Ontology Mapping Specification in Description Logics for Cooperative Systems

Thibault Poulain, Nadine Cullot and Kokou Yétongnon

Laboratoire d'Électronique, Informatique et Image
Université de Bourgogne
Bat. Mirande - BP 47870 21078 DIJON Cedex FRANCE

poulain@khali.u-bourgogne.fr, nadine.cullot@u-bourgogne.fr,
kokou.yetongnon@u-bourgogne.fr

Abstract

The rapid development of the semantic Web is associated with the specification of various ontologies to modelize domains or tasks by some communities of people. That leads to heterogeneous representations of some common domains or tasks that can have distinct or overlapped descriptions of information. In cooperative systems, it is necessary to align, merge or integrate these ontologies to solve queries or use web services taking advantage of theses agreed and shared knowledge. A key-point in these methodologies is the specification or the semi-automatic discovery of mappings between the concepts specified in ontologies. After a brief state of art of the existing methods and systems allowing the definition of mappings between ontologies, a methodology to semi-automatically discover mappings between ontologies is proposed. This method is illustrated by a running example and embedded in a general architecture that we also present.

1. Introduction

The definition of the "Semantic Web" originally introduced by Berners-Lee et al [13] in 2001 was born from the rapid development of Web technologies and the difficulty in managing increasingly large quantities of information. The development of models, methodologies, tools and architectures to provide automated access to data is becoming a great challenge to make the Web a real information sharing tool not only for people but also for computers.

Ontologies are a key point of technologies for the semantic Web. They allow the specification of the semantic of a domain. Their various models of representation are based on approaches resulting from logics or from databases [8]. Some languages based on description logics such as OWL-DL specified by the W3 Consortium are becoming standards. Description logics are a family of languages allowing formal representation of knowledge and providing reasoning tools for classification and consistency checking. The rapid development of multiple ontologies on various domains presses the need for methodologies to map or interconnect them into cooperative systems.

Cooperative systems must address data heterogeneity issues which may arise from syntactic, structural or semantic differences in data sources. The management and resolution of semantic heterogeneity is a great research challenge which requires methodologies and tools to specify data meanings. Mappings are necessary to specify correspondences between the local data meanings and the agreed definitions of shared and common data.

Mappings allow queries to be solved through various knowledge bases. The mapping generation process is non-trivial both manually and automatically. Numerous works have dealt with this problem. We present below some typical tools emphasing their strength and weakness.

The aim of the paper is to propose a framework for querying cooperative systems. The local data is represented by a local ontology which is mapped to one or more reference ontologies according to their application domain. A special focus is put on the discovery and the representation of the mappings from the local ontologies to a reference ontology. A running example is given using description logics to model the local and reference ontologies and their mappings. The paper is organized as follows. *Section 2* gives a brief reminder of description logics and some related works on mappings. *Section 3* presents the ontology mapping approach. *Section 4* describes the proposed cooperative framework. *Section 5* concludes the paper.

2. Related Works

In this section, we present a brief reminder of description logics and discuss existing mapping systems.

2.1. Description logics

Description Logics are a family of terminological formalism for specifying and reasoning on knowledge whose expressiveness depends on the considered constructors. They rely on two basic notions: concepts and roles.

Complex concepts and roles can be built from atomic concepts by using DL constructors. A detailed presentation of DL can be found in [4].

The basic $\mathcal{ALC}$ DL provides the following concept constructors: $\neg C$ (negation), $C \sqcap D$ (conjunction), $\forall R.C$ (value restriction) and $\exists R.C$ (exists restriction) where C and D are concepts and R is an atomic role.

The $\mathcal{SHIQ}$ DL [5] extends the basic $\mathcal{ALC}$ DL to provide an expressive language. $\mathcal{SHIQ}$ DL [5] is implemented in the RACER prover[14]. The elementary descriptions are the atomic concepts and roles, the universal concept $\top$ and the bottom concept $\bot$. The $\mathcal{SHIQ}$ DL includes additionally the inverse role $\mathcal{I}$, role hierarchies $\mathcal{H}$ and qualifying number restrictions $\mathcal{Q} \geq n\ R.C$ and $\leq n\ R.C$. Complex concepts can be defined using the inclusion ($\subseteq$) and equivalence ($\equiv$) axioms. For example, the following formula states that PhD students are included in the set of teachers who work in a laboratory.

PhDStudent $\subseteq$ *Teacher* $\cap \exists$ *works_in.Lab*

2.2. Mapping Tools

Significant works have been done on mapping specifications. We focus below on three relevant projects: GLUE, MAFRA and the Prompt suite project. They offer tools for the discovery and representation of mappings between ontologies.

The **GLUE**[1] project proposes an automatic mapping creation system which uses instances to determine which concepts should be matched. It relies heavily on learners and training methods borrowed from artificial intelligence to achieve concepts matching. Learners are trained with the instances of one ontology to classify data which belong or not to a specific concept.

The training phase used to compute mappings makes the whole process hardly efficient to add new ontologies on the fly. The fact that the method works on instances reduces the number of cases where it can be applied. It can not map ontologies without instances.

The **MAFRA**[11] (MApping FRAmework) project, provides an automatic mapping creation tool as a plug-in to the KAON (KArlsruhe ONtology) project. MAFRA defines the notion of *semantic bridges*. It describes relations between ontology entities such as concepts, relations, and attributes in a detailed way. The strength of this work is clearly the specification of mappings using semantic bridges allowing the representation of complex mappings. However, there is not a well-defined methodology or a tool to discover and specify mappings using this semantic brige formalism.

The Prompt tools suite[9] consists of several components which are integrated to Protege [10] as plug-ins. The project proposes a tool for merging ontologies. It uses a list of matching concepts to propose new mappings and detects conflicts with the mappings added by the user.

Prompt is only a semi-automatic tool, where human interaction is needed at each step. The mapping propagation rules rely only on the taxonomic structure of the ontologies. However, the fact that concepts of ontologies are not always organized into a single taxonomic tree weakens Prompt's approach.

3. Ontology mapping for a cooperative system

3.1. Overview of the approach

The analysis of existing mapping systems leads us to specify some relevant requirements for systems whose goals are to create and use mappings in a decentralized way. The system has to be evolutive to allow the addition of new methods of mapping discovery. It has to be as automatic as possible, reducing human interventions to the beginning and finishing phases only and to provide mapping discovery methods working on the specifications of the ontologies without taking into account their instances.

The system that we propose carries out these requirements. It manages the interaction between some clients with the help of servers working on specific knowledge domains.

Clients are systems that own local ontologies and their associated data. They interact with servers, and indirectly with any other client connected to the same server. To be able to take advantage of the knowledge of a server, a client needs to map its ontology to the ontology of the server. A client can be connected to several servers.

Each server has a domain ontology and several mapping tools. These tools provide specific mapping information to the clients wishing to connect to the server, as well as links to any external data which can be usefull to perform the mappings, such as an external global taxonomy, like WordNet [3]. The domain ontology holds by a server has no instances. When a client maps its own ontology to the server's one, it can enrich it with new information but this enrichment is limited to the server's domain.

3.2. Mapping creation

To illustrate the process of mapping between the ontologies of a server and a client, we use a running example that models information sharing in a university domain. The server's ontology describes knowledge about the university structure, as shown in Figure 1. We call it Ontology1 or O1. The client's ontology, which covers the same domain, is represented in Figure 2. We call it Ontology2 or O2. In their graphical representations, a concept is described by a box which displays the name and the properties of the concept. Roles are represented by a circle. We assume that both the client and server use DL to describe their ontologies. Figure 3 shows a partial specification of the two ontologies O1 and O2 in DL.

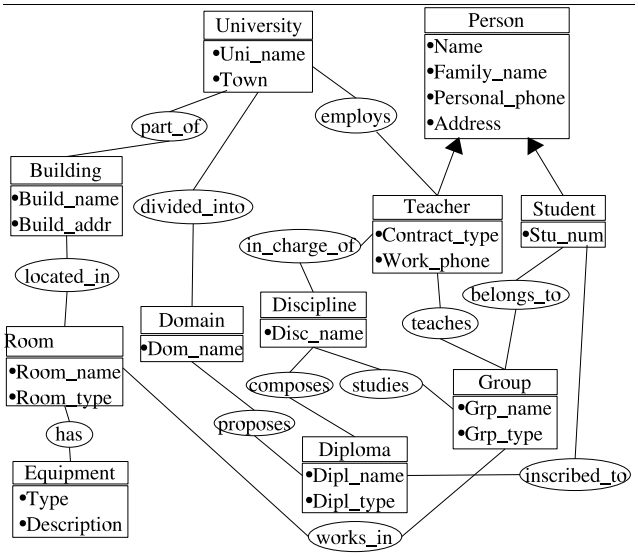

Figure 1. Domain Ontology on the Server (O1)

The mapping creation methodology consists of three phases: Preprocessing, Matching and Refining. Details about the mapping process depend on the the rules specified by the server. These rules describe the mapping methods and point to any needed external data. When the automatic process is complete, the experts may improve the computed mapping manually.

3.2.1 Preprocessing

Concept names are used to create mappings. They usually are defined in a compacted form, and the goal of the preprocessing step is to transform them into a group of recognized words. This step separates the different words concatenated

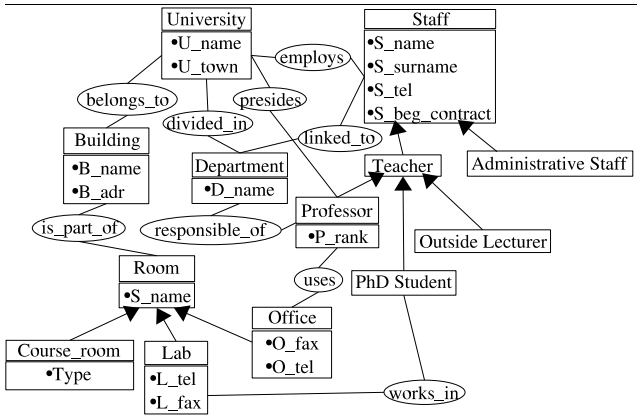

Figure 2. Local Ontology on the Client (O2)

to create a concept name, and expands acronyms and abbreviations. For example, the concept name *Family_Name* is understood as the concatenation of words *Family* and *Name*. To perform this step, an external dictionary or taxonomy is needed. In the case where no obvious translation is possible, the more realistic hypothesis in terms of word similarity are supplied.

3.2.2 Matching

The matching phase is devoted to the definition of an estimated similarity value to each pair of concepts between the two ontologies being matched. When two terms have an estimated similarity greater than a given threshold, they are likely to be synonyms. On the opposite, a low similarity measure indicates that the two terms are disjoints.

There are numerous similarity computation methods, originally coming from the linguistic field. They often rely on external data, *ie.* knowledge not available directly through the two ontologies, such as dictionaries(Lesk[7]) and taxonomies (Jiang and Conrath[6]). As the quality of the results of the different methods can vary depending on the ontologies being mapped, we allow each server to select the method which seems to give the most accurate results. Furthermore, several methods can be available in a single server. The weighted mean of the results is used as a final smilarity computation when several methods are used. The selection of the methods as well as their respective weights is decided by experts.

3.2.3 Refining

The *refining phase* is used to improve the consistency of the generated mappings. This is an iterative process which

O1.Person ≡	∃ O1.Name.O1.PersonName
	∩ ∃ O1.Family_name.O1.PersonFamilyName
	∩ ∃ O1.Personal_phone.O1.PersonTelNumber
	∩ ∃ O1.Address.O1.PersonAddress
O1.Teacher ⊆	O1.Person
	∩ ∃ O1.Contract_type.O1.TeacherContract
	∩ ∃ O1.Work_phone.O1.TeacherTelNumber
	∩ ∃ O1.teaches.O1.Group
	∩ ∃ O1.in_charge_of.O1.Discipline
O1.Student ⊆	O1.Person
	∩ ∃ O1.Stu-num.O1.StudentNumber
	∩ ∃ O1.belongs_to.O1.Group
	∩ ∃ O1.inscribed_to.O1.Diploma
O2.Staff ≡	∃ O2.S_name.O2.StaffName
	∩ ∃ O2.S_surname.O2.StaffSurname
	∩ ∃ O2.S_tel.O2.StaffTelNumber
	∩ ∃ O2.S_beg_contract.O2.StaffContractNumber
O2.Professor ⊆	O2.Teacher
	∩ ∃ O2.P_rank.O2.ProfessorGrade
	∩ ∃ O2.responsible_of.O2.Department
	∩ ∃ O2.uses.O2.Office
O2.OutsideLecturer ⊆	O2.Teacher
O2.PhDStudent ⊆	O2.Teacher
	∩ ∃ O2.works_in.O2.Lab
O2.Office ⊆	O2.Room
	∩ ∃ O2.O_fax.O2.OFaxNumber
	∩ ∃ O2.O_tel.O2.OTelNumber
O2.Lab ⊆	O2.Room
	∩ ∃ O2.L_fax.O2.LFaxNumber
	∩ ∃ O2.L_tel.O2.LTelNumber

Figure 3. Partial description of the ontologies

takes as input the results from the matching phase. The similarity values are modified using neighbourhood rules.

Rules are used to determine if the similarity value between two concepts must be increased or decreased based on how similar is the neighbourhood. These rules can be either generic or knowledge-domain specific.

For example, if the computed similarity suggests that the concepts C1 from Ontology 1 and C2 from Ontology 2 match, then it is likely that their respective father concepts in the hierarchy match also, possibly allowing us to infer a similarity. If the modification makes the fathers' similarity value high enough, the same rule can in turn be applied to the upper step of the hierarchy for the next iteration.

The rules are not necessarily linked to taxonomic relations between concepts. The neighbours of the concepts can affect the similarity results independently of the role they are linked to. Figure 4 shows an example of a non-taxonomy rule. Consider concepts *Person* from the reference ontology and *Staff* from the local ontology, their respective neighbourhoods include all concepts directly linked to them by any role. The proportion of matching terms between the two groups is compared to a threshold. If the matching is high enough, then it is likely that the concepts matc.. Their similarity is increased as a consequence.

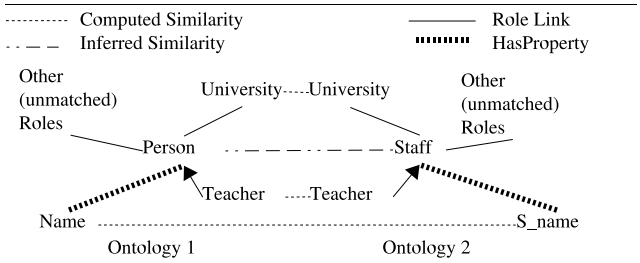

Figure 4. Neighbourhood affects concept similarity

In our system, experts are allowed to design knowledge-domain rules which represent logical statements about concept organization. They are linked to specific notions, as *Teacher* or *University* and we associate them with the concepts of our reference ontology.

For example, a specific rule of the university domain could be: If the computed similarity suggests that Concept2 matches *Teacher*, then the probability that his father Concept1 matches *Professor* which is a child concept of *Teacher*, decreases.

Each iteration takes the neighbours from each concept and modifies the similarities as required. The process is then repeated as long as needed. This process does not always reach a stable state. To avoid this problem and to reduce the execution time, the maximal number of iterations can be set to a fixed value as explained by Hameed Preece and Sleeman[2].

The following mapping is obtained through the refining step. As the concepts named *Teacher* from both ontologies match, the similarity between the fathers of concepts *Person* and *Staff* has been increased, enough to match both concepts. Experts finally transform the equivalence into a subsumption.

O2.Staff $\subseteq$ *O1.Person*

After some iterations, the concepts *Staff* and *Person* are matched. There is one rule stating that if we consider a concept whose father matches with another concept, then the considered concept is subsumed by the matched one. Thus all children of the concept *Staff* are subsumed by the concept *Person*. *O2.AdministrativeStaff* $\subseteq$ *O1.Person*

Once all these phases are executed, experts are needed to exploit the results in association with a prover. They can thus check and solve any inconsistency in the generated set of mappings, as well as describe complex mappings, which

can not be inferred automatically. Changes include numerical factors affected to solve scale differences or other kinds of data modification which can hardly be done automatically.

In the running example, the experts can modify existing mappings or create new ones. For example, the automatic phase can have returned the following result:

O1.Student $\equiv$ *O2.PhDStudent*

which can be converted into a subsumption:

O2.PhDStudent $\subseteq$ *O1.Student*

4. System description

This section provides the functional and architectural base details of the server and client components introduced in the previous section.

4.1. Client System

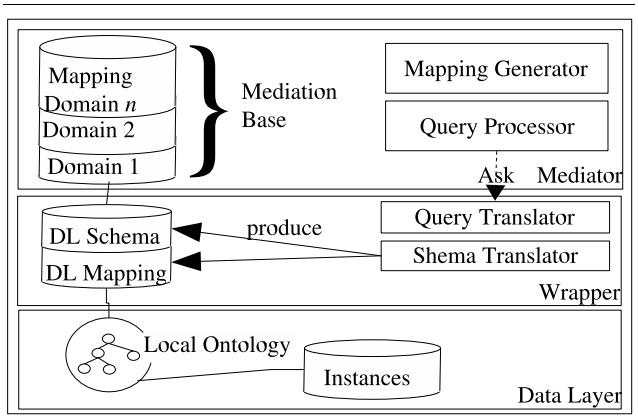

Figure 5. Details of the client

As shown in Figure 5, a client is divided into three parts: Data Layer, Wrapper and Mediator.

Data Layer is the core of the client system. It is composed of an ontology with its instances. The language of the ontology is not defined, as we assume that we can translate it to Description Logics with the wrapper.

The **Wrapper** is a translation interface between the language of the local ontology and DL.

The wrapper generates a DL representation of the local ontology, with the correspondences between the original ontology and its translation. The representation (called DL Schema) and the correspondences (called DL mapping) are then stored to avoid the translation each time a user wants to solve a query.

Queries asked directly on the client system (by opposition with queries asked on the server) must be translated by the wrapper by its Query Translator. They are then transferred to the query processor, one component of the mediator.

The **Mediator** is composed of three parts. The mapping generator uses the DL Schema, the reference ontology from the chosen server as well as its similarity computation variables to generate the mapping from the local ontology towards a given server.

Resulting mappings are stored inside the mediator. Several mappings corresponding to several server systems are stored in the mediation base, which is the second part of the mediator.

The query processor solves queries using the server system.

4.2. Server System

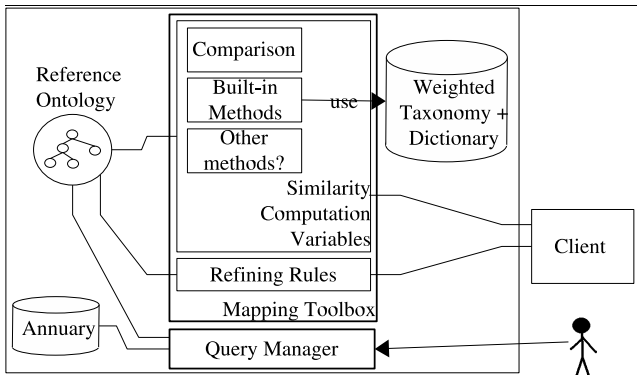

Figure 6. Details of the server

As shown in Figure 6, the server system is composed of a *Reference Ontology*, a *Mapping Toolbox*, a *Query Manager* and an *Annuary*.

The ontologies of the client systems are mapped to the server **Reference Ontology**. The reference ontology is extended when its structure is found to be incomplete. The reference ontology describes common information over a specified knowledge domain. We insist on the fact that the goal is not to match all data from clients ontologies but only the ones that are relevant domain-wise. However, nothing prevents the clients to be mapped to several server systems.

The mapping toolbox contains all elements useful to perform mappings: methods, external data and refining rules.

Methods are used as plug-ins to compute similarity between two terms. A value is associated with each method to

define its weight in the whole estimation process [11].

External data are reference items used to compute similarity. They are usually a reference taxonomy and a reference dictionary. Refining rules are a list of conditions to be checked in while processing the refining phase. There is a basic set of rules which can be extended by the experts as they wish.

The Query Manager centralizes queries and dispatches them to clients which are likely to provide relevant answers. The query manager uses the annuary to know which clients it should ask.

The Annuary contains a list of concordances between the generic ontology and each local ontology. Concordances are not a complete description of the mappings but a list of which client ontologies are linked to each concept of the reference ontology. Thus, we are able to query only clients which can bring information.

4.3. Queries

Queries can be submitted by users to a client or directly to a server. If the query is sent to a client, the query processor solves the part related to its local ontology, then it sends the results and the query to the query manager of the server. The query is translated in DL and adapted for the reference ontology with the help of the domain mappings stored in the client.

The server's query manager sends the query to any client likely to own relevant information based on the annuary content. For example, consider a query for finding all the laboratories managed by some universities. If one client's local ontology does not take into account the research activity of the university, it is irrelevant to query this client. The query manager then only asks the relevant clients. Once a query is solved by a client, the results are sent back to the server which must organize and complete them.

5. Conclusion

After a quick overview of ontologies, we presented some existing mapping tools with their strength and weakness.

We proposed a mapping method designed to be integrated in an ontology management structure, that we also presented.

Our proposed architecture uses several features. It is a scalable structure, able to cope with new methods of automatic mappings.

Restrictions on the domain of the server's ontology allow the clients to know what kind of information can be found on a given server. Moreover, it prevents the ontology on the server to become too big with new information constantly added, even if the server supports ontology improvement.

The queries are managed cooperatively by the server and the clients, notably through the use of an annuary.

The DL language used for the system allows us to use prover. This can be useful, notably in order to infer new ways to solve queries which do not match exactly with the available ontologies and data.

References

[1] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Ontology matching: A machine learning approach. In S. Staab and R. Studer [12], pages 385–404.

[2] A. Hameed, A. Preece, and D. Sleeman. Ontology reconciliation. In S. Staab and R. Studer [12], pages 231–250.

[3] C. Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, May 1998.

[4] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2003.

[5] I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 2000.

[6] J. Jiang and D. Conrath. Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy. In *International Conference Research on Computational Linguistics (ROCLING X)*, Aug. 1997.

[7] M. Lesk. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *SIGDOC '86: Proceedings of the 5th annual international conference on Systems documentation*, pages 24–26, New York, NY, USA, 1986. ACM Press.

[8] N. Cullot, C. Parent, S. Spaccapietra, and C. Vangenot. Ontologies : A contribution to the DL/DB debate. In *Proceedings of the 1st International Workshop on Semantic Web and Database (SWDB'2003) Co-located VLBD'2003*, pages 109–130, 2003.

[9] N. Noy. Tools for mapping and merging ontologies. In S. Staab and R. Studer [12], pages 365–384.

[10] N. Noy, R. Fergerson, and M. Musen. The knowledge model of Protege-2000: Combining interoperability and flexibility. In *EKAW*, 2000.

[11] N. Silva and J. Rocha. Service-Oriented Ontology Mapping System. In *Proceedings of the Workshop on Semantic Integration of the International Semantic Web Conference*, Sanibel Island (FL), USA, Oct. 2003.

[12] S. Staab and R. Studer, editors. *Handbook on Ontologies*. International Handbooks on Information Systems. Springer, 2004.

[13] T. Berners-Lee, J. Hendler, and O. Lassila. The Semantic Web. *Scientific American*, May 2001.

[14] V. Haarslev and R. Möller. Racer system description. In *Proc. of International Joint Conference on Automated Reasoning, IJCAR 2001*, pages 701–705. Springer-Verlag, 2001.