 7.6 shows the experimental results found out by HOWERD, compared to the ones of Marson [40] and COMA++ [11] in respect to the gold standard.

[1] http://www.cs.toronto.edu/ yuana/research/ maponto/relational/testData.html

OWL	ERD	NGS	EGS
AdministrativeStaff	admin_staff	✓	✓
advisor	supervisor	✓	✓
alumnus	student	✓	✓
Assistant	academic_staff	✓	
AssistantProfessor	academic_staff	✓	
AssociateProfessor	academic_staff	✓	
Chair	academic_staff	✓	
Chair	admin_staff	✓	
ClericalStaff	admin_staff	✓	
ClericalStaff	technical_staff	✓	
Course	course	✓	✓
Dean	academic_staff	✓	
Dean	admin_staff	✓	
Director	academic_staff	✓	
emailAddress	academic_staff.email	✓	✓
emailAddress	admin_staff.email	✓	✓
emailAddress	student.email	✓	✓
emailAddress	technical_staff.email	✓	✓
Faculty	academic_staff	✓	✓
FullProfessor	academic_staff	✓	
GraduateStudent	student	✓	
hasGroup	researchGroup	✓	
hasTaS	ta_assignment	✓	✓
instructor	instructor	✓	✓
Lecturer	academic_staff	✓	
listedCourse	course	✓	
ofArea	area	✓	
ofArea	areas_of_interest	✓	
orgName	areas_of_interest.name	✓	
Person	academic_staff	✓	
Person	admin_staff	✓	
Person	student	✓	
Person	technical_staff	✓	
personName	academic_staff.name	✓	✓
personName	admin_staff.name	✓	✓
personName	student.name	✓	✓
personName	technical_staff.name	✓	✓
PostDoc	academic_staff	✓	
Professor	academic_staff	✓	
Research	roles.researchGroup	✓	
ResearchAssistant	academic_staff	✓	
ResearchGroup	researchGroup	✓	
researchInterest	areas_of_interest	✓	✓
researchProject	roles.researchGroup	✓	
Schedule	Course	✓	
scheduleTitle	courseTitle	✓	
Student	student	✓	✓
SystemsStaff	technical_staff	✓	
takesCourse	course.ta_assignment	✓	
teacherOf	course.instructor	✓	✓
TeachingAssistant	academic_staff	✓	
teachingAssistantOf	instructor	✓	
UndergraduateStudent	student	✓	
VisitingProfessor	academic_staff	✓	
Worker	admin_staff	✓	
Worker	academic_staff	✓	
Worker	technical_staff	✓	
workTitle	course.courseTitle	✓	✓

Table 7.3: *The Gold Standards comparison for the dataset UTCS/UnivCS*

The set of well-known measures were considered: the *Precision* p, the *Recall* r, and the *F1-measure* $f1$. The measures are computed comparing the "set of the derived matches" D_m , which contains the matches identified automatically by the tool, with the "set of the real matches" R_m resulted from the gold standard. The set D_m is the union set of the true positives set T_p , and the false positives set F_p , so $D_m = T_p \bigcup F_p$. On the other hand, the set of the real matches R_m is the union set of the false negatives set F_n and T_p , so $R_m = F_n \bigcup T_p$. The three measures we referred were defined as follows:

$$p = \frac{|T_p|}{|D_m|} \tag{7.1}$$

$$r = \frac{|T_p|}{|R_m|} \tag{7.2}$$

$$f1 = 2 \cdot \frac{p \cdot r}{p + r} \tag{7.3}$$

The table reports the value of *True Positives (T_p) False Positives (F_p)*, *True Negatives (T_n)* and *False Negatives (F_n)* for each dataset; the $f1$ measure was also computed.

It is worth noting that HOWERD discovers more TPs than the others; this is brought by the fact that our tool estimates not only the label correspondences, but evaluates also the possible structural matches. In this way, many other correspondences can be considered even if labels can not be matched directly.

In the first thread, whose results are shown in Table 7.4 and in Figure 7.1, we considered the original gold standard as provided at the cited site, and HOWERD has the best recall values in all cases, while p and $f1$ measures are never good.

This behavior has a specific reason: we noticed that the provided gold standard missing many possible mappings that HOWERD discovers that are objectively correct when inspected by a human operator. As a consequence, many false positives produced by our tool over the original gold standards are not false positives, but true positives; therefore they are additional false negatives for the others tools, and they were not really counted.

Although the gold standards lack such mappings, we had the better recall in all cases: our tool estimates always something in correspondence of the OWL constructs, and the estimation is often a true positive; the system produced the lowest number of false negatives than the others, and r increased consequently.

In the second thread the Extended Gold Standard is considered and the results were recomputed for all the tools in respect to such new gold standards, the results were very satisfactory for HOWERD as reported in Table 7.5 and in Figure7.2 because the true positives increased in all cases like p and $f1$.

	UTCS/UnivCS			VLDB/Conf			DBLP/Bibliog			Observer/Bibliog			Country/Fact		
	M	**C**	**H**	**M**	**C**	**H**	**M**	**C**	**H**	**M**	**C**	**H**	**M**	**C**	**H**
TP	15	9	**18**	25	19	**34**	16	14	**20**	**57**	33	33	**16**	9	**16**
FP	12	10	59	3	9	13	11	9	131	25	42	129	5	13	224
TN	68	70	21	19	16	0	119	121	0	74	72	0	213	208	0
FN	3	9	0	4	7	4	5	7	0	6	15	0	6	10	0
F1	**0.67**	0.49	0.38	**0.88**	0.70	0.80	**0.67**	0.64	0.23	**0.79**	0.54	0.34	**0.74**	0.44	0.13

Table 7.4: *Compared results using Marson's gold standard (M=Marson, C=Coma++, H=Howerd)*

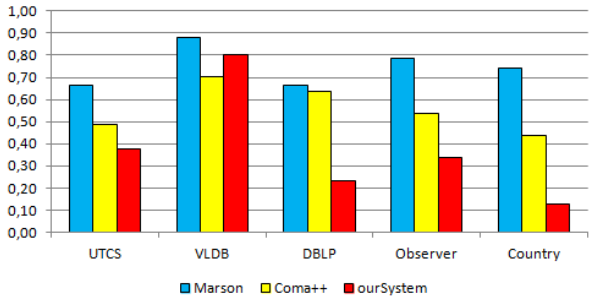


Figure 7.1: *F1-Measure (Marson's Gold Standard)*

	UTCS/UnivCS			VLDB/Conf			DBLP/Bibliog			Observer/Bibliog			Country/Fact		
	M	**C**	**H**	**M**	**C**	**H**	**M**	**C**	**H**	**M**	**C**	**H**	**M**	**C**	**H**
TP	23	12	**43**	28	22	**37**	26	23	**87**	84	56	**96**	27	16	**37**
FP	4	8	40	1	6	12	1	0	59	27	31	103	1	7	214
TN	46	44	16	6	5	0	51	51	0	21	20	0	208	210	0
FN	31	40	5	18	20	4	73	77	5	67	92	0	15	18	0
F1	0.57	0.33	**0.66**	0.75	0.63	**0.82**	0.41	0.37	**0.73**	0.64	0.48	**0.77**	0.56	0.26	**0.65**

Table 7.5: *Compared results using the Extended Gold Standard*

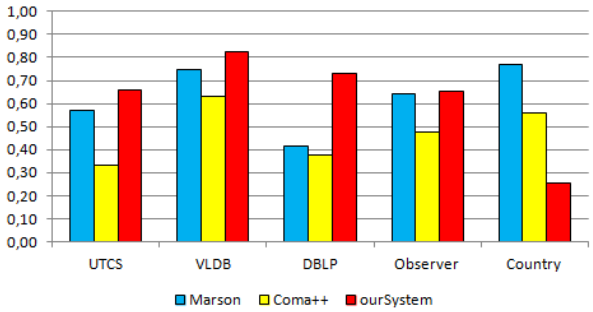


Figure 7.2: *F1-Measure (Extended Gold Standard)*

	Experimental Task				
	1	2	3	4	5
Observable States	99	66	311	160	306
Hidden States	28	41	33	117	27
Axioms	229	160	636	425	484
Time Emission Comp.(ms)	23126	17437	153908	288332	71516
Time Transition Comp.(ms)	157	94	62	203	78
Time Viterbi Comp.(ms)	485	344	2953	1437	1562
Total Time (ms)	24860	18641	157720	291160	73907

Table 7.6: *Time performances*

There are two main reasons for this behavior: first, the proposed system considers both structural and label constrains when building the mappings, while the other tools do not keep into account the structural aspect. As a consequence, there are more false positives for the other tools, because they often match constructs that should not be in correspondence due to their structural characteristics; this is the case of an OWL object property that often is mapped to an entity since they have a similar label, while an object property can be considered as a relation, primary key and never as an entity in a database as it emerges from the literature rules. Such a mapping is never a true positive. As false positives grown, p decreased in the other tools, while HOWERD p value increased due to the increased number of real mappings it discovers.

Second, HOWERD considers the semantic similarity of the labels using the Wu-Palmer distance that is known to be the best measure for this purpose; as a consequence, it find more label mappings than the others. For example, in the third experiment the mapping between the table doc and the classes Document, Article, Book are not trivial, in fact they are not all retrieved by the others. Instead we retrieved these kinds of correspondences.

The setting environment in these experiments was the following: Intel Core i7-4510U @2.00GHz 2.60GHz processor, 8GB RAM, Windows 10 and Java SE 6. The execution time strictly depends on the dimension of dataset and labels. The computation of semantic similarities, useful for the computation of emission matrix, takes an huge part of time. This is caused by Wordnet interfacing, expecially when the labels are composed by many tokens. The performance of Viterbi computation is strictly dependent on the number of ontology axioms.

Chapter 8

Conclusions and future developments

This thesis introduces an innovative approach to the alignment process, addressed to solve the semantic gap in any alignment scenarios where one tries to devise a computational model for automatic alignment between two T-Boxes in order to use it for Ontology Based Data Access and ontology storing systems.

The idea is so to provide an automatic system that helps the user in creating mapping rules wich can be used in the OBDA scenario, aiming at improving the querying of a database through a conceptual level and, at the same time, to provide a system that allows to ensure a persistent storage of any information modeled conceptually into a well structured set of data reducing the information lacks.

The issue of linking relational and ontological level is particularly outstanding in those contexts where many subjects need to interoperate and share informations frequently changing. It is typical of corporate enviroments, that constitute an interesting application enviroment for the techniques proposed. The target in this sense could be identified in public administrations or big companies, where the knowledge management system changes dynamically and continuosly and where the data are distibuted along the all organization system and where departments or agencies informative systems need to use their own vocabulary for internal information management purposes but also to refer to a big conceptual net for sharing and retrieving tasks.

The methodology is devised according to the different alignment scenarios treated, over the years, in many works in literature in sense both of transformation and mapping, where transformation indicates the process of translating any single element of the first representation in an element of the other one representation (referred as $ERD \rightarrow OWL$ or $OWL \rightarrow ERD$ for the complementar task) and mapping (referred as $OWL \leftrightarrow ERD$) is the process that allows to create some

correspondences rules between the two representation in both directions.

For $ERD \rightarrow OWL$ the most known procedures in literature are used for the purpose, collecting the rules operating in this sense. When using these procedures a suitable repository of the ERD-OWL couples has been created using some constructs correspondences derived from the works in literature operating in any sense of alignment and extending them through the observation of a set of ERD and OWL schemas couples in real-world domains.

For the other two scenarios the proposed techniques achieve the goal by means of a Hidden Markov Model to estimate the most likely sequence of symbols in the database, expressed as statements in a suitable language, that correspond to the constructs of an OWL axiom in the ontology.

In both methods the OWL constructs are the observable states in the HMM, while hidden states are the symbols of a Context-Free Grammar (CFG) defined purposely for the lexical description of the database.

The $OWL \rightarrow ERD$ process has as input only the ontology so the CFG refers to any possible ERD, and does not consider the labels as constructs so the process estimates the most likely sequence of ERD constructs that approximates the sequence of structural OWL constructs, while the $OWL \leftrightarrow ERD$ process has as inputs both an ontology and a database, so a customized grammar is set up at any single iteration on the given database, so also the ontological and relational labels are inclused as descriptive constructs.

Due the differences between the transformation and the mapping approaches computing the emission and transition probabilities is quite different. In particular, since in $OWL \leftrightarrow ERD$ process the labels are involved the emission is computed also on defining a similarity measure between the labels. The measure takes account of a syntactical feature and a semantic feature, as to bridge the problems arising from the use of strings in real-world domains.

The work reports the techniques description, along with the architectural design and the implementation of the interactive tools, one for the single $OWL \rightarrow ERD$ transformation process called OMEGA (OWL $\rightarrow$ Markov $\rightarrow$ ERD generation application) and one for the mapping process called HOWERD (HMM OWL $\leftrightarrow$ ERD), which supports the user in selecting and/or creating the correct matches based on her knowledge modeling needs.

The user interaction is also a fundamental component of the mapping process itself, so the systems own their suitable GUIs allowing users to interact with the overall alignment process.

The mapping tool has been tested against the main alignment approaches reported in the literature, and the results are satisfactory.

Future works can be devoted to integrate the system in a more general framework for querying a database using Natural Language. In such a scenario, the NL query is mapped to an OWL fragment that in turn forms the basis for running the

system.

The probabilistic techniques can further improved increasing the order of the model, or applying some generalized models allowing to make a correspondence between a single observation and a sequence of states.

The feasible applications in enterprise enviroments can suggest also to attempt to apply the probabilistic techniques with big databases and with multiple data sources, without simply interating the approach for any single involved database.

Bibliography

[1] N. Alalwan, H. Zedan, and F. Siewe. Generating owl ontology for database integration. In *Advances in Semantic Processing, 2009. SEMAPRO '09. Third International Conference on*, pages 22–31, Oct 2009. doi: 10.1109/SEMAPRO.2009.21.

[2] Khalid M Albarrak and Edgar H Sibley. A survey of methods that transform data models into ontology models. In *Information Reuse and Integration (IRI), 2011 IEEE International Conference on*, pages 58–65. IEEE, 2011.

[3] Patrick Arnold. Semantic enrichment of ontology mappings: Detecting relation types and complex correspondences. In *Grundlagen von Datenbanken*, pages 34–39, 2013.

[4] Patrick Arnold and Erhard Rahm. Semantic enrichment of ontology mappings: A linguistic-based approach. In *Advances in Databases and Information Systems*, pages 42–55. Springer, 2013.

[5] Patrick Arnold and Erhard Rahm. Enriching ontology mappings with semantic relations. *Data & Knowledge Engineering*, 93:1–18, 2014.

[6] Irina Astrova. Towards the semantic web-an approach to reverse engineering of relational databases to ontologies. *ADBIS research communications*, 152, 2005.

[7] Irina Astrova. Rules for mapping sql relational databases to owl ontologies. In *Metadata and Semantics*, pages 415–424. Springer, 2009.

[8] Irina Astrova, Ahto Kalja, and Nahum Korda. Automatic transformation of owl ontologies to sql relational databases. In *IADIS European Conf. Data Mining (MCCSIS)*, pages 5–7, 2007.

[9] Irina Astrova, Nahum Korda, and Ahto Kalja. Rule-Based Transformation of SQL Relational Databases to OWL Ontologies. *Transformation*, pages 1–16, 2007. doi: 10.1007/978-0-387-77745-0. URL

http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.110.8189&rep=rep1&type=pdf.

[10] Irina Astrova, Nahum Korda, and Ahto Kalja. Storing owl ontologies in sql relational databases. *International Journal of Electrical, Computer and Systems Engineering*, 1(4):242–247, 2007.

[11] David Aumueller, Hong-Hai Do, Sabine Massmann, and Erhard Rahm. Schema and ontology matching with coma++. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, SIGMOD '05, pages 906–908, New York, NY, USA, 2005. ACM. ISBN 1-59593-060-4. doi: 10.1145/1066157.1066283. URL http://doi.acm.org/10.1145/1066157.1066283.

[12] David Aumueller, Hong-Hai Do, Sabine Massmann, and Erhard Rahm. Schema and ontology matching with coma++. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 906–908. ACM, 2005.

[13] John W Backus. The syntax and semantics of the proposed international algebraic language of the zurich acm-gamm conference. *Proceedings of the International Comference on Information Processing, 1959*, 1959.

[14] Timea Bagosi, Diego Calvanese, Josef Hardi, Sarah Komla-Ebri, Davide Lanti, Martin Rezk, Mariano Rodríguez-Muro, Mindaugas Slusnys, and Guohui Xiao. The ontop framework for ontology based data access. In *The Semantic Web and Web Science*, pages 67–77. Springer, 2014.

[15] Sikha Bagui. Mapping owl to the entity relationship and extended entity relationship models. *International Journal of Knowledge and Web Intelligence*, 1(1-2):125–149, 2009.

[16] Richard Barker and Cliff Longman. *Case* Method*. Addison Wesley, 1990.

[17] Leonard E Baum and Ted Petrie. Statistical inference for probabilistic functions of finite state markov chains. *The annals of mathematical statistics*, pages 1554–1563, 1966.

[18] Dave Beckett and Brian McBride. Rdf/xml syntax specification (revised). *W3C recommendation*, 10, 2004.

[19] Sidi Mohamed Benslimane, Djamal Benslimane, Mimoun Malki, Youssef Amghar, and Hamadou Saliah-Hassane. Acquiring owl ontologies from data-intensive web sites. In *Proceedings of the 6th International Conference on*

Web Engineering, ICWE '06, pages 361–368, New York, NY, USA, 2006. ACM. ISBN 1-59593-352-2. doi: 10.1145/1145581.1145593. URL `http://doi.acm.org/10.1145/1145581.1145593`.

[20] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Data complexity of query answering in description logics. *KR*, 6:260–270, 2006.

[21] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, and Riccardo Rosati. Ontology-based database access. In *SEBD*, pages 324–331, 2007.

[22] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Eql-lite: Effective first-order query processing in description logics. In *IJCAI*, volume 7, pages 274–279, 2007.

[23] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The dl-lite family. *Journal of Automated reasoning*, 39 (3):385–429, 2007.

[24] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Path-based identification constraints in description logics. In *KR*, pages 231–241, 2008.

[25] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Conceptual modeling for data integration. In *Conceptual Modeling: Foundations and Applications*, pages 173–197. Springer, 2009.

[26] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Antonella Poggi, Mariano Rodriguez-Muro, Riccardo Rosati, Marco Ruzzi, and Domenico Fabio Savo. The mastro system for ontology-based data access. *Semantic Web*, 2(1):43–53, 2011.

[27] Roger HL Chiang, Terence M Barron, and Veda C Storey. Reverse engineering of relational databases: Extraction of an eer model from a relational database. *Data & Knowledge Engineering*, 12(2):107–142, 1994.

[28] Kenneth Ward Church. A stochastic parts program and noun phrase parser for unrestricted text. In *Proceedings of the second conference on Applied natural language processing*, pages 136–143. Association for Computational Linguistics, 1988.

[29] World Wide Web Consortium et al. Document object model (dom) level 3 core specification. 2004.

[30] Michael E. Cotterell and Terrance Medina. A markov model for ontology alignment. *CoRR*, abs/1304.5566, 2013. URL `http://arxiv.org/abs/1304.5566`.

[31] Mona Dadjoo and Esmaeil Kheirkhah. An approach for transforming of relational databases to owl ontology. *arXiv preprint arXiv:1502.05844*, 2015.

[32] Steven J DeRose. Grammatical category disambiguation by statistical optimization. *Computational Linguistics*, 14(1):31–39, 1988.

[33] Hong-Hai Do and Erhard Rahm. Coma: a system for flexible combination of schema matching approaches. In *Proceedings of the 28th international conference on Very Large Data Bases*, pages 610–621. VLDB Endowment, 2002.

[34] Xin Luna Dong, Alon Halevy, and Cong Yu. Data integration with uncertainty. *The VLDB Journal-The International Journal on Very Large Data Bases*, 18(2):469–500, 2009.

[35] Muhammad Fahad. Er2owl: Generating owl ontology from er diagram. In *Intelligent Information Processing IV*, pages 28–37. Springer, 2008.

[36] Anuradha Gali, Cindy X Chen, Kajal T Claypool, and Rosario Uceda-Sosa. From ontology to relational databases. In *Conceptual modeling for advanced application domains*, pages 278–289. Springer, 2004.

[37] Asunción Gómez-Pérez, Mariano Fernández-López, and Oscar Corcho. *Ontological Engineering: with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer Science & Business Media, 2006.

[38] Alon Halevy, Anand Rajaraman, and Joann Ordille. Data integration: the teenage years. In *Proceedings of the 32nd international conference on Very large data bases*, pages 9–16. VLDB Endowment, 2006.

[39] R. Harrison and C.W. Chan. Distributed ontology management system. In *Electrical and Computer Engineering, 2005. Canadian Conference on*, pages 661–664, May 2005. doi: 10.1109/CCECE.2005.1557017.

[40] Wei Hu and Yuzhong Qu. Discovering simple mappings between relational database schemas and ontologies. In Karl Aberer, Key-Sun Choi,

Natasha Noy, Dean Allemang, Kyung-Il Lee, Lyndon Nixon, Jennifer Golbeck, Peter Mika, Diana Maynard, Riichiro Mizoguchi, Guus Schreiber, and Philippe Cudré-Mauroux, editors, *The Semantic Web*, volume 4825 of *Lecture Notes in Computer Science*, pages 225–238. Springer Berlin Heidelberg, 2007. ISBN 978-3-540-76297-3. doi: 10.1007/978-3-540-76298-0_17. URL `http://dx.doi.org/10.1007/978-3-540-76298-0_17`.

[41] Ernesto Jiménez-Ruiz and Bernardo Cuenca Grau. Logmap: Logic-based and scalable ontology matching. In *The Semantic Web–ISWC 2011*, pages 273–288. Springer, 2011.

[42] Ernesto Jiménez-Ruiz, Bernardo Cuenca Grau, Yujiao Zhou, and Ian Horrocks. Large-scale interactive ontology matching: Algorithms and implementation. In *ECAI*, volume 242, pages 444–449, 2012.

[43] Ernesto Jiménez-Ruiz, Evgeny Kharlamov, Dmitriy Zheleznyakov, Ian Horrocks, Christoph Pinkel, Martin G Skjæveland, Evgenij Thorstensen, and Jose Mora. Bootox: practical mapping of rdbs to owl 2. In *The Semantic Web-ISWC 2015*, pages 113–132. Springer, 2015.

[44] P. Johannesson. A method for transforming relational schemas into conceptual schemas. In *Data Engineering, 1994. Proceedings.10th International Conference*, pages 190–201, Feb 1994. doi: 10.1109/ICDE.1994.283030.

[45] Donald E Knuth. Backus normal form vs. backus naur form. *Communications of the ACM*, 7(12):735–736, 1964.

[46] Maurizio Lenzerini. Data integration: A theoretical perspective. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 233–246. ACM, 2002.

[47] Man Li, Xiao-Yong Du, and Shan Wang. Learning ontology from relational database. In *Machine Learning and Cybernetics, 2005. Proceedings of 2005 International Conference on*, volume 6, pages 3410–3415 Vol. 6, Aug 2005. doi: 10.1109/ICMLC.2005.1527531.

[48] Dekang Lin. An information-theoretic definition of similarity. In *ICML*, volume 98, pages 296–304, 1998.

[49] Lili Lin, Zhuoming Xu, and Ying Ding. Owl ontology extraction from relational databases via database reverse engineering. *Journal of Software*, 8(11): 2749–2760, 2013.

[50] Frank Manola, Eric Miller, Brian McBride, et al. Rdf primer. *W3C recommendation*, 10(1-107):6, 2004.

[51] Sabine Massmann, Salvatore Raunich, David Aumüller, Patrick Arnold, and Erhard Rahm. Evolution of the coma match system. *Ontology Matching*, 49, 2011.

[52] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 117–128. IEEE, 2002.

[53] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[54] Prasenjit Mitra, NatashaF. Noy, and AnujRattan Jaiswal. Omen: A probabilistic ontology mapping tool. In Yolanda Gil, Enrico Motta, V.Richard Benjamins, and MarkA. Musen, editors, *The Semantic Web - ISWC 2005*, volume 3729 of *Lecture Notes in Computer Science*, pages 537–547. Springer Berlin Heidelberg, 2005. ISBN 978-3-540-29754-3. doi: 10.1007/11574620_39. URL `http://dx.doi.org/10.1007/11574620_39`.

[55] Boris Motik, Peter F. Patel-Schneider, Bijan Parsia, Conrad Bock, Achille Fokoue, Peter Haase, Rinke Hoekstra, Ian Horrocks, Alan Ruttenberg, Uli Sattler, and Michael Smith. OWL 2 Web Ontology Language - Structural Specification and Functional-Style Syntax (Second Edition). *Online*, pages 1–133, 2012.

[56] I. Myroshnichenko and M.C. Murphy. Mapping er schemas to owl ontologies. In *Semantic Computing, 2009. ICSC '09. IEEE International Conference on*, pages 324–329, Sept 2009. doi: 10.1109/ICSC.2009.61.

[57] Natalya F Noy, Deborah L McGuinness, et al. Ontology development 101: A guide to creating your first ontology, 2001.

[58] Daniel E. O'Leary. Enterprise ontologies: Review and an activity theory approach. *International Journal of Accounting Information Systems*, 11(4):336 – 352, 2010. ISSN 1467-0895. doi: http://dx.doi.org/10.1016/j.accinf.2010.09.006. URL `http://www.sciencedirect.com/science/article/pii/S1467089510000722`.

[59] Petros Papapanagiotou, Polyxeni Katsiouli, Vassileios Tsetsos, Christos Anagnostopoulos, and Stathes Hadjiefthymiades. Ronto: Relational to ontology schema matching. *AIS Sigsemis Bulletin*, 3(3-4):32–36, 2006.

[60] Ted Pedersen, Siddharth Patwardhan, and Jason Michelizzi. Wordnet:: Similarity: measuring the relatedness of concepts. In *Demonstration papers at hlt-naacl 2004*, pages 38–41. Association for Computational Linguistics, 2004.

[61] Arianna Pipitone and Roberto Pirrone. A hidden markov model for automatic generation of er diagrams from owl ontology. In *Proceedings of the 2014 IEEE International Conference on Semantic Computing*, ICSC '14, pages 135–142, Washington, DC, USA, 2014. IEEE Computer Society. ISBN 978-1-4799-4003-5. doi: 10.1109/ICSC.2014.19. URL `http://dx.doi.org/10.1109/ICSC.2014.19`.

[62] Antonella Poggi, Domenico Lembo, Diego Calvanese, Giuseppe De Giacomo, Maurizio Lenzerini, and Riccardo Rosati. Linking data to ontologies. In *Journal on data semantics X*, pages 133–173. Springer, 2008.

[63] Yuzhong Qu, Wei Hu, and Gong Cheng. Constructing virtual documents for ontology matching. In *Proceedings of the 15th international conference on World Wide Web*, pages 23–31. ACM, 2006.

[64] Salvatore Raunich and Erhard Rahm. Target-driven merging of taxonomies. *arXiv preprint arXiv:1012.4855*, 2010.

[65] Salvatore Raunich and Erhard Rahm. Towards a benchmark for ontology merging. In *On the Move to Meaningful Internet Systems: OTM 2012 Workshops*, pages 124–133. Springer, 2012.

[66] Mariano Rodríguez-Muro, Roman Kontchakov, and Michael Zakharyaschev. Ontology-based data access: Ontop of databases. In *The Semantic Web–ISWC 2013*, pages 558–573. Springer, 2013.

[67] G. Russo, F. Anastasio, A. Pipitone, A. Gentile, and R. Pirrone. Vebo: Validation of e-r diagrams through ontologies and wordnet. In *Semantic Computing (ICSC), 2012 IEEE Sixth International Conference on*, pages 342–344, Sept 2012. doi: 10.1109/ICSC.2012.41.

[68] Gerard Salton and Michael J McGill. Introduction to modern information retrieval. 1986.

[69] Guus Schreiber, Bob Wielinga, and Wouter Jansweijer. The kactus view on the 'o'word. In *IJCAI workshop on basic ontological issues in knowledge sharing*, pages 159–168, 1995.

[70] Alessandro Solimando, Ernesto Jiménez-Ruiz, and Giovanna Guerrini. Detecting and correcting conservativity principle violations in ontology to ontology mappings. In *The Semantic Web–ISWC 2014*, pages 1–16. Springer, 2014.

[71] Fabian M Suchanek, Serge Abiteboul, and Pierre Senellart. Paris: Probabilistic alignment of relations, instances, and schema. *Proceedings of the VLDB Endowment*, 5(3):157–168, 2011.

[72] Quang Trinh, K. Barker, and R. Alhajj. Rdb2ont: A tool for generating owl ontologies from relational database systems. In *Telecommunications, 2006. AICT-ICIW '06. International Conference on Internet and Web Applications and Services/Advanced International Conference on*, pages 170–170, Feb 2006. doi: 10.1109/AICT-ICIW.2006.159.

[73] Sujatha R Upadhyaya and P Sreenivasa Kumar. Eronto: a tool for extracting ontologies from extended e/r diagrams. In *Proceedings of the 2005 ACM symposium on Applied computing*, pages 666–670. ACM, 2005.

[74] Mike Uschold, Martin King, Stuart Moralee, and Yannis Zorgios. The enterprise ontology. *The knowledge engineering review*, 13(01):31–89, 1998.

[75] Ernestas Vysniauskas and Lina Nemuraite. Transforming ontology representation from OWL to relational database. *Information technology and control*, 35(3):333–343, 2006.

[76] William E Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. 1990.

[77] Zhibiao Wu and Martha Palmer. Verbs semantics and lexical selection. In *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138. Association for Computational Linguistics, 1994.

[78] Zhuoming Xu, Shichao Zhang, and Yisheng Dong. Mapping between relational database schema and owl ontology for deep annotation. In *Proceedings of the 2006 IEEE/WIC/ACM international Conference on Web intelligence*, pages 548–552. IEEE Computer Society, 2006.

[79] Shufeng Zhou, Haiyun Ling, Mei Han, and Huaiwei Zhang. Ontology generator from relational database based on jena. *Computer and Information Science*, 3(2):p263, 2010.