Building Applications and Tools for OWL – Experiences and Suggestions

Thorsten Liebig[1], Marko Luther[2], Olaf Noppens[1], Massimo Paolucci[2], Matthias Wagner[2], and Friedrich von Henke[1]

[1] Dept. of Artificial Intelligence, University of Ulm, D-89069 Ulm, Germany
thorsten.liebig@uni-ulm.de, olaf.noppens@uni-ulm.de
friedrich.von-henke@uni-ulm.de
[2] Future Networking Lab, DoCoMo Communications Laboratories Europe, D-80687 Munich, Germany
<lastname>@docomolab-euro.com

Abstract. The success of the Semantic Web will largely depend on whether W3C's Web Ontology Language can reach broad acceptance and a critical mass of industry-strength applications. We have been exploiting the use of OWL with a particular focus on tool support for ontology authoring and on providing access to the Semantic Web for mobile applications. In the latter case our vision is to overlay the Semantic Web on ubiquitous computing environments making it possible to represent and interlink content and services as well as users, devices, their capabilities and the functionality they offer. In this paper we present our first experiences and lessons learned from early work and try to give constructive feedback for possible enhancements of OWL and its tools.

1 Introduction

OWL ontologies are arguably key building blocks of the future Semantic Web. In fact, the success of the Semantic Web will largely depend on whether W3C's Web Ontology Language can reach broad acceptance and a critical mass of industry-strength applications. In our research – with the vision of mobile and ubiquitous access to the Semantic Web in mind – ontologies are also crucial in aspects of mobile and pervasive computing. For the past few years, we have exploited the usage of OWL within several practical projects that we briefly introduce in the following. These projects are either concerned with fundamental support for OWL-based development and/or with OWL-based services and applications in the mobile computing arena.

After shortly presenting our projects in the following section, we discuss fundamental language as well as technical aspects of the Web Ontology Language from the perspective of developing OWL applications and tools. Within the context of our projects we have identified limitations and problematic issues in the language specification itself as well as in the tool support. As the usage of ontologies is typically tightly coupled with reasoning systems, a particular focus is put on DL-based reasoning support for OWL.

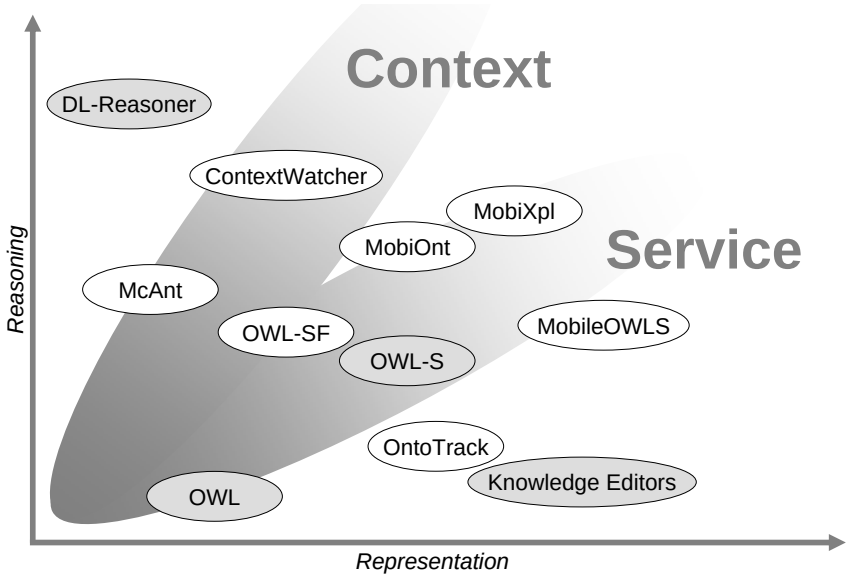

Fig. 1. Overview of Semantic Web project activities exploiting OWL.

2 Ongoing Projects

Figure 1 sketches the range of projects related to the Semantic Web that we currently pursue in our laboratories (in white shading). It also positions these activities w.r.t. established (pre-)standards and Semantic Web components (grey shading). On a research map, our ongoing project activities can be aligned according to their level of concern with OWL fundamentals in terms of representation capabilities and reasoning support.

While OntoTrack aims at tool support for OWL developers, the remaining projects can be projected to an additional dimension representing activities that target at semantic-based mobile applications in terms of support for context representation and management, support for mobile services or both of these.

2.1 OWL-SF

Major challenges in designing ubiquitous context-aware systems include the distributed nature of context information and the heterogeneity of devices that provide services and deliver context. We have approached these challenges within the project Owl-SF [1], a distributed semantic-based service framework. In the Owl-SF prototype, OWL is used to capture high-level context elements in semantically well founded ways. Devices, sensors and other environmental entities are encapsulated and connected to the upper context ontology using OMG's Super Distributed Objects technology and communicate using the Representa-

tional State Transfer model. An early use case of OWL-SF studies enhanced presence control to realize intelligent call forwarding [2].

2.2 ContextWatcher

Within the IST project MobiLife[3] [3] we have implemented CONTEXTWATCHER, an early prototype for semantic-based monitoring of mobile users. The project aims at frameworks for context-aware services that support users in their daily life. OWL upper context ontologies define the basic contextual categories and the relations among them. Such high-level structuring of context information enables its integration and consolidation on a semantic basis. Furthermore, the axiomatic descriptions of context elements such as personal situations (i.e., Working, At_home, etc.) can directly be used by logical inference engines to realize reasoning about the user's presence and virtual location [4].

2.3 McAnt

With our prototype MCANT we try to explore possibilities and core technologies towards leveraging the Semantic Web for desktop application enhancements [5]. The idea is to deploy qualitative reasoning on the user's personal desktop environment to enable a more refined support for personal information organization. As in our related activities, OWL and DL-based reasoning are explored as enabling technologies for semantic enrichment. MCANT currently extends Apple's desktop environment in terms of the Apple Address Book and the iCal calendar tool. The project introduces a set of core ontologies that describe novel OWL-based smart groups that build on Apple's smart groups and folders.

2.4 OntoTrack

ONTOTRACK [6] is a novel graphical ontology editor and authoring tool. In contrast to many other ontology authoring tools it combines an interactive graph-based visualization with instant reasoning feedback about logical consequences of ontology changes. The graphical representation provides a directly editable subsumption graph. Moreover, each editing step is instantly synchronized with a reasoner to provide feedback about logical consequences such as subsumption relationships or unsatisfiablity of classes. This kind of direct feedback promise to help the user identify non-intended modeling errors. Recently, we extended ONTOTRACK with an on-demand textual explanation of subsumption.

2.5 MobiONT and MobiXPL

The discovery of adequate services will become a more and more demanding problem especially for the mobile user who has to cope with changing context

[3] MobiLife is an integrated project within the 6th Framework Program of the European Commission, Project-No. IST-2004-511607(IP)

and limitations of mobile terminals. We have implemented MobiOnt and MobiXpl – a semantic matchmaker for service discovery and a personal mobile client – to explore mobile user-centered services on the Semantic Web [7]. Our vision is to take full advantage of future complex service offerings on limited client devices and to handle the need for personalized service discovery in mobile environments. Main contributions are in support for browsing service ontologies, the cooperative discovery of services as well as an intuitive preference model that can be easliy managed on restricted clients [8].

2.6 MobiOWLS

MobiOwlS is a new project in which we attempt to extend the OWL-S [9] upper ontology for services to describe services for mobile and ubiquitous computing. Our initial investigation concentrated on extending the OWL-S Profile to include crucial quality-of-service information and contextual information that in our experience is required to locate the best service that satisfies the needs of the user. For example, we extended OWL-S with properties such as `Media` that specifies the type of media, such as video vs text, that is used to deliver the service, or the `CostModel` of the service, such as flat rate or a fee-per-use.

3 Experiences and Suggestions

3.1 Representation

References, Imports, and Ontology Headers. As a vocabulary extension of RDF the Web Ontology Language inherits all of the reference and distribution mechanisms of the lower language levels. For example, an OWL class (or property) can be identified with help of an unambiguous URI given in the `rdf:ID` tag. Furthermore, OWL allows for the distribution of a class (or property) across multiple defining axioms within various RDF documents. It is even possible to define a class bearing a virtual URI not related to any of the defining documents. Syntactically this will raise a number of issues not only w.r.t. editing support with help of authoring tools [6]. Because, when referring to this class from elsewhere, its corresponding definition cannot be found at the URI implied by the class ID.

Another problem is related to imports and ontology editing. Importing means to include all statements of the imported ontology into the importing ontology. Now, an ontology authoring tool has to distinguish between those statements of the importing and those of the imported ontology during editing [10]. Imported axioms should be visible because they are important for capturing the complete set of definitions of the importing ontology. On the other hand, the imported axioms do not belong to the ontology the user currently is editing. Very likely the user is not interested to change imported ontologies by accident.

Furthermore, the notion of an ontology, by specifying an ontology header with the `owl:Ontology` tag, is an additional structuring element within OWL.

Such an ontology header is optional or may even appear more than once in a document. As a result, the relationship between class and property axioms and an ontology is unclear. To make things even more confusing, an ontology header is the only way to import other documents or to encode versioning information for ontologies.

In addition, the vision of the Semantic Web heavily builds on sharing and re-using ontologies. However, neither importing nor referencing are appropriate mechanisms for this task. Importing requires to include all ontologies (within the transitive closure of the import relationship) into reasoning. In contrast, referencing entities of other ontologies is just a syntactical mechanism without considering any semantics. To overcome these fundamental problems, OWL needs a clearer semantical foundation, and practical mechanisms for handling imports, ontologies and cross ontology references in first place. The recently proposed $\mathcal{E}$-connection mechanisms [11] seem to be a good starting point for supporting the sharing and re-usage of ontologies.

The Sublanguages of OWL. OWL comes with three increasingly expressive sublanguages in order to meet different user requirements and developer efforts. OWL Lite – the least expressive language – is aiming to provide a useful subset of language features that are easy to present to naive users and relatively straightforward for tool implementers to support [12]. However, the syntactical limitations come with relatively little loss in expressive power. With the help of syntactical tricks all of OWL DL, the sub-language next-higher in expressiveness, can be captured in OWL Lite with the exception of those descriptions containing either individuals or cardinalities greater than 1 [13]. In fact, OWL Lite is a very expressive language, whose set of syntactical language constructs has been more or less randomly restricted. As a result, when using the whole expressiveness of OWL Lite the resulting ontology is much harder to grasp as a semantically equivalent OWL DL representation. The most expressive sub-language, OWL Full, combines OWL DL with all of RDF(S). As a result of non-standard and second order features of RDF(S) (reification, no division between individuals and classes, etc.) OWL Full is not decidable which restricts its practical use. Furthermore, OWL Full bears conceptual problems like the possibility to redefine language constructs of OWL itself.

In conclusion, OWL virtually consists of only one language, OWL DL, since Lite is nearly as complex as DL and Full is not attractive for semantical and computational reasons. We suggest to restrict OWL Lite in a way that we believe was originally intended by the language designers. In particular, we propose to forbid general concept inclusions (GCIs) as well as multiple definitions for one class identifier, as we have done in the context of our ontology authoring tool OntoTrack [6]. Very likely those constructs will rarely occur in real world OWL Lite ontologies. Instead, users will presumably use OWL DL when disjunction or negation are needed for a certain application domain. In addition, we believe it would make sense to have a set of more finely graded sublanguages ranging from $\mathcal{AL}$ or $\mathcal{ALC}$ up to OWL DL, including at least two intermediate sublanguages.

First, this probably would make it easier for novice users to adopt OWL. Second, the less expressive languages would benefit from a broader variety of reasoning technology (e. g. from the Datalog/DB community). Third, this would allow us to add non-standard language extensions (see below) to OWL, which typically require to be combined with less expressive languages.

Language Extensions. OWL is a declarative language tailored to represent conceptual knowledge. Many applications, however, require the modeling of actions, states, uncertainty, or an explicit notion of time. Besides the ongoing efforts for an Semantic Web Rules Language (SWRL) we strongly suggest the development of other extensions. For example, recent extensions of Description Logics with fuzzy modifiers [14,15] should be taken into consideration for further investigations. Other DL extensions concerning action formalisms [16] and time [17] are also candidates for optional OWL add-ons. Beyond that, OWL lacks an official query language which is needed in order to have a common platform for applications and reasoning systems.

Representing Services. Our experience with MobiOWLS highlighted two problems related to representing and reasoning with service descriptions. The first problem is that it is very difficult to restrict the ontology within the boundaries of OWL DL. For example, it is very difficult to describe a service that works only in some contexts such as office buildings. The problem of such a description is that it is within OWL Full, because the instance of a service refers to the class of all office buildings. To work around this problem we adopted the same solution adopted by OWL-S, which is to refer to URIs of the OWL classes and instances that we want to use. As a consequence our ontology is still within the boundaries of OWL-DL, at the expense of essentially giving up OWL reasoning on our service descriptions. The second problem is that there is no representation, and reasoning, for quantities and comparing quantities. For example, it is impossible to define a broadband terminal as a terminal with bandwidth greater than 1Mbps or any other quantity. In general, the lack of representation and reasoning about quantities hampers the representations of important quality of service features of services, greatly reducing the usefulness of OWL for the representation of services.

3.2 Reasoning

Incremental Reasoning and Retraction. Our experience in developing ontologies shows us that instant reasoning feedback is a key functionality for ontology authoring tools. Therefore, we developed OntoTrack to effectively support the user in creating and maintaining ontologies [6]. It should not rely on the user's discipline to (re-)classify the ontology after some editing steps. Moreover, understanding all classification consequences (such as taxonomy changes and inconsistency) even after a few editing steps becomes a difficult task.

Unfortunately, current OWL reasoners typically only provide some kind of batch-oriented reasoning procedure. After loading an ontology and classification one can pose TBox as well as ABox queries. But there is no possibility to retract or change statements without reloading the updated ontology. A notable exception here is RACER [18] which offers low-level retraction support for most of its statements. However, because of the lack of algorithms for appropriatly handling incremental additions as well as retractions [19], complete reclassification is necessary after each change in the ontology. In addition, when using an optimized tableaux-based reasoner for a language as expressive as OWL, retracting and changing definitions may be of high complexity because of optimization techniques such as absorption. However, incremental reasoning and retraction is an important premise for almost all interactive OWL-based tools or applications. An ad-hoc solution could consist of a heuristic that analyzes the retraction statement and decides about local deletion of statements or complete reclassification.

Furthermore, technical aspects concerning standard interfaces and communication are also an important issue for building interoperable applications. State-of-the-art reasoners should be network-aware, able to manage multiple clients, and support standard interfaces such as DIG [20]. We suggest to extend the DIG interface for incremental reasoning and retraction. Even if reasoners do not distinguish between some kind of on-the-fly retractions and reclassification, providing such functionality in DIG will not only reduce the application's burden in an interactive environment such as ontology authoring, but also minimize the amount of unnecessary data transfer. For instance, interactive applications (resp. reasoner) which would like to perform retractions via a retraction-unaware interface would have to re-submit the whole ontology after each retraction step althought most of the explicit statements are unmodified and therefore typically already available in the reasoner system as explicitly added statements.

Non-standard Inference Services. OWL is expected and explicitly intended to be used by users without much formal background. Therefore, the ability to support users to understand and debug ontologies is crucial for the Semantic Web in general and identified as a major research challenge [21]. Non-standard reasoning services are of cource not directly related to the OWL specification. However, they are important for users in order to design and maintain sound and well-balanced ontologies. Therefore, we suggest to add some of them as an optional extension to standard interfaces commonly used within OWL reasoning.

Potential helpful non-standard services mainly cover explaining the reasoning about and debugging of ontologies on both the conceptual (TBox) and the instance (ABox) side. Explaining TBox queries like subsumption, unsatisfiability, and equivalence for nearly all of OWL Lite can be found in [22]. There is also work on explaining ABox query answers [23] as well as why such a query failed [24]. Debugging of ontologies by identifying the core of incoherence also produced some prototypical services [25], [26]. There are additional non-standard reasoning services suitable to support ontology authoring like matching of class patterns,

rewriting or approximation [27] as well as the least common subsumer (TBox) or the most specific concept (ABox) [28]. Work on explaining non-subsumption (why a subsumption does not hold) is still missing.

3.3 Infrastructure

Publish-Subscribe Standard Protocol. It is not only applications like OntoTrack which are sensitive to incremental logical consequences. Almost all reasoning intensive applications require an API offering a high-level publish-subscribe mechanism. Otherwise such an application would be responsible for querying the reasoner for all consequences it is interested in – or even worse the application needs to compute or filter the desired consequences out of low-level reasoner queries. For instance, to become aware of changes in the ontology's hierarchy an application has to check for direct superclasses for almost all classes in the worst case. Then it has to compute the differences with respect to the previous state. Using an external reasoner via a standard interface such as DIG [20] will cause additional communication overhead, especially if the ratio between ontology entities (or axioms) involved in the change(s) and queries needed to perform is low. However, information about logical consequences are typically already available on reasoner side after reclassification.

We therefore propose a TBox publish-subscribe mechanism on the reasoner side which roughly corresponds to a similar technique of RACER's ABox publish-subscribe mechanism. As not all applications will be interested in all ontology changes (or even in none of them) and in order to minimize a possible notification overhead, such a publish-subscribe is to be characterized as follows: (i) it should be optional, (ii) specific w.r.t. the kind of ontology query, (iii) selective to the set of ontology entities (subset or all), and (iv) granular on the interval it will give its notification feedback. Adjusting the granularity may allow for defining notification "points" in an incremental environment. For instance, an application may only want to be informed about the consequences of a whole set of changes – not about the consequences of each of the changes separately.

In order to standardize such a publish-subscribe mechanism we suggest to extend the specification of the common DIG interface in the following way. We propose to divide the DIG interface into two parts: the core DIG and optional extensions. It is desirable that the core will be extended to support an interface for incremental reasoning and retraction of statements. The extensions should cover a publish-subscribe mechanism as well as non-standard inference service (see the previous subsection). The main idea behind a modularized DIG interface is that all DIG-compliant reasoners support the core interface, but it depends on the functionality of the reasoner which extension it will support – the more the better.

4 Conclusion and Suggestions

The wide adoption and success of Semantic Web applications strongly depends on a description language reflecting the character of distributed resources on

the Semantic Web, as well as on inference services and appropriate interfaces to access them. Due to our experience in a broad spectrum of Semantic Web applications ranging from an ontology tool to applications in mobile and service-related environments, we have identified general problematic issues.

We argue that for sharing and reusing ontologies the currently provided ontology import, which brings all triples into the imported ontologies, is not sufficient. Therefore, an appropriate mechanism on the syntactical as well as the semantic level is necessary for importing ontologies and referencing entities in other ontologies. Another important issue is the sublanguage ranking of OWL. We have shown that OWL basically consists only of one language and therefore propose a more finely graded sublanguage ranking. From a less expressive sublanguage than OWL DL non-standard language extensions can also benefit which may be otherwise un-decidable.

In addition, in a number of applications we have encountered the limitations of purely conceptual knowledge. We therefore suggest to also take into consideration extensions such as action formalisms, time, and uncertainty. Technical issues such as non-standard inference services and incremental reasoning are also important, especially for interactive environments. We propose a modularized extension to the common DIG interface not only to support incremental reasoning and retraction of statements but also to plug-in additional extensions for non-standard inference services as well as a publish-subscribe protocol for the notification of logical changes within an ontology.

References

1. Mrohs, B., Luther, M., Vaidya, R., Wagner, M., Steglich, S., Kellerer, W., Arbanowski, S.: OWL-SF – a distributed semantic service framework. In: Proc. of the Workshop on Context Awareness for Proactive Systems (CAPS'05), Helsinki (2005) 67–77
2. Luther, M., Mrohs, B., Vaidya, R., Wagner, M.: OWL-SF – distributed owl-based reasoning on objects in the real world. In: Proc. of ISWC'05 Demo Track, Galway (2005)
3. MobiLife: Project homepage. http://www.ist-mobilife.org (2005)
4. Luther, M., Böhm, S., Wagner, M., Koolwaaij, J.: Enhanced presence tracking for mobile applications. In: Proc. of ISWC'05 Demo Track, Galway (2005)
5. Böhm, S., Luther, M., Wagner, M.: Smarter groups – reasoning on qualitative information from your desktop. In: Proc. of the 1st Workshop on The Semantic Desktop at ISWC'05, Galway (2005)
6. Liebig, T., Noppens, O.: OntoTrack: A semantic approach for ontology authoring. Journal of Web Semantics **3** (2005) in press.
7. Wagner, M., Liebig, T., Noppens, O., Balzer, S., Kellerer, W.: Towards Semantic-based Service Discovery on Tiny Mobile Devices. In: Proc. of the Workshop on Semantic Web Technology for Mobile and Ubiquitous Applications at ISWC'04, Hiroshima (2004) 90–101
8. Wagner, M., Noppens, O., Liebig, T., Luther, M., Paolucci, M.: Semantic-based Service Discovery on mobile Devices. In: Proc. of ISWC'05 Demo Track, Galway (2005)

9. Martin, D., Burstein, M., Hobbs, J., Lassila, O., McDermott, D., McIlraith, S., Narayanan, S., Paolucci, M., Parsia, B., Payne, T., Sirin, E., Srinivasan, N., Sycara, K.: OWL-S: Semantic Markup for Web Services. Member Submission, W3C (2004)
10. Kalyanpur, A., Parsia, B., Hendler, J.: A Tool for Working with Web Ontologies. Journal of Semantic Web and Information Systems **1** (2005) 36–49
11. Grau, B.C., Parsia, B., Sirin, E.: Working with Multiple Ontologies on the Semantic Web. In: Proc. of the 3rd Int. Semantic Web Conference (ISWC'04), Hiroshima, Springer (2004) 620–634
12. Bechhofer, S., van Harmelen, F., Hendler, J., Horrocks, I., McGuinness, D.L., Patel-Schneider, P., Stein, L.A.: OWL Web Ontology Language Reference. W3C Recommendation (2004)
13. Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From SHIQ and RDF to OWL: The making of a web ontology language. Journal of Web Semantics **1** (2003)
14. Straccia, U.: Fuzzy $\mathcal{ALC}$ with Fuzzy Concrete Domains. In: Proc. of the Int. Workshop on Description Logics (DL'05), Edinburgh (2005) 96–103
15. Hölldobler, S., Nga, N.H., Khang, T.D.: The Fuzzy Description Logic $\mathcal{ALC}_{FLH}$. In: Proc. of the Int. Workshop on Description Logics (DL 2005), Edinburgh (2005)
16. Baader, F., Lutz, C., Miličić, M., Sattler, U., Wolter, F.: Integrating Description Logics and Action Formalisms: First Results. In: Proc. of the Int. Workshop on Description Logics (DL'05), Edinburgh (2005) 192–199
17. Artale, A., Franconi, E.: A Survey of Temporal Extensions of Description Logics. Annals of Mathematics and Artificial Intelligence (AMAI) **30** (2001) 171–210
18. Haarslev, V., Möller, R.: RACER System Description. In: Proc. of the Int. Joint Conference on Automated Reasoning (IJCAR'2001), Siena, Springer (2001) 701–705
19. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P., eds.: Description Logics Systems. In: The Description Logic Handbook. Cambridge University Press (2003)
20. Bechhofer, S.: The DIG Description Logics Interface: DIG/1.1. Technical report, University of Manchester (2003)
21. Horrocks, I.: Applications of Description Logics: State of the Art and Research Challenges. In: Proc. of the 13th Int. Conf. on Conceptual Structures (ICCS'05). (2005) to appear.
22. Liebig, T., Halfmann, M.: Explaining Subsumtion in $\mathcal{ALEHF}_{R^+}$ TBoxes. In: Proc. of the Int. Workshop on Description Logics (DL'05), Edinburgh (2005) 144–151
23. McGuinness, D., da Silva, P.: Explaining Answers from the Semantic Web: The Inference Web Approach. Journal of Web Semantics **1** (2004) 397–413
24. Chalupsky, H., Russ, T.: WhyNot: Debugging Failed Queries in Large Knowledge Bases. In: Proc. of the Innovative Applications of Artificial Intelligence Conf. (IAAI-02), Edmonton, AL, Canada (2002) 870–877
25. Parsia, B., Sirin, E., Kalyanpur, A.: Debugging owl ontologies. In: The 14th International World Wide Web Conference (WWW2005), Chiba, Japan (2005)
26. Wang, H., Horridge, M., Rector, A., Drummond, N., Seidenberg, J.: A Heuristic Approach to Explain the Inconsistency in OWL Ontologies. In: Proc. of the Int. Protégé Conference, Madrid, Spain (2005)
27. Brandt, S., Turhan, A.Y.: An Approach for Optimized Approximation. In: Proc. of the Workshop on Applications of Description Logics (ADL'02). (2002)
28. Baader, F., Sertkaya, B., Turhan, A.Y.: Computing the Least Common Subsumer w.r.t. a Background Terminology. In: Proc. of the 2004 Int. Workshop on Description Logics (DL 2004), Whistler (2004) 11–20