Enabling Rich Discovery of Web Services by Projecting Weak Semantics from Structural Specifications

Leo Obrst, Dru McCandless, Michael Bankston

The MITRE Corporation
{lobrst, mccandless, mbankston}@mitre.org

Abstract. Although we would prefer using defined ontologies that express the domains and specifications of web services, and thus more easily discover and compose these, we know that in the mainstream world represented by the US Department of Defense we will not have those ontologies available soon. In the meantime we have to ensure a transition from structural to semantic methods, including web service discovery methods. In this paper, we are proposing a different approach for dynamic web service discovery that takes advantage of the structure inherent in web services that are defined by WSDL documents. Since the structure is usually based on XML Schema, there is enough information present in these documents to develop a broadly applicable approach. Furthermore, if a consistent and detailed naming convention of schema artifacts is followed, then discovery can be made more precise. This paper describes our approach for projecting weak semantics from structural information for discovery of web services.

1 Introduction[*]

The use of web services has grown steadily over the past few years due to their ease of use and modularity for providing information in a standard way (including information retrieval, such as in a digital library). Web services have demonstrated their value for solving information needs that are part of regular, foreseen tasks. Thanks to standards such as the Web Service Description Language (WSDL) [8] and the Business Process Execution Language (BPEL) [9], and approaches such as Service Oriented Architecture (SOA), the way web services are defined and presented has also become more universal. However, many information integration tasks are unforeseen at the time the services are constructed, and are therefore difficult to perform "on the fly". Users lack the tools to search for what they want (i.e., the services that provide the specific information they

[*] Approved for Public Release: 10-3699. Distribution Unlimited. ©2010 The MITRE Corporation. All rights reserved.

desire) and the tools to quickly and effectively examine a potential web service to determine if it meets their needs. The ability to quickly discover and chain web services to accomplish some dynamic information need is still a ways off, and will require additional technology before it is fully realized.

Discovery in this context is the ability to locate and understand a web service that is defined by the WSDL standard. This is usually a preliminary step before the service is accessed and used. Web service discovery is a long recognized problem, and several approaches have developed as attempts to solve it. The current standard for web service discovery is Universal Description, Discovery, and Integration (UDDI) [1], which uses tModels to describe a service's content. Although useful for locating services of which the searcher is already aware, it is decidedly less useful for discovering previously unknown services.

To alleviate this, much work in the area of Semantic Web services has been done. If a searcher were able to discover web services using the same tools and techniques for performing semantic queries, then better results could be obtained. Several efforts are showing promise in this area, such as FUSION [2, 3], WSMX [4], Service-Finder [5], and EASY [6-7]. However, the premise of many Semantic Web approaches is that the source information is first organized into an underlying ontology, and then the service definitions are derived from that ontology. Unfortunately, if the organization creating the web service doesn't use this approach (i.e., have an underlying ontology) then the search and discovery benefits cannot be realized. In the current technology environment (particularly within the US Department of Defense) while web services abound, the use of ontologies is being only slowly adopted, so relying on their use for service discovery won't work. The reasons for this are varied: lack of requirements, mismatch in skill sets between ontology development and writing web services, lack of leadership support, etc. Furthermore, since service discovery is by definition an act of finding resources not under the control of the searcher, there is no way (short of re-factoring each discovered service) to force or otherwise cause other service providers to use an ontology (or to adhere to a specific ontology even if they do). The question then becomes: is there some way to realize the benefits of Semantic Web technology for rich discovery of services without having the services based on a formal ontology? Fortunately for a certain class of services there is a way to do this.

We are proposing a different approach for dynamic web service discovery that takes advantage of the structure inherent in web services that are defined by WSDL documents. Since the structure is usually based on XML Schema, there is enough information present in these documents to develop a broadly applicable approach. Furthermore, if a consistent and detailed naming convention of schema artifacts is followed (and recommendations for such are presented in this paper) then discovery can be made more precise. Such an approach can be viewed as a path to richer semantics.

The format of this paper is the following. In section 2, we describe the general approach of our discovery mechanism, and provide a view of the user interface. In section 3, we discuss the application of our approach in the larger picture of an architecture and implementation that combines many service layers and research pieces. In section 4, we provide our initial evaluation results. Finally, in section 5, we briefly discuss some issues and look toward the next steps.

2 Approach

In previous work, we have demonstrated the effective use of ontologies as a means of performing data integration [10-13] and service chaining [14, 15].

This work extends those previous efforts into the area of web service discovery and assembly. Our approach is as follows:

1. XML Schema definitions of web service input and output items are extracted from a body of WSDL documents.
2. A registry is created that contains the XML Schema definitions, message structure, and the operations of each WSDL file, along with a small amount of metadata.
3. The schema elements are efficiently encoded as a graph set to enable fast lookup as part of the search.

Service discovery is intended to allow subject matter experts to locate information services that provide information needed for some unforeseen task. The assumption is that searchers will be knowledgeable about the domain and somewhat knowledgeable about services, although they may rely on software tools to perform the actual integration (a method for which will be described later in this paper). It is not expected that they will be experts in logic, ontologies, or even search technology.

The mechanics of searching breaks the search terms into three separate parts. The terms are hierarchical. The top level is the overall topic of the search and is usually broad in scope (e.g., sports, medicine, airplanes, etc.). The topic is usually assigned as metadata when the service is registered. Ideally the topic is selected from a pre-set list so that services covering the same subject will use the same topic description, allowing a searcher to use something such as a pull-down list to pick the topic (and thus simplifying the search task). The advantage of this approach is the pre-filtering or ranking of services for locating the type of service being sought (as well as help in disambiguating terms). The disadvantage is the extra bookkeeping that must be done by the registry owner to select the topics and keep them consistent.

The next level is the subject: that is, a term (or terms) for the actual thing being sought. This is a free-text field, and is matched against the text labels of the XML elements of the messages that are in the registry. Various match techniques are used to catch partial or imperfect matches (CamelCase parsing, partial term matching, term expansion using a thesaurus or semantic data model). This approach is sound insofar as the thing being sought is an actual or measurable thing, and not an abstract thing.

The bottom level consists of terms describing the properties or attributes of the subject. These are also matched to the names of elements and attributes of the schemas that make up the messages within the WSDL files.

The three levels of terms can be thought to constitute a very primitive ontology, and the process of searching for matching WSDL messages is similar to a graph-based ontology matching attempt. In fact, the XML structure of the messages in our approach are encoded using a graph-encoding technique similar to the one described in Ait-Kaci [16].

The heart of the search method is in matching the subject search terms to WSDL operations. The other two matching operations (topic and properties) essentially modify the scores of the items returned by the subject search. The assumption is that a searcher will provide a subject, but may not always provide a topic or properties. When these are not provided, the score remains unchanged.

2. 1 Subject Search Method

The following describes our overall subject search method. We index these for readability.

(1) The search software is designed to support the use of multiple multi-word search terms, where each term is separated by a comma (e.g., "satellite status, owner"). The first step is to take the search phrase and turn it into an array of search terms, where multi-word terms are also combined into a single word. So the subject search "satellite status, owner" becomes the string array ["satellite", "status", "satellitestatus", "owner"]. The reason for creating the combined word is that XML schema designers often use "camel case" to name nodes (i.e., elements) in the XML message structure. The idea is that if a combined term happens to be found in a message node name, then that increases the likelihood that it constitutes a good match, and that node's corresponding operation receives a higher score.

(2) The terms in the string array formed in step (1) are expanded. Each search term (including multi-word terms) are checked against a known library of synonyms, abbreviations, and acronyms. Thus for example "frequency" is expanded to include "freq", "point of contact" now includes "poc", and "space object" is expanded to include "satellite". The expanded array terms are also pluralized. This forms the final array of search terms.

(3) The following scoring loop is then executed. For each search term in the array formed in step (2):

(3.a) The term is looked up in the service registry index, and if found then its corresponding list of operation nodes is returned. The operation nodes are stored as a dot-separated set: WSDLName.OperationName.NodeName

(3.b) For each operation node in (a):

i. The operation node is split into the WSDLName.OperationName (a 'key') and the NodeName (the value)

ii. If the key doesn't already appear in the set of answers, then it is added with a beginning score of zero.
iii. The NodeName is split into its component CamelCase parts.
iv. The score is then computed based on the number of terms the NodeName is split into, where the score is 1/number of terms unless the matching term is the last one of the camel case terms; in which case the score is 0.8. For example, the NodeName SatellitePayloadStatus is split into 3 terms, so the score for that operation would be 1/3 if the search term were 'satellite' or 'payload', and .8 if it were 'status'. Note that if a NodeName consists of a single term, then the score is 1 (the highest possible) if the search term matches it. The last word in a camel case term is considered more significant since it tends to be the most significant (i.e., more "noun-like"), whereas preceding words are more adjectival. This score is a called the "occurrence score" – the score for that occurrence of a particular operation. Also note that the same service operation can be encountered multiple times as steps i – iv are repeated.

(3.c) For each operation, the occurrence scores are added up to compute a "term score" for that operation, in the following manner:

$$Term\ Score = \sqrt{\#\ occurences} + Wt\ \times Avg\ Occurrence\ Score - \log(1 + Occurrence\ Descent\ Level) \quad (1)$$

where:

# occurrences is the total number of times that operation occurs,
Avg Occurrence Score is the average of all the occurrence scores (i.e., $\frac{\sum occurrence\ scores}{\#\ occurrences}$),
Wt is a weighting factor,
Occurrence Descent Level is the integer count of how far down the highest occurring NodeName is from the root parent node of that operation.

This scoring approach seeks to preserve a balance between how many times a search term appears throughout the named labels of a service operation, how significant that term appears to be, on average, in those labels, and how important the nodes are by computing how far down the message hierarchy they are.

(3.d) The term scores are then added up for each search term. The result is a subject score for each operation.

2.2 Property Score Method

The property scores are computed in a similar manner as the scores for the subject. The main differences are that (1) the property score for each operation is scaled by the number of properties present in that operation (e.g., if a search contained 3 properties, and an operation contained 2 of them, then the property score for that operation would be multiplied by 2/3), and (2) only operations that are returned as part of the subject search are considered – so operations that match at least one property but not the subject terms are thrown out. For each operation the property score is then added to the subject score.

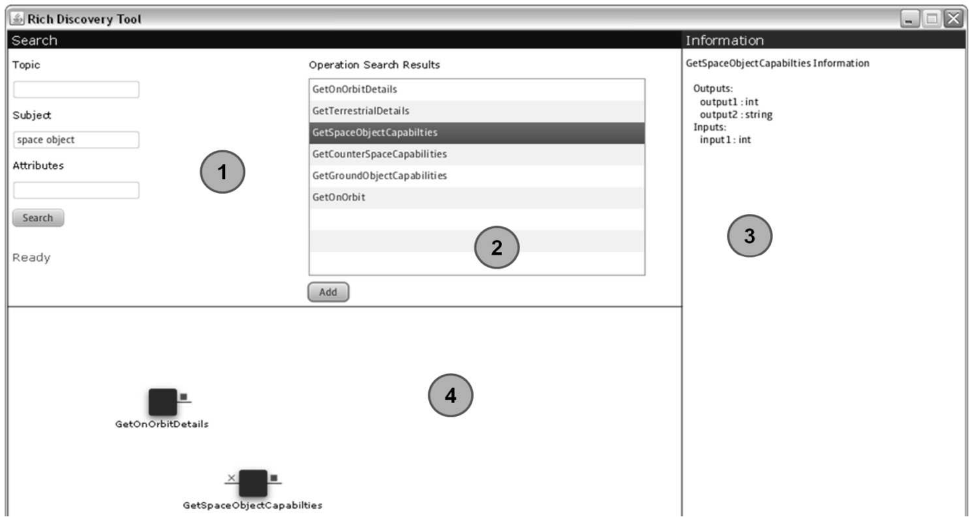
2.3 Topic Score

The topic of a WSDL is kept as a metadata item in the service registry. The way this item is included in the search is via a pulldown list which contains all of the topics in the registry. This removes the need for the searcher to try and guess what the topic is. Operations matching the topic keep their scores; those with a different topic have their scores reduced, so that they are kept but with a lower ranking.

One of the main reasons for including the topic is to provide some contextual filtering of the other terms. For example, a search for "tank" and "weight" with a topic of "armored vehicles" will score a service operation about Army vehicle features higher than a service about fuel tank capacity under the topic of "Logistics".

2.4 User Interface

Figure 1 depicts our user interface, which is still evolving. It displays four kinds of information: 1) In the upper left of the figure, users can type in *Topic*, *Subject, Attributes* to search for and discover Web services of prospective interest to them; in the example, the user has entered "space object" in the *Subject* field. 2) In the upper middle of the figure, the search returns the service operations. We focus on the highlighted "GetSpaceObjectCapabilities".



Subject: space object

GetOnOrbitDetails
GetTerrestrialDetails
GetSpaceObjectCapabilties
GetCounterSpaceCapabilities
GetGroundObjectCapabilities
GetOnOrbit

GetSpaceObjectCapabilties Information

Outputs:
output1 : int
output2 : string
Inputs:
input1 : int

Fig. 1. Service Discovery Graphical User Interface.

3) In the upper right of the figure, details of the service operations are available: in this case, the inputs and outputs of "GetSpaceObjectCapabilities", which includes their data types. 4) Finally, in the lower left of the figure, operations are displayed as boxes that can be connected to form a chain of services via drag-and-drop.

3 Application

We are integrating the service discovery mechanism into a larger effort at MITRE focused on Command and Control (C2), called Composable Capabilities on Demand (CCOD). CCOD intends to provide the capability to rapidly customize virtual systems based on the mission and threat of the day, promoting local innovation by leveraging the use of layered architectures and global integration based on loose couplers at all layers of the architecture. CCOD consists of over 20 individual research projects all providing some capabilities. There are a number of mission scenarios under CCOD. We are involved in a vignette supporting the US Department of Defense Combatant Command (COCOM) information-sharing and interaction to supply assistance to a population experiencing a natural disaster such as the Haiti earthquake of January, 2010. However, for our initial research, because the existing WSDL files that were available were C2-based, we used those, intending to work out a general method for service discovery that could be used for logistical and aid-support operations, when web services for those were more numerous.

4 Evaluation

Currently our evaluation is focused on subjective measures. Typical objective measures, such as recall and precision, slightly modified to incorporate partial or whole aspects of our approach, remain incomplete, requiring a larger population of web services than we have acquired so far.

At a recent CCOD Integration Event in June, 2010, a combined activity for all the research projects involved in the CCOD initiative, we had five people use our tool to discover appropriate services based on several search questions they were given. The questions were to find services that provided:

- The length of a runway
- The callsign and fuel code of a tanker aircraft
- An aircraft's fuel capacity
- The departure base of a tanker aircraft
- An aircraft's tail number and home base
- The payload status and owner of a space object

This gave us 5 people times 6 questions or 30 total test cases. However, some of the search questions turned out to be ambiguous or had multiple correct answers, so we had to throw a few of them out. After this reduction we ended up having a total of 21 person/questions. We compared our approach to that of a generic UDDI type interface, and the results were:

Our approach: 20/21 successful searches
UDDI: 10/21 successful searches

The testers all were very complimentary of our system's search tool and its ability to present results in an understandable manner. A couple, though, expressed some skepticism about web service search and discovery in general.

Concerning our approach in terms of estimating recall, we found this to be really subjective, i.e., deciding which terms to test for. In Table 1 below are displayed the 31 terms which represent the recall terms, and two numbers: how many WSDL files were returned and how many WSDL files should have been returned (as hand determined by us, knowing the domain). The overall recall performance was 162 of 189 returned WSDLs or 85.7%, which is respectable but not great. One issue that caused us problems is the fact that services about aircraft were not returned when the search terms "airplane", "fighter" and "bomber" were used. Although they were clearly implied (and therefore should have been found) in the service, the necessary term matches were not present in the synonym library. Precision measured 162 / 165 or 98%. Of potential significance is our ability to raise recall without reducing precision, which we attribute to using at least weak semantics and the graph structuring.

Table 1. Recall Term Counts (actual vs. expected)

Recall Terms	
aircraft: 12/13	position: 9/10
platform: 2/4	location: 16/16
satellite: 3/3	route: 1/2
mission: 16/16	vehicle: 3/3
airplane: 0/2	airspace: 4/6
airframe: 1/4	target: 6/6
track: 4/4	tanker: 2/4
runway: 4/4	fighter: 0/2
callsign: 9/10	bomber: 0/2
status: 18/18	airfield: 3/4
tail number: 2/3	plan: 5/6
payload: 3/3	ato: 7/7
home base: 1/4	equipment: 7/7
fuel: 8/8	jammer: 3/3
fuel capacity: 3/4	emitter: 1/4
weapon: 7/7	
Recall: 162 / 189 (85.7%) Precision: 162/165 (98%)	

5 Discussion and Future Research

Although our approach described in this paper uses only weak term semantics projected from the actual WSDL structures, when combined with heuristics used by the algorithm (different weights for topic, subject, property, and use of the hierarchic structure of the WSDL), and a simple lexicon of synonyms and acronyms to assist in query expansion, it promises to assist technically unsophisticated users at discovering relevant Web services, and then enabling their drag-and-drop composition – all without the use of a sophisticated domain ontology. We do believe that more sophisticated terminologies and domain-specific ontologies will enable more accurate search and discovery methods, and will draw upon these as they become available. For example, some early Community of Interest (COI) vocabularies are emerging within the DoD for command and control, logistics, etc., and some with associated ontologies to represent the term meanings. But in general, these are not yet emerged. Also, some external lexical-conceptual resources such as WordNet can be employed and integrated into our approach, but in general, we have found these to not have the term specificity and complexity of domain knowledge that we need. Hence, we view our technical contribution as enabling a transition strategy between purely syntactic (keyword-based) methods and much richer semantic methods using ontologies, to assist users and developers over the near and mid-term. In addition, we believe our method advances the state of the art and can be expanded on when richer resources become available.

Concerning structural issues, WSDL and XML files (and WSDL's syntax is XML-based) are tree-based, since XML's underlying data model is tree-based. So in our design and implementation of an efficient representation for the structure of WSDL services, we focused on a tree-based representation, though we did plan for a graph-based generalization, and our encoding scheme reflects that. We considered various graph-based representations and efficient encodings of subsumption reasoning based on those representations. These include considerations from early bit encodings such as [16-21] to more recent work that includes prime number encodings, such as [22-23].

We are also very interested in addressing more objective evaluative measures such as typically captured by recall and precision metrics (sufficiently adjusted to fit the circumstances). However, our total set of WSDL services is really not large enough yet to apply these measures meaningfully. The set of structure-based WSDL-defined Web services will undoubtedly grow in the near term, given that ontologies that define more precise semantics for domains of those services unfortunately will continue to be lacking. So we think therefore that we will have more service definitions to work with in the future, and hence be able to evaluate our discovery methods more precisely against that larger set.

Finally, we have identified several recommendations and conventions for service schema developers to adopt, to increase the value of the syntactic and structural services they define when using a light semantic approach for discovery such as ours, but which will also benefit approaches with richer semantics: 1) use descriptive names for elements and attributes (FuelStationLocation) and avoid general terms ("items", "response"); use whole words and avoid contractions ("track" not "trk"); follow a consistent style for CamelCase; and design schemas with as much specificity as possible (e.g., avoid xs:any, xs:all). These practices, when combined with a service-discovery approach such as ours, can help find and reuse web services – until richer ontologies and lexical resources are available, and until developers become more sophisticated with semantic technologies.

Acknowledgments. The views expressed in this paper are those of the authors alone and do not reflect the official policy or position of The MITRE Corporation or any other company or individual.

References

1. Baumgartner, R., Flesca, S., Gottlob, G.: Visual Web Information Extraction. VLDB Conference, 2001

2. Kourtesis D., Paraskakis I.: Combining SAWSDL, OWL-DL and UDDI for Semantically Enhanced Web Service Discovery. In Bechhofer S. et al.(Eds.): ESWC 2008, Lecture Notes in Computer Science 5021, Springer-Verlag Berlin Heidelberg, pp. 614-628 (2008)
3. Kourtesis, D., Paraskakis, I., Friesen, A., Gouvas, P., Bouras, A.: Web Service Discovery in a Semantically Extended UDDI Registry: the Case of FUSION. In: Camarinha-Matos, L., Afsarmanesh, H., Novais, P., Analide, C. (Eds.) IFIP International Federation for Information Processing, Establishing the Foundation of Collaborative Networks, vol. 243, Springer, Boston, pp. 547-554 (2007)
4. Web Service Execution Environment, http://www.wsmx.org/
5. Service-Finder, http://www.service-finder.eu/
6. Mokhtar, S. B., Kaul, A., Georgantas, N., Issarny, V.: Efficient Semantic Service Discovery in Pervasive Computing Environments. Proceedings of the ACM/IFIP/USENIX 2006 International Conference on Middleware, IFIP International Federation for Information Processing, Melbourne, Australia. M. van Steen and M. Henning (Eds.): Middleware 2006, Lecture Notes in Computer Science 4290, pp. 240–259 (2006)
7. Mokhtar, S. B., Preuveneers, D., Georgantas, N., Issarny, V., Berbers, Y.: EASY: Efficient semAntic Service discoverY in pervasive computing environments with QoS and context support. The Journal of Systems and Software 81, pp. 785–808 (2008)
8. Web Services Description Language (WSDL), http://www.w3.org/TR/wsdl
9. Business Process Execution Language (BPEL), http://en.wikipedia.org/wiki/Business_Process_Execution_Language
10. Obrst, L., Stoutenburg, S., McCandless, D., Nichols, D., Franklin, P., Prausa, M., Sward, R.: Ontologies for Rapid Integration of Heterogeneous Data for Command, Control, & Intelligence. Chapter in: Obrst, L., Janssen, T., Ceusters, W. (eds.) Ontologies and Semantic Technologies for the Intelligence Community, pp. 71-89. IOS Press, Amsterdam, The Netherlands. IOS Press book series: Volume 213 Frontiers in Artificial Intelligence and Applications (2010)
11. Samuel, K., Obrst, L., Stoutenberg, S., Fox, K., Franklin, P., Johnson, A., Laskey, K., Nichols, D., Lopez, S., Peterson, J.: Applying Prolog to Semantic Web Ontologies & Rules: Moving Toward Description Logic Programs. The Journal of the Theory and Practice of Logic Programming (TPLP), Massimo Marchiori, ed., Cambridge University Press, Volume 8, Issue 03, pp 301-322 (2008)
12. Obrst, L., McCandless, D., Stoutenberg, S., Fox, K., Nichols, D., Prausa, M., Sward, R.:. Evolving Use of Distributed Semantics to Achieve Net-centricity. Regarding the “Intelligence” in Distributed Intelligent Systems, AAAI Fall Symposium, Arlington VA, Nov. 8-11 (2007)
13. Stoutenberg, S., Obrst, L., Nichols, D., Franklin, P., Samuel, K., Prausa, M.: Ontologies and Rules for Rapid Enterprise Integration and Event Aggregation. Vocabularies, Ontologies and Rules for the Enterprise (VORTE 07), EDOC 2007, Annapolis, MD, Oct. 15-19 (2007)
14. McCandless, D., Obrst, L.: Using Ontology Alignment to Dynamically Chain Web Services. Ontology Matching Workshop, poster, International Semantic Web Conference (ISWC) 2009, Oct. 25-29, Chantilly, VA (2009)
15. McCandless, D., Obrst, L.: Dynamic Web Service Chaining using OWL and a Theorem Prover. Third IEEE International Conference on Semantic Computing, Berkeley, CA, USA - September 14-16 (2009)
16. Ait-Kaci, H., Boyer, R., Lincoln, P., Nasr, R.: Efficient Implementation of Lattice Operations. TOPLAS 11-1 (1989)

17. Ait-Kaci, H.: A Lattice-Theoretic Approach to Computation Based on a Calculus of Partially-Ordered Type Structures. Ph.D thesis, Computer and Information Science Dept., Univ. of Pennsylvania, Philadelphia, PA (1984)
18. Caseau, Y., Habib, M., Nourine, L., Raynaud, O.: Encoding of Multiple Inheritance Hierarchies and Partial Orders. Computational Intelligence 15 (1), pp. 50-62 (1999)
19. Fall, A.: Heterogeneous Encoding. In Proceedings of International KRUSE Symposium: Knowledge Retrieval, Use, and Storage for Efficiency, Gerard Ellis, Robert Levinson, Andrew Fall, Veronica Dahl, eds., Santa Cruz, CA, Aug. 11-13, pp. 134-146 (1995)
20. Hellerstein, J.M., Naughton, J.F., Pfeffer, A.: Generalized search trees for database systems. In: Proceedings of the 21st International Conference of Very Large Data Bases, VLDB'95 (1995)
21. Krall, A., Vitek, J., Horspool, N.: Near optimal hierarchical encoding of types. 11th European Conference on Object Oriented Programming (ECOOP'97). Springer (1997)
22. Preuveneers, D., Berbers, Y.: Prime numbers considered useful: Ontology encoding for efficient subsumption testing, Tech. Rep. CW464. http://www.cs.kuleuven.be/publicaties/rapporten/cw/CW464.abs.html. Department of Computer Science, Katholieke Universiteit Leuven, Belgium (October 2006).
23. van Bommel, M. F., Wang, P.: Encoding multiple inheritance hierarchies for lattice operations. Data & Knowledge Engineering, Volume 50, Issue 2, August, pp. 175-194 (2004)