Final 5/3/03

Description Logics for the Semantic Web:
Racer as a Basis for Building Agent Systems

Ralf Möller, Univ. of Applied Sciences of Wedel, 22880 Wedel, Feldstr. 143
Volker Haarslev, Concordia University, Comp. Sc. Dept., 1455 de Maisonneuve
Blvd. W., Montreal, Quebec H3G 1M8, Canada

The term Semantic Web denotes a vision of a new World-Wide Web in which different kinds of resources (data, services, Web pages, etc.) are accessed and shared on the basis of formal representation structures [7]. The main idea of the Semantic Web is to answer different kinds of queries w.r.t. a specific conceptual data model rich enough in expressivity to adequately describe the interrelationships between basic representational units (names, symbols) on which a piece of software for a certain application domain is based. Since symbols might have different meanings in different software applications, it is important to explicitly refer to meaning definitions given as a certain instance of an underlying conceptual data model. In the context of the Semantic Web, these definitions are seen as resources themselves and, for reasons of brevity, are often called ontologies. Ontologies provide the basis for implementing specific Semantic Web resources for encapsulating computational processes. These resources are called services [19].

In order to be unambiguous and suitable for machine processing, ontologies must be defined with reference to a formal semantics. A logical semantics provides the basis not only for the meaning of the representational language but also for the meaning of the associated query and command language. In many applications the logic behind conceptual data models must be decidable but very expressive [1] such that, in principle, query answering involves solving nontrivial logical inference problems. This article briefly introduces the foundation of Semantic Web ontology representation languages, namely description logics [2]. For actually building systems for the Semantic Web, however, we consider practical aspects of inference systems and their integration into a distributed systems context. Using the Racer System [16] as an example the paper sketches how description logic inference engines can be used to implement, for example, information retrieval services on the Semantic Web.

Agents, Ontologies, and Problem-Solving

The demand for more and more flexible software systems led to the emergence of the agent paradigm for software development [22]. In particular, in the context of the Semantic Web, agent systems offering and utilizing services are an integral part of the grand vision. The development of autonomous agents for interacting with software systems in foreign environments is a difficult problem, and there are three different perspectives on agent systems that are to be discussed here.

First of all, almost all software systems, including agent systems, are based on some assumptions about the meaning of names. Hence, for constructing an agent system, the names used in the agent architecture itself must be systematically related to one another in order to ensure the development of a system architecture that is understandable, extensible, adaptable, etc. This is known as conceptual data modeling at agent development time [8].

Second, and probably even more challenging, there is the problem of relating the names of the terminology internally used by an agent to the names used in the host environment the agent is eventually interacting with. The problem of assigning meaning to names is prevalent since the meaning is subject to explicit computational processing at agent runtime (see [9]).

Third, the ultimate goal could be to implement (parts of) problem-solving processes of agents such as, for instance, information retrieval, planning, scheduling, controlling, negotiation etc. on the basis of reasoning processes, i.e. subproblems of applications are to be modeled as logical decision problems. Specific ontologies are required for representing problem-solving knowledge and for supporting adequate logical formalizations of application problems to be solved when the agent is in operation [2].

In all three views discussed above, ontologies are a prerequisite for agent problem solving on the Semantic Web. In the next section we give an overview about the logical background for defining ontologies.

Description Logics for Specifying Ontologies and Knowledge Bases

The first decision to be made for developing an ontology concerns so-called ontological commitments [22, p. 165]. What is the assumed nature of reality? Experiences from decades of research and practice indicate that a view in which the world consists of objects and relations between them provides an appropriate basis for many applications. Description logics exactly support this view. Concept names denote sets of objects from a so-called universe of discourse, role names denote sets of tuples of objects. For historical reasons, (binary) relations between objects are called roles, the objects on the right-hand side of a tuple in a relation are called role fillers. In order to demonstrate the benefit of description logics for developing ontologies we consider a very basic scenario everyone should be able to intuitively understand.

First of all, for a particular ontology the names for basic concepts and roles have to be chosen. In our example, we consider concept names such as HUMAN, PARENT, WOMAN, MAN, AUDI, BWM, MERCEDES, TAXI, and so on. In addition, role names used in our example ontology are the following: *has-child*, *has-part*, *owns*, etc.

In order to adequately describe sets of objects for certain application purposes (see below for an agent information retrieval application), names can be combined using concept construction operators to form new concepts. For instance, the concept "a man who has a child which is a parent" can be specified as

$$\text{MAN} \sqcap (\exists \textit{ has-child } \text{PARENT})$$

Note that for all objects which are member of the set of objects described by this concept, the child has to exist (and it must be a person). Sometimes it is not the goal to specify the existence of a filler for a certain role but rather to specify restrictions for all fillers of a role (if there are any). For instance, "women all of whose children are men" can be described with

$$\text{WOMAN} \sqcap (\forall \textit{ has-child } \text{MAN})$$

Note that the restriction is met if the woman has no children at all. Another example demonstrates the use of negation. The catholic church still forbids female popes such that ¬WOMAN, the complement of the set denoted by WOMAN, might be part of a concept for

describing popes. Concepts can be recursively combined. For instance, the notion of "a human who owns an AUDI, a BMW or a MERCEDES" can be specified using the expression

$$\text{HUMAN} \sqcap (\exists \ owns \ (\text{AUDI} \sqcup \text{BMW} \sqcup \text{MERCEDES}))$$

The nested concept (AUDI ⊔ BMW ⊔ MERCEDES) also demonstrates the representation of disjunctive or underspecified information. Another example could be a "woman with at least 2 female children but not more than 4 children overall". This can be described with the concept

$$\text{WOMAN} \sqcap (\geq 2 \ has\text{-}child \ \text{WOMAN}) \sqcap (\leq 4 \ has\text{-}child)$$

So far we have used concepts for *describing* sets of objects. However, for specifying ontologies it is also important to relate different sets of objects to each other, for example w.r.t. the subset relationship. This can be done using so-called terminological axioms. The simplest example uses just names:

$$\text{MAN} \sqsubseteq \text{HUMAN} \quad \text{and} \quad \text{WOMAN} \sqsubseteq \text{HUMAN}$$

The set described by the symbol MAN is a subset of the set described by HUMAN etc. Ontological modeling in description logics is not limited to using names only. We might want to enforce that all taxies are autos with at least 4 seats.

$$\text{TAXI} \sqsubseteq \text{AUTO} \sqcap (\geq 4 \ has\text{-}part \ \text{SEAT})$$

$$\text{SMART} \sqsubseteq \text{AUTO} \sqcap (\leq 2 \ has\text{-}part \ \text{SEAT})$$

The axiom specifies necessary restrictions for all taxis and Smarts. Once it is known that an object is a taxi, it becomes clear that it has at least 4 seats. On the other hand, an auto with at least 4 seats is not necessarily a taxi. Similar restrictions are imposed for Smarts. As another example a necessary restriction for all parents is imposed as follows:

$$\text{PARENT} \sqsubseteq \text{HUMAN} \sqcap (\exists \ has\text{-}child \ \text{HUMAN})$$

Disjointness of concepts can easily be declared with axioms of the form:

$$\text{MAN} \sqsubseteq \neg\text{WOMAN}$$

The set of men is included in the set of non-women and, therefore, it is disjoint from the set of women. Coverings can also be expressed.

$$\text{HUMAN} \sqsubseteq (\text{WOMAN} \sqcup \text{MAN})$$

The set of humans is covered by the union of the sets with names WOMAN and MAN. Other conceptual modeling tasks can also be accomplished using terminological axioms. For instance, once we know that a human has a child which is a human, this is also sufficient to call this human a parent.

$$\text{HUMAN} \sqcap (\exists \ has\text{-}child \ \text{HUMAN}) \sqsubseteq \text{PARENT}$$

The axiom can be paired with the above axiom for PARENT. Since such a pair of axioms C ⊑ D and D ⊑ C is quite common in ontology modeling, the shorthand C ≡ D for both axioms is often used as follows.

$$\text{PARENT} \equiv \text{HUMAN} \sqcap (\exists \textit{ has-child } \text{HUMAN})$$

In a similar way necessary and sufficient conditions for the concept UNCLE can be specified:

$$\text{UNCLE} \equiv \text{MAN} \sqcap (\exists \textit{ has-sibling } \text{PARENT})$$

Using description logics, terminological knowledge for ontologies can be specified in an adequate way for many applications. Note that it is absolutely possible to specify sufficient conditions only. For instance, if it is known that for someone there exists a child which is a human, then this someone must be a mother or a father.

$$(\exists \textit{ has-child } \text{HUMAN}) \sqsubseteq (\text{MOTHER} \sqcup \text{FATHER})$$

In ontologies, not only concepts are important, but restrictions on roles have to be specified as well. For instance, one might want a role (relation) *has-mother* to be functional because only one mother should be assigned to each object. The role *has-descendant* might be defined as transitive, and the role *has-child* should probably be a subrole of *has-descendant* because every tuple in the relation *has-child* should also be in the relation *has-descendant*. The relation *has-descendant* is the inverse of *has-ancestor*. In addition, for our ontology we might assume that for the relation *has-child*, the domain and range are PARENT and HUMAN, respectively (see also [3] for many examples involving the above-mentioned modeling constructs).

In description logics, for historical reasons, a set of axioms of these kinds is also referred to as a T-box. Using T-boxes adequate conceptual notions required for ontologies can be defined. In terms of description logics, the name ontology is often used as a synonym for T-box.

Dealing with sets of objects and T-boxes for conceptual modeling we have painted only half of the complete picture of description logics. For agent problem solving, referring to single objects and tuples of single objects is equally important, of course. Names for single objects are to be specified as well (these names are usually called individuals). In our family domain, we might need individuals such as ALICE, BETTY, etc. For declaring knowledge about individuals so-called assertional axioms are provided. If we assume that ALICE is a woman, this can be stated as

$$\text{ALICE} : \text{WOMAN}$$

With an assertion of the form
$$(\text{ALICE, BETTY}) : \textit{has-child}$$

individuals of our family domain can be related to one another.

A set of these assertions is called an A-box. A pair (T, A) of a T-box T and an A-box A is called a knowledge base. The name ontology is often also used to refer to a knowledge base.

As a summary, it should be mentioned that Racer supports the logic $SHIQ(D_n)^-$ [17, 12] including reasoning about individuals (A-box reasoning). Due to space restrictions we cannot explain all details of $SHIQ(D_n)^-$ here.

Reasoning and Inference Problems

Given a T-box, various kinds of queries can be answered. Based on the logical semantics of the representation language, different kinds of queries are defined as inference problems (hence, answering a query is called providing inference service). As a summary, we list only the most important ones here and discuss examples in the context of an agent scenario later on:

> Concept consistency w.r.t. a T-box: Is the set of objects described by a concept empty?
> Concept subsumption w.r.t. a T-box: Is there a subset relationship between the set of objects described by two concepts?
> Find all inconsistent concepts mentioned in a T-box. Inconsistent concepts might be the result of modeling errors.
> Determine the parents and children of a concept w.r.t. a T-box: The parents of a concept are the most specific concept names mentioned in a T-box which subsume the concept. The children of a concept are the most general concept names mentioned in a T-box that the concept subsumes. Considering all concept names in a T-box the parent (or children) relation defines a graph structure which is often referred to as taxonomy. Note that some authors use the name taxonomy as a synonym for ontology.

Note that whenever a concept is needed as an argument for a query, not only predefined names are possible. If also an A-box is given, among others, the following types of queries are possible:

> Check the consistency of an A-box w.r.t. a T-box: Are the restrictions given in an A-box w.r.t. a T-box too strong, i.e., do they contradict each other? Other queries are only possible w.r.t. consistent A-boxes.
> Instance testing w.r.t. an A-box and a T-box: Is the object for which an individual stands a member of the set of objects described by a certain query concept? The individual is then called an instance of the query concept.
> Instance retrieval w.r.t. an A-box and a T-box: Find all individuals from an A-box such that the objects they stand for can be proven to be a member of a set of objects described by a certain query concept.
> Computation of the direct types of an individual w.r.t. an A-box and a T-box: Find the most specific concept names from a T-box of which a given individual is an instance.

To be practically useful, description logics have to be integrated into current Web software architectures. We first turn to syntax issues and deal with distributed access to inference services afterwards.

Description Logics and Semantic Web Representation Languages

One of the standards for the Semantic Web is the Resource Description Format (RDF, [18]). Since RDF is based on XML it shares its document-oriented view of grouping sets of declarations or statements. With RDF's triple-oriented style of data modeling, it provides means for expressing graph-structured data over multiple documents (whereas XML can only

express graph structures within a specific document). As a design decision, RDF can talk about everything. Hence, in principle, statements in documents can also be referred to as resources. In particular, conceptual domain models can be represented as RDF resources. Conceptual domain models are referred to as "vocabularies" in RDF. Specific languages are provided for defining vocabularies (or ontologies). An extension of RDF for defining ontologies is RDF Schema (RDFS [21]) which only can express conceptual modeling notions such as generalization between concepts (aka classes) and roles (aka properties). For properties, domain and range restrictions can be specified. Thus, from a description logic perspective the expressiveness of RDFS is quite limited.

A more expressive ontology representation language is DAML+OIL [11]. Basically, the logic behind DAML+OIL can be characterized with the logic $SHIQ(D_n)^-$, i.e., with some restrictions, DAML+OIL ontology documents can be automatically represented as T-boxes [3]. The RDF-Part of DAML+OIL documents which does not refer to schema knowledge can be translated to A-boxes (this can be seen as an abstraction w.r.t. meta-level reasoning but we cannot go into details here).

DAML+OIL also allows for individuals in concepts (and T-box axioms). For example, expressing the fact that all human stem from a single human called ADAM requires an individual in a concept (and a T-box):

$$HUMAN \sqsubseteq (\exists \textit{has-ancestor} \ \{ADAM\}).$$

Note that, individuals in concept term can only be approximated with the logic $SHIQ(D_n)^-$. Since the approximation does not impose severe limitations in practice, Racer can be used to reason about RDF/DAML+OIL representation structures. In addition to RDF/DAML+OIL, Racer also supports the (less expressive) OWL representation language [20].

Building and Processing Resources Specified in RDF or DAML+OIL

Agent programs can instruct Racer to read RDF/DAML+OIL documents from Web servers and pose queries. One proposal for a query language for RDF resources and for interconnecting inference engines is specified in the HTTP-based DIG standard (see [5, 6]). DIG provides an XML-based language for defining terminological knowledge, for defining knowledge about individuals, for specifying queries, and for receiving answers to queries. With its additional socket-based TCP interface, Racer extends the DIG standard by supporting many additional query and declaration possibilities [13]. APIs for Java and C++ are available.

In order to make description logics practical, graphical interfaces are of utmost importance for building ontologies. One powerful system that can be used for graphically building ontologies and maintaining DAML+OIL documents is OilEd [4]. OilEd is developed by Sean Bechhofer from Manchester University. Obeying the DIG protocol and using Racer (among others), OilEd can verify the consistency of concepts, find implicit generalization relationships in the domain model, determine the direct type(s) of individuals, etc.

Let us assume that the T-box introduced above is specified with OilEd. We assume that necessary and sufficient condition for a concept BROTHER is specified as follows by an OilEd user:

$$BROTHER \equiv MAN \sqcap (\exists \textit{has-sibling} \ HUMAN)$$

OilEd can employ the description logic system RACER to automatically detect that the concept BROTHER is subsumed by the concept UNLCE already defined in the knowledge base. Although syntactically not directly apparent in the axiom above, the reason is that parents are also humans (compare the axiom for UNCLE presented above).

Furthermore, if an ontology designer specifies

$$ECONOMIC\_TAXI \equiv TAXI \sqcap SMART$$

RACER reveals an inconsistency. Due to the example T-box considered here, there can be no economic taxis as specified by this axiom because Smarts have at most 2 seats whereas taxis require at least 4 (see above). Inconsistent concept names are usually due to modeling errors. Why should one give a (more or less) complex axiom for something inconsistent? Hence, it be comes clear that description logics in general, and Racer in particular, can be used for systematically building domain models.
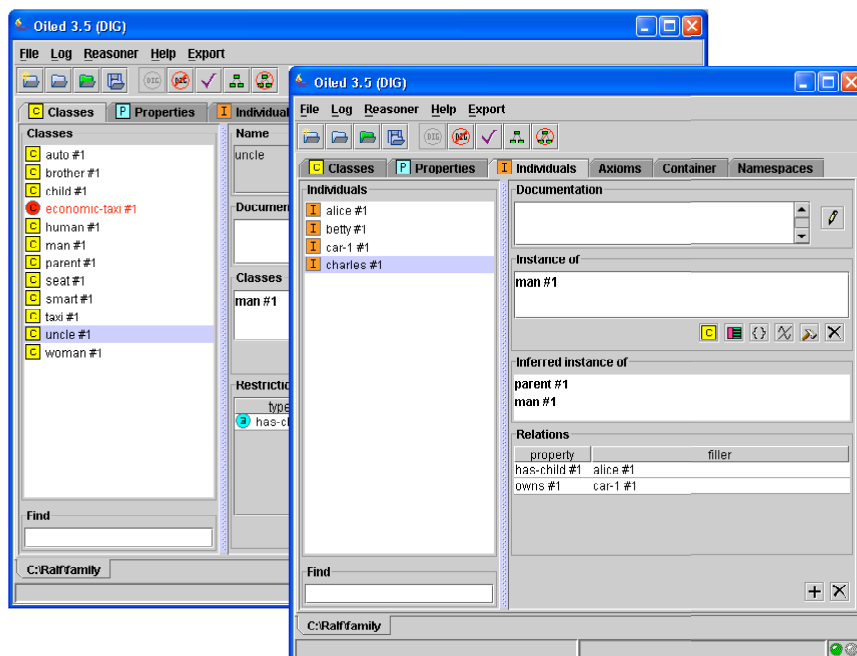

Figure 1: An example ontology defined with OilEd.

Another flexible graphical interface for managing ontologies with Racer is RICE (Racer Interactive Client Environment [10]). RICE is developed by Ronald Cornet, University of Amsterdam, Department of Medical Informatics. Based on the TCP interface of Racer, RICE provides full access to all inference services provided by Racer (see Figure 2).
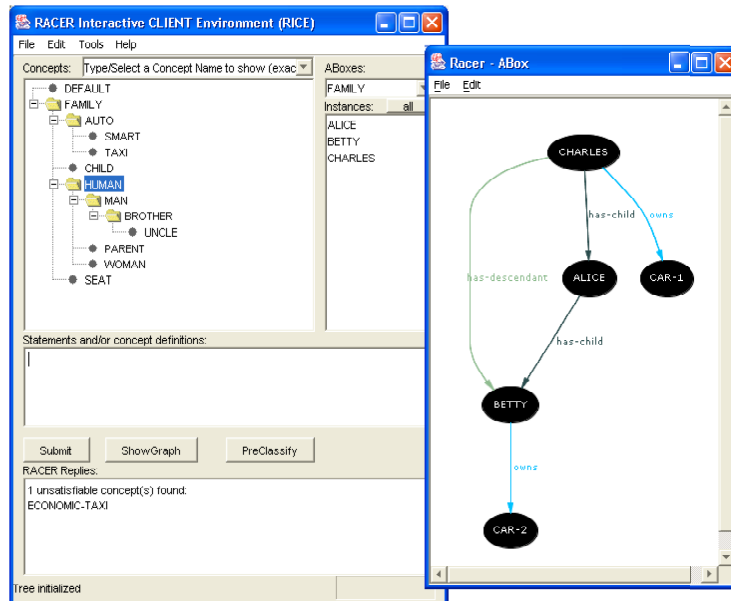
Figure 2: Snapshot of the RICE interface.

It should be clear that the use of description logics is not limited to system development time. At runtime, applications can also directly connect to a Racer Server and solve application problems using logical queries. This is discussed in the next section.

A Vision for Building Agent Systems with a Description Logic Inference Engine

In the following we consider an information retrieval scenario in which we assume that agents migrate to a specific agent host environment [22, p. 45] in order to retrieve information for a certain client. Based on certain application-specific epistemological decisions [22, p. 165] agents and host environments can be built using description logic inference services. We assume that information to be retrieved is stored in the host environment in the form of a description logic A-box. The background knowledge (conceptual data model) for retrieval tasks is given in the form of description logic T-boxes.

Within agent-based information retrieval framework, different scenarios are possible: First, the host environment can be tailored to a specific class of agents, i.e., the agents and the host environment share the same ontology (T-box). Second, and probably more challenging, information retrieval agents and host environments use different ontologies (T-boxes) and employ a third instance, a so-called ontology agent, to provide so-called inter-schema translation facilities.

Agents and Host Environment Share the Same Ontology

We assume that the T-box of the agent and the host environment consists of the axioms discussed above. In addition to the A-box presented above, we assume that the A-box (of the host environment) contains some additional assertions:

CHARLES : MAN
(CHARLES, ALICE) : has-child
(CHARLES, CAR-1) : owns
(BETTY, CAR-2) : owns

CAR-1 : MERCEDES
CAR-2 : BMW

A visualization of the complete structure of the A-box is shown in Figure 2. If the agent is responsible for finding possible people, who most probably would buy certain types of cars, it might pose an instance retrieval query to the A-box of our host environment as follows.

concept-instances(MAN $\sqcap$ ($\exists$ *has-child* PARENT) $\sqcap$ ($\exists$ *owns* (BMW $\sqcup$ MERCEDES))

The result of this retrieval query is {CHARLES} because CHARLES is a man which has a child which, in turn, is a parent. Note that BETTY is a human because of the assumed range restriction for *has-child*. In addition, CHARLES owns a MERCEDES car. We assume that the individuals in the result set of this query may be candidates for advertisement operations. In order to check this out, we assume that a specific test is carried out for CHARLES.

instance?(CHARLES, ($\exists$ *has-child* WOMAN ($\exists$ *has-child* WOMAN)) $\sqcup$

($\exists$ *has-descendant* ($\neg$WOMAN $\sqcap$ ($\exists$ *owns* BMW))))

Indeed, the answer is "yes". However, it is not quite obvious why this is the case. Let us consider the first disjunct. From what is specified in our example A-box, CHARLES has a child which is a woman (ALICE) and who also has a child (BETTY). But for BETTY it is not known whether the individual is an instance of WOMAN. Let us assume that BETTY is indeed a woman. In this case the first disjunct is true and, therefore the answer is yes. If, on the other hand, BETTY is assumed to be an instance of $\neg$WOMAN, the second disjunct becomes true because BETTY is a descendant of CHARLES, and BETTY owns a BWM.

Agents and Host Environment Use Different Ontologies

If an agent uses an ontology that is different from the ontology of a host environment visited by the agent, both ontologies must be related to each other. As an example we assume that the host environment uses the T-box defined above (let us call it T1), and the agent uses another T-box (T2). For instance, it might be the case that the agent uses concepts BMW_OWNER or MERCEDES_OWNER for a human who owns a BMW or Mercedes, respectively, rather than represent the same information with a role (relation) *owns* and an individual which is an instance of BMW or MERCEDES. The agent might ask

concept-instances(MAN $\sqcap$ ($\exists$ *has-child* PARENT) $\sqcap$

(BMW_OWNER $\sqcup$ MERCEDES_OWNER))

If the query is answered w.r.t. the A-box of the host environment and its own T-box (T1), then the retrieval result is the empty set because the names BMW_OWNER or MERCEDES_OWNER are not related to the names used in T1. In fact, in T1 there is no concept name at all for describing the situation of owning a BMW or a Mercedes. In T1 this is

expressed with ($\exists$ *owns* (BMW $\sqcup$ MERCEDES)). Hence, T1 uses a different style of conceptual modeling.

Since the host environment can check where the agent comes from we assume that the host environment can ask a so-called ontology broker for providing inter-schema knowledge [9].

Inter-schema knowledge related different ontologies and can be conveniently expressed using terminological axioms in some description logic. We assume that an inter-schema ontology broker can supplied the required axioms (possibly as a certain DAML+OIL resource that can be downloaded by the Racer Server of the host environment).

$$\text{BMW\_OWNER} \equiv (\exists \; owns \; \text{BMW})$$

$$\text{MERCEDES\_OWNER} \equiv (\exists \; owns \; \text{MERCEDES})$$

With these additional axioms, the agent gets the same answer as above. Due to space restrictions we cannot explain the use of inter-schema knowledge in detail and refer to [9] for further examples.

Accessing Inference Services

A standard information retrieval scenario used a document management system provided by an agent host environment. Documents can be annotated with metadata formalized with RDF. Information about documents can be represented using A-boxes (documents are represented by individuals) in the host environment. RDF annotations for documents are read by Racer and corresponding assertions are added to an A-box. Agents can retrieve documents by posing instance retrieval queries to these A-boxes w.r.t. to specific T-boxes in the way exemplified above. If we consider an instance retrieval query Q w.r.t. an A-box A, then it is clear that the solution set for Q could be extended if more information is added to A over time (whoever is responsible for that, another agent or the agent host environment). It would be a waste of resources to frequently poll the host environment with the same query (and repeated migration operations). Therefore, Racer supports the registration of queries at some server w.r.t. to some A-box (Publish/Subscribe Interface). With the registration, the agent specifies an IP address and a port number. The corresponding Racer Server passes a message to the agent if the solution set of a previously registered instance retrieval query is extended. The message specifies the new individuals found to be instances of the query concept Q. For details see the Racer manual [13].

Due to space restrictions, we can only give ideas for applications and services which can be implemented using logic. The examples we have given here should stimulate developers of agent systems and agent host environments to use the facilities of state of the art description logic inference engines.

Summary

This article motivated what description logics can do for Semantic Web applications. We sketched the design of agent systems using practical description logic inference engines. In the context of the Semantic Web a dedicated architecture that is integrated into the distributed systems context is required. In particular, the article presented important features of the Racer System, and described its architecture and background.

Large parts of DAML+OIL can be formalized using the description logic $SHIQ(D_n)^-$. However, Racer also goes beyond DAML+OIL in many respects. Whereas DAML+OIL offers only data type reasoning based on subsumption between names, Racer also provides means for true algebraic reasoning on information about "concrete values" which are associated with individuals. Depending on the domain and the language for defining

predicates over the domain, different so-called "concrete domains" are offered [12]. Racer supports interval reasoning over integers, inequations with order relations with respect to linear polynomials with real variables (with rational coefficients) or cardinal variables (with integer coefficients), and nonlinear multivariate non-linear polynomial (in)equations over complex numbers (Racer version 1-7-7).

Application demands and practical experiences with Racer demonstrated the importance of research on representation languages as well as query answering strategies to deal with realistic T-boxes and A-boxes that can be expected to be the basis for implementing future agent systems operating on the Semantic Web (see [14, 15] for encouraging evaluation results).

References

[1]   F. Baader, Logic-Based Knowledge Representation. In M.J. Wooldridge and M. Veloso, editors, *Artificial Intelligence Today, Recent Trends and Developments*, number 1600 in LNCS, pages 13-41. Springer Verlag, 1999.
[2]   F. Baader, et al. (Editors), The Description Logic Handbook: Theory, Implementation and Applications, Cambridge University Press, 2003.
[3]   F. Baader, I. Horrocks, and U. Sattler. Description Logics as Ontology Languages for the Semantic Web, In Dieter Hutter and Werner Stephan, editors, *Festschrift in honor of Jörg Siekmann*, LNAI. Springer-Verlag, 2003.
[4]   S. Bechhofer, I. Horrocks, C. Goble, R. Stevens, *OilEd: a Reason-able Ontology Editor for the Semantic Web*. Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence, September 19-21, Vienna. Springer-Verlag LNAI Vol. 2174, pp 396-408. 2001.
[5]   S. Bechhofer, The DIG Description Logic Interface DIG 1.1, http://dl-web.man.ac.uk/dig/ (visited Feb. 2003).
[6]   S. Bechhofer, R. Möller, P. Crowther, The DIG Description Logic Interfac*e*, Submitted to: International Description Logic Workshop – DL-2003, Rome.
[7]   T. Berners-Lee, J. Hendler, and O. Lassila, "*The semantic web*," Scientific American, May 2001.
[8]   D. Calvanese, D., M. Lenzerini, M, D. Nardi 1998. Description Logics for Conceptual Data Modeling. In Chomicki, J., and Saake, G., eds., *Logics for Databases and Information System*s. Kluwer Academic Publisher. 229–264.
[9]   D. Calvanese, G. De Giacomo, M. Lenzerini, Description Logics for Information Integration, In A. Kakas and F. Sadri, editors, *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski*, volume 2408 of *LNCS*, pages 41-60. Springer-Verlag, 2002.
[10]   R. Cornet, RICE is the Racer Interactive Client Environment, http://www.b1g-systems.com/ronald/rice/ (visited Feb. 2003).
[11]   DAML+OIL (March 2001) http://www.daml.org/2001/03/daml+oil-index.html (visited, Feb. 2003).
[12]   V. Haarslev, R. Möller, M. Wessel, The Description Logic ALCNHR+ Extended with Concrete Domains: A Practically Motivated Approach, In *Proc. International Joint Conference on Automated Reasoning, IJCAR'2001, June 18-23, 2001, Siena, Italy.*
[13]   V. Haarslev, R. Möller, Racer User's Guide and Manual, http://www.fh-wedel.de/~mo/racer/racer-manual-1-7-7.pdf (visited Feb. 2003).

[14] V. Haarslev, R. Möller, High Performance Reasoning with Very Large Knowledge Bases: A Practical Case Study, In *Proc. 7th Int. Joint Conf. on Artificial Intelligence, IJCAI-01*, August 4-10, 2001, Seattle, Washington, USA.

[15] V. Haarslev, R. Möller, Optimization Strategies for Instance Retrieval, In Proc. Int. Workshop on Description Logics DL-2002, Toulouse, 2002.

[16] V. Haarslev. R. Möller, Racer System Description, In *Proc. International Joint Conference on Automated Reasoning, IJCAR'2001, June 18-23, 2001, Siena, Italy.*

[17] I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic *SHIQ*. In David MacAllester, editor, *Proc. of the 17th Int. Conf. on Automated Deduction (CADE-17)*, number 1831 in Lecture Notes in Artificial Intelligence, pages 482-496. Springer-Verlag, 2000.

[18] O. Lassila and R. R. Swick. Resource Description Framework (RDF) Model and Syntax Specification. Recommendation, W3C, February 1999. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222 (visited Feb. 2003).

[19] S. McIlraith, T.C. Son and H. Zeng, Semantic Web Services, *IEEE Intelligent Systems, Special Issue on the Semantic Web*, Volume 16, No. 2, pp. 46-53, March/April, 2001.

[20] OWL Web Ontology Language 1.0 Reference, http://www.w3.org/TR/2002/WD-owl-ref-20020729/ (visited Feb. 2003).

[21] RDFS, http://www.w3.org/TR/2002/WD-rdf-schema-20020430/ (visited Feb. 2003).

[22] S. Russell, P. Norvig, Artificial Intelligence: A Modern Approach, Prentice Hall, 1995.