Complement Concept and Capability Discovery

Nacer Boudjlida and Cheng Dong

UHP Nancy 1, LORIA, Campus scientifique, BP 239
54506 vandœuvre Ls Nancy CEDEX (F) {nacer or cheng}@loria.fr,

Abstract. This paper concerns the advertisement and the discovery of the capabilities or the know-how of an "object" (like a business partner, an employee, a software component, a Web site, etc.). One of the originality of our proposal is in the nature of the answers the intended system can return: they are not Yes/No answers but when no single object meets the search criteria, we attempt to find out what a set of "complementary" objects that do satisfy the whole search criteria, every object in the resulting set satisfying part of the criteria. Description Logics (DL) is used as a knowledge representation formalism and the determination of the "complementary objects" is founded on the DL *complement* concept.

1 Motivation and Architecure

Component-based programming, electronic business (*e-business*) and even enterprise knowledge management [18] are among the application domains in which there is a need for the discovery of services or capabilities an "entity" offers. The only syntactic description of an entity's capability (like the *signature* of software component's services, for example) is far from being satisfactory when trying to match service offers and a request: additional semantic description is required. Moreover, the elicitation of possible relationships among services may contribute to find out "the best" service or the "the best complementary" services that satisfy a search query. As a matter of comparison, in [13], "entities" are design fragments that are described thanks to keywords, the relationships between the fragments are constituted by metrics that measure the similarity between fragments. In [3], entities are object-oriented software components and description logics is used, notably, to describe their intended semantics together with possible constraints involving objects methods. In [4], "entities" are software objects and the capabilities of a software object are specified giving a syntactic part (signature of the object's methods) and a semantic part expressed as logical expressions (method's pre-condition and a post-condition). The syntactic conformance of a method to a query uses sub-typing relationship identification [6, 5] and the semantic conformance uses theorem proving techniques. In [16], a DAML-S based ontology and a DL reasoner are used to support Web services advertisement and discovery.

The purpose of this work requires formalizing and structuring the knowledge that concern the "entities" in a given application domain. We opted for Description Logics (DL) [11, 15] as a knowledge representation formalism. DL has

been used in various domains. In the database one [2][1], DL was used not only for querying but also for database schema design and integration [8], for reasoning about queries (query containment, query refinement, query optimisation, ...) [1]. From our concern, description logics is used for query purposes with the special objective to produce more than "Yes or No" results.

From the system architecture point of view, we opted for a trader (also called mediator) based architecture very similar to the notion of *discovery agency* in the Web services architecture [20] and conform to RM-ODP (*Reference Model for Open Distributed Processing*): an "entity", called *exporter*, publishes (*tells*) its capabilities at one or more mediators sites (see figure 1). Entities, called *importers*, send requests (*queries*) to the mediator asking for exporters fitted with a given set of capabilities.

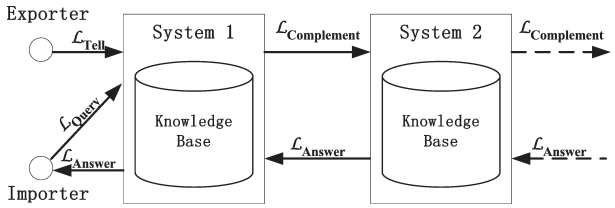

Fig. 1. The Mediator-based Architecture

The mediator explores its knowledge base to try to satisfy the query. The search process is founded on the exported capabilities and on relationships between them, these relationships being transparently established by the mediator. When some exporters known from the mediator satisfy the query, the references of those exporters are sent back to the importer in $\mathcal{L}_{answer}$. Nevertheless, satisfying the query falls into different cases [7] as graphically summarized on figure 2 (the figured cases correspond to those respectively called *Exact, Subsume, PlugIn, Intersection* and *Disjoint* in [16]).

One should notice that cases 4 and 5 would conduct to a failure of the query when only one mediator is implied. But, if we assume a grouping of mediators (into a federation of mediators), these cases are typical cases where cooperation among the mediators is required. In the case 5, the whole query is transmitted for evaluation to other mediators whereas in the case 4, we need to determine "what is missing" to the individuals to satisfy Q, that means to determine what part of the query is not satisfied by the found individuals. This part as well as the original query are transmitted then to a mediator of the federation. Conceptually, we can see the query as being addressed to "the union" of by the federated mediators' knowledge bases. Concretely, this union is explored from "near to near" within the federation, that means from a mediator to an other one. The

[1] See also [11] for the proceedings of the various "Knowledge Representation meets DataBase" (KRDB) Workshops.

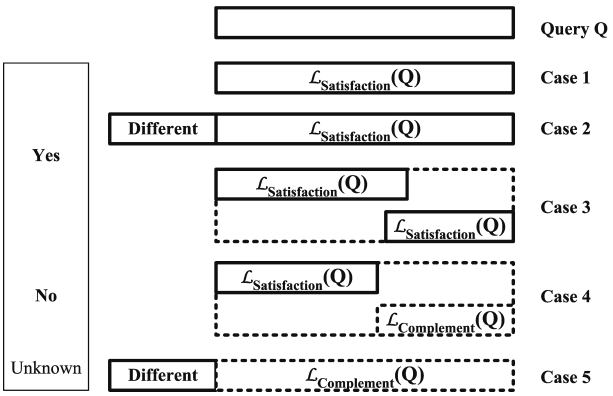

Fig. 2. Matching and mismatching cases

ultimate goal of this work is the design and the development of a set of services to export, import and mediate, as well as the study and the development of alternative strategies and mechanisms for mediators cooperation. The coming sections detail the adopted approach.

2 Complement and composite answer

The query language is defined by $\mathcal{L}_{Query} \doteq \{$ DLs formulaes$\}$, and $\mathcal{L}_{Answer}$ is defined by two components the $\mathcal{L}_{Satisfaction}$ and $\mathcal{L}_{Complement}$. $\mathcal{L}_{Satisfaction}$ describes a satisfaction, that is a single entity or a set of entities satisfying a query Q. If Q is completely satisfied, its complement (noted $\mathcal{L}_{Complement}(Q)$) will be empty. In the contrary, the system will try to determine a complement for this answer. We define a function $Comp(-,-)$ that calculates the complement of an answer to a query: $\mathcal{L}_{Complement}(Q) = Comp(\mathcal{L}_{Satisfaction}(Q), Q)$. Intuitively this complement designates "the missing part" to an entity in order for that entity to satisfy the query.

The determination of the unsatisfied part of a query is founded on the concept of *complement* [19] in DLs and the test of the *subsumption* relationship between the concepts of the query and those in the mediator's base[2]. The algorithm that we propose to test if this relation exists or not, presents the originality, in the case where the relation is not true, to identify the concepts of the query that are the reason of the non existence of the relation: these concepts describe "the part that is missing" to the individuals who partly satisfy the query, i.e. $Comp(A, B) = C$ is the minimal additional knowledge C that is missing to an instance of B to be an instance of A.

The determination of the complement is based on the subsumption relationship using a *Normalize-Compare* process. The aim of the normalization is

[2] Hereafter, the presentation is deliberately unformal. For introductions to DL, refer to [11, 15, 17] and for a more formal presentation of this work refer to [7] and [10].

to put defined concepts to be compared, let say A and B, under a conjunctive form: A = (and $A_1, A_2, A_3, \ldots, A_n$) and B = (and $B_1, B_2, B_3, \ldots, B_m$). Once normalized, two concepts can be easily compared to check wether the subsumption relationship holds between pairs of them or not: giving A = (and $A_1, A_2, A_3, \ldots, A_n$) and B = (and $B_1, B_2, B_3, \ldots, B_m$), the test "does the concept A subsume the concept B?" returns "true", if and only if $\forall A_i$ (i $\in$ 1,...,n) $\exists\ B_j$ (j $\in$ 1,...,m) such that $B_j \sqsubseteq A_i$. The implementation of this process uses an array of Boolean (called "Table_Of_Test" further) to record the results of the subsumption relationship evaluation. In the figure 3, $C_1, C_2, C_3, \cdots, C_n$ denote the query concept under its normal form and $D_1, D_2, D_3, \ldots, D_m$ denotes the concepts "known from" the mediators, i.e. every D_j has to be viewed under its normal form $D_j^1, D_j^2, \cdots, D_j^{nj}$. Then $Table\_Of\_Test[D_j, C_i] = true$ means that $D_j^i \sqsubseteq C_i$. When the value returned by the function $Subsumes(C, D_j)$ is "false" (i.e. the concept D_j does not fully satisfy the concept C.), therefore we need to determine a possible complement of D_j relatively to C. Using the Table_Of_Test it is easy to get the complement of the concept D_j relatively to the concept C: $Comp(C, D_j) = and_{k=1}^{r} C_k | TableOfTest[D_j, C_k] = false$[3]. That means that the complement is given by the conjunction of all the atomic concepts for which the corresponding values in the "Table Of Test" are "false".

	C_1	C_2	...	C_n
D_1	False	False	...	True
D_2	False	True	...	True
...	...	...	...	...
D_m	False	False	...	False
ORoD	False	True	...	True

ORoS	ANDoS
True	False

Fig. 3. The "Table Of Test"

The composition of the truth values determines the cases of satisfaction. Consider a table $ORoD[1..n]$ as $ORoD[i] = \bigvee_{j=1}^{m} T[D_j, C_i]$. $ORoD[i] = true$ means that the concept C_i is satisfied by at least a D_k. If the conjunction of the values of $ORoD$, noted $ANDoS$, is true (i.e. $\bigwedge_{i=1}^{n} ORoD[i] = True$), it means that all the C_is are satisfied and therefore the query. When $ANDoS$ is false, the logical disjunction of the values of $ORoD$, noted $ORoS$, enables to determine a possible partial satisfaction: if $ORoS = \bigvee_{i=1}^{n} ORoD[i] = True$, it means that there exist some C_k that are satisfied. If both $ORoS$ and $ANDoS$ are false then no atomic concept D_j^k ($j \in 1..m$) satisfies a C_i. The figure 4 summarizes this discussion (in this figure, X, $\top$, and $\bot$ respectively denote a concept, the TOP concept and the BOTTOM concept; *MSSC* and *MGSC* respectively denote the Most Specific Subsuming Concept and the Most General Subsumed Concept;

[3] In the following, we will use T[i, j] instead of TableOfTest[i, j]

finally the numbers in the CASE column refers to the satisfaction cases listed in the figure 2).

MSSC	MGSC	ORoS	ANDoS	CASE
X	X	True	True	1
X	$\bot$	True	True	2
$\top$	$\bot$	True	True	3
$\top$	$\bot$	True	False	4
$\top$	$\bot$	False	False	5

Fig. 4. The analysys of the satisfaction case

An experimental platform has been developed in *Java* where services, like testing the subsomption relationship, determining the complement of a concept and computing a composite answer, have been implemented. These services are accessed through a *Web Server* in a "classical" Web client/server architecture. The services accept DL concepts written in DAML+OIL [14], an ontology language in *XML*. The DL concepts are encoded in *XML* and transmitted to the *Web Server* who in turn transmits them to the appropriate mediator service. Then a normalization class transforms them into *Java* objects and an experimental knowledge base, described in XML, is loaded and normalized into a *TBox* object when the service is started. All the algorithms of the mediator's services are implemented in the *TBox* class. The services' outputs are also encoded in *XML*. XML is also used to exchange information and concepts between the mediators when mediators' copperation is needed. In the current status of the implementation, a mediator "discovers" an other mediator using a static mediator address list. More complex and dynamic discovery techniques will be supported in the coming versions. Moreover, we deliberately ignored the search of the actual *individuals* (*ABox*) that satisfy a query, i.e. in the current implementation, only *TBoxes* are considered.

3 Conclusion

In this paper, we presented a method and an algorithm for testing the subsumption relationship, determining the complement concept and finding composite answers. One of the originality of this work is in the type of query answering we provide and also in the way we used and implemented the complement concept. Further work is considering the complexity of the algorithm, and heterogeneous mediators cooperation, i.e. mediators where knowledge bases are described in different languages (in the spirit of the work described in [12] and [9]).

References

1. Beeri, C., Levy, A., and Rousset, M.-C. (1997). Rewriting Queries Using Views in Description Logics. In *ACM PODS*, 99–108, Tucson, Arizona.
2. Borgida, A. (1995). Description Logics in Data Management. *IEEE Trans. on Knowledge and Data Eng.*, 7(5):671–682.
3. Borgida, A. and Devanhu, P. (1999). Adding more "DL" to IDL: Towards more Knowledgeable Component Interoperability. In *21rst Intern'l Conf. on Soft. Eng., ICSE'99*, 378–387, Los Angeles, CA, ACM Press.
4. Bouchikhi, M. and Boudjlida, N. (1998). Using Larch to Specify the Behavior of Objects in Open Distributed Environments. In *Proceedings of the 1998 Maghrebian Conf. on Soft. Eng. and AI*, 275–287, Tunis, Tunisia.
5. Boudjlida, N. and Perrin, O. (1994). A Formal Framework and a Procedural Approach for Data Integration. In *Proceedings of the Intern'l Conf. on Systems Integration*, ICSI'94, 476-485, IEEE Comp. Society Press, São Paulo, Brazil, August.
6. Boudjlida, N. (1995). Knowledge in Interoperable and Evolutionary Systems. In Dreschler-Fischer, L. and Pribbenow, S., editors, *KRDB'95, Workshop on "Reasoning about Structured Objets: Knowledge Representation Meets Databases"*, 25–26, Bielefeld, Germany.
7. Boudjlida, N. (2002). A Mediator-Based Architecture for Capability Management. In Hamza, M., editor, *Proceedings of the 6th Intern'l Conf. on Soft. Eng. and Applications, SEA 2002*, 45–50, MIT, MA.
8. Calvanese, D., de Giacomo, D., Lenzerini, M., Nardi, D., and Rosati, R. (1998). Information Integration: Coceptual Modeling and Reasoning Support. In *6th Intern'l Conf. on Cooperative Information Systems, CoopIS'98*, 280–291.
9. Calvanese, D., de Giacomo, D., and Lenzerini, M. (2001). A Framework for Ontology Integration. In *Proceedings of the Intern'l Semantic Web Working Symp., SWWS 2001*, 303-316.
10. Cheng, D. and Boudjlida, N. (2004). Federated Mediators for Query Composite-Answers. *Proceedings of the 6th Intern'l Conf. on Enterprise Information Systems, ICEIS'04*, Vol.4, pages 170–175, Porto, Portugal, April 14-17, ISBN 972-8865-00-7.
11. DL-org (2003). Description logics. http//dl.kr.org/.
12. Halevy, A. Y. (2001) Answering Queries Using Views: a Survey, VLDB Journal, 10(4):270-294.
13. Han, T.-D., Purao, S., and Storey, V. (1999). A Methodology for Building a Repository of Object-Oriented Design Fragments. In *18th Intern'l Conf. on Conceptual Modelling, ER'99*, 203–217, Paris, LNCS 1728.
14. Horrocks, I. (2002a). DAML+OIL: a description logic for the semantic web. *IEEE Data Engineering Bulletin*, 25(1):4–9.
15. Horrocks, I. (2002b). Description Logic: Axioms and Rules. Talk given at Dagstuhl "Rule Markup Technique" Workshop. http://www.cs.man.ac.uk/ horrocks/Slides/.
16. Li, L. and Horrocks, I. (2003). A Software Framework for Matchmaking Based on Semantic Web Technology. *WWW2003*, May 20-24, Budapest, Hungary, ACM.
17. Napoli, A. (1997). Une introduction aux logiques de description. Technical Report RR No 3314, INRIA-LORIA, Nancy.
18. Nonaka, I. and Takeuchi, H. (1995). *The Knowledge Creating Company; How Japanese Companies Create the Dynamics of Innovation.* Oxford University Press.
19. Schmidt-Schauss, M. and Smolka, G. (1991). Attribute Concepts Description with Complements. *Artificial Intelligence Journal*, 48(1):1–26.
20. uddi.org. UDDI: Universal Description, Discovery and Integration. (http://uddi.org).