Racer: A Core Inference Engine for the Semantic Web

Volker Haarslev[†] and Ralf Möller[‡]

[†]Concordia University, Montreal, Canada (haarslev@cs.concordia.ca)
[‡]Technical University Hamburg-Harburg, Germany (ra.moeller@tu-harburg.de)

Abstract. In this paper we describe Racer, which can be considered as a core inference engine for the semantic web. The Racer inference server offers two APIs that are already used by at least three different network clients, i.e., the ontology editor OilEd, the visualization tool RICE, and the ontology development environment Protege 2. The Racer server supports the standard DIG protocol via HTTP and a TCP based protocol with extensive query facilities. Racer currently supports the web ontology languages DAML+OIL, RDF, and OWL.

1 Motivation

The Semantic Web initiative defines important challenges for knowledge representation and inference systems. Recently, several standards for representation languages have been proposed (RDF, DAML+OIL, OWL). One of the standards for the Semantic Web is the Resource Description Framework (RDF [12]). 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 [6]) 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, the expressiveness of RDFS is very limited. Much more expressive representation languages are DAML+OIL [15] and OWL [14]. Although still in a very weak way, based on XML-Schema, OWL and DAML+OIL also provide for means of dealing with data types known from programming languages.

The representation languages mentioned above are defined with a model-theoretic semantics. In particular, for the language OWL, a semantics was defined such that very large fragments of the language can be directly expressed using so-called description logics (see [1]). The fragment is called OWL DL.

With some restrictions that are discussed below one can state that the logical basis of OWL (or DAML+OIL) can be characterized with the description logic $\mathcal{SHIQ}(\mathcal{D}_n)^-$ [3] (DAML+OIL documents are to be interpreted in the spirit of OWL DL). This means, with some restrictions, OWL documents can be automatically translated to $\mathcal{SHIQ}(\mathcal{D}_n)^-$ T-boxes. The RDF-Part of OWL documents can be translated to $\mathcal{SHIQ}(\mathcal{D}_n)^-$ A-boxes.

In the remainder of this paper Racer, its APIs, and its inference services are briefly described. The use of Racer as network server is illustrated by RICE offering an interactive visualization and query interface. The paper is concluded by reporting on Racer's constraint-based data types support whose functionally exceeds the current OWL standard.

2 Racer: A Description Logic Inference Engine

The logic $\mathcal{SHIQ}(\mathcal{D}_n)^-$ is interesting for practical applications because highly optimized inference systems are available (e.g., Racer [8]). Racer is freely available for research purposes and can be accessed by standard HTTP or TCP protocols (the Racer program is subsequently also called Racer server). Racer can read DAML+OIL and OWL knowledge bases either from local files or from remote Web servers (i.e., a Racer server is also a HTTP client). In turn, other client programs that need inference services can communicate with a Racer server via TCP-based protocols. OilEd [4] can be seen as a specific client that uses the DIG protocol [5] for communicating with a Racer server, whereas RICE [13] is another client that uses a more low-level TCP protocol providing extensive query facilities (see below).

The DIG protocol is a an XML- and HTTP-based standard for connecting client programs to description logic inference engines. DIG allows for the allocation of knowledge bases and enables clients to pose standard description logic queries. The main ideas behind DIG are described in detail in [5]. As a standard and a least common denominator it cannot encompass all possible forms of system-specific statements and queries. Let alone long term query processing instructions (e.g., exploitation of query subsumption, computation of indexes for certain kinds of queries etc., see [9]). Therefore, Racer provides an additional TCP-based interface in order to send instructions (statements) and queries. For interactive use, the language supported by Racer is not XML- or RDF-based but is largely based on the KRSS standard with some additions and restrictions. The advantage is that users can spontaneously type queries which can be directly sent to a Racer server. We will see below that RICE can be used as a shell for Racer. However, the Racer TCP interface can be very easily accessed from Java or C++ application programs as well. For both languages corresponding APIs are available.

The following code fragment demonstrates how to interact with a Racer server from a Java application using Racer's TCP-based API. The aim of the example is to demonstrate the relative ease of use that such an API provides.

```
public class KillerApplication {
```

```
   public static void main(String[] argv) {
      RacerClient client=new RacerClient("racer.cs.concordia.ca", 8088);
      try {
         client.openConnection();
         try {
            String kbName=
              client.send("(owl-read-document
              \"http://www.cs.concordia.ca/~faculty/haarslev/family.owl\")");
            String queryResult=
                client.send("(individual-direct-types |#CHARLES|)");
            System.out.println(racerResult);
            }
         catch (RacerException e) {
            ...
            }
         }
         client.closeConnection();
      } catch (IOException e) {
         ...
      }
   }
}
```

The connection to the Racer server is represented with a client object (of class `RacerClient`). The client sends messages to a Racer server running on the machine with name `"racer.cs.concordia.ca"` on port 8088. The Java program can be run on another computer, of course. The program instructs the Racer server to load an OWL document from a remote server. In addition, the Java client program executes a query and prints the result set.

3 A Selection of Supported Inference Services

In description logic terminology, a tuple consisting of a T-box and an A-box is referred to as a knowledge base. An individual is a specific named object. OWL also allows for individuals in concepts (and T-box axioms). For example, expressing the fact that all humans stem from a single human called ADAM requires to refer to an individual in a concept (and a T-box). Only part of the expressivity of individuals mentioned in concepts can be captured with A-boxes. However, a straightforward approximation exists (see [10]) such that in practice suitable $\mathcal{SHIQ}(\mathcal{D}_n)^-$ ontologies can be generated from an OWL document. Racer can directly read OWL documents and represent them as description logic knowledge bases (aka ontologies). In the following a selection of supported queries is briefly introduced.

- 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.

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.
– Computation of the fillers of a role with reference to an individual.

Given the background of description logics, many application papers demonstrate how these inference services can be used to solve actual problems with DAML+OIL or OWL knowledge bases. The query interface is extensively used by RICE, which is briefly described in the next section.

4 RICE: Racer Interactive Client Environment

RICE [13] is a tool for Racer that visualizes taxonomies and A-box structures and enables users to interactively define queries using these visualizations. RICE is started as an application program and can be configured to connect to a Racer server by giving a host name and a port. When RICE connects to a Racer server it retrieves all T-boxes and displays them in an unfoldable tree view (in a similar way as OilEd [4] does).

In Figure 1 the taxonomy induced by the T-box specified above is presented (left window, unfoldable tree display). The taxonomy is accompanied by the pane for displaying A-box individuals (to the right of the tree display). Selecting a concept name in the taxonomy corresponds to posing an instance retrieval query with that concept name as a query concept. The result set is displayed in

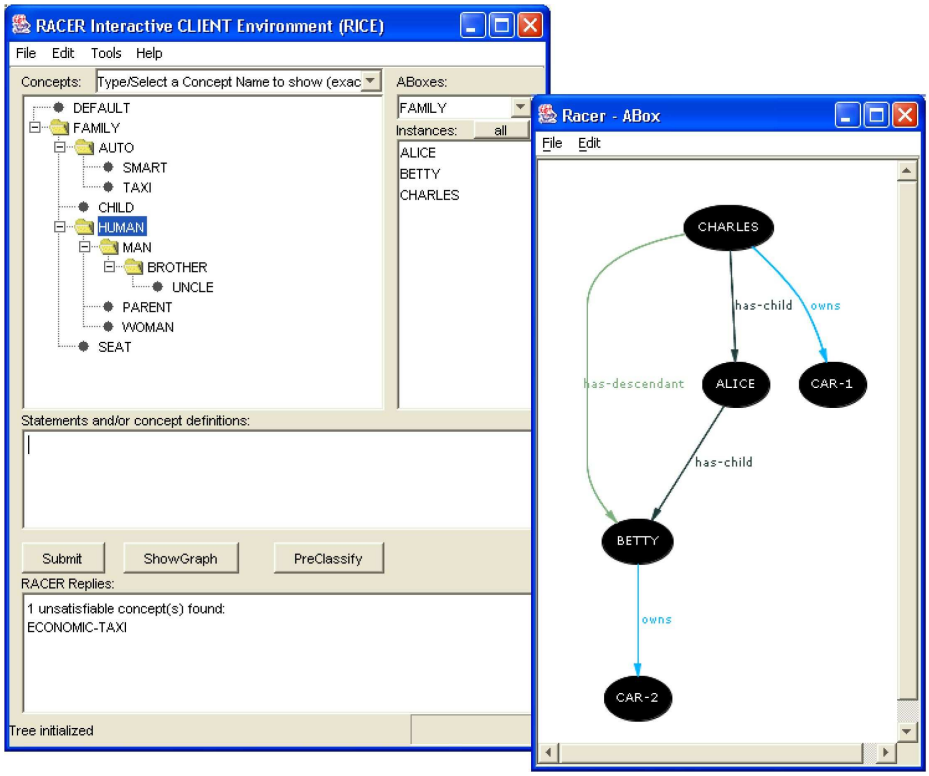

Fig. 1. Snapshot of RICE displaying an example knowledge base (T-box and A-box).

the instances pane. In the example in Figure 1 all humans are displayed. The structure of the whole A-box can be displayed by pressing the button "Show Graph". The graph window to the right appears. Clicking on individuals is interpreted as posing queries for the direct types of the individuals. In Figure 2 the individual $CHARLES$ is selected. The taxonomy is automatically unfolded such that all concept names which are direct types can be seen as highlighted nodes. Figure 2 also demonstrates that graphical attributes (e.g., color, shape) for displaying A-boxes can be (interactively) specified as appropriate.

RICE users can interactively type instructions and queries into the interaction pane in the middle. In Figure 2 we see an instance retrieval query. Query results are printed into the lower pane. Since Racer supports multiple A-boxes, users can interactively select the A-box subsequent queries should refer to (see the main window in Figure 2, top-right selection box). The current T-box can also be easily set by clicking on a T-box name.

If a user specifies a knowledge base with OilEd, Racer can be used to verify and classify it with a single click. The knowledge base is then known to the Racer server. If RICE connects to the Racer server, the knowledge base is visible. Note that OilEd and RICE can access a Racer server in parallel without any problems

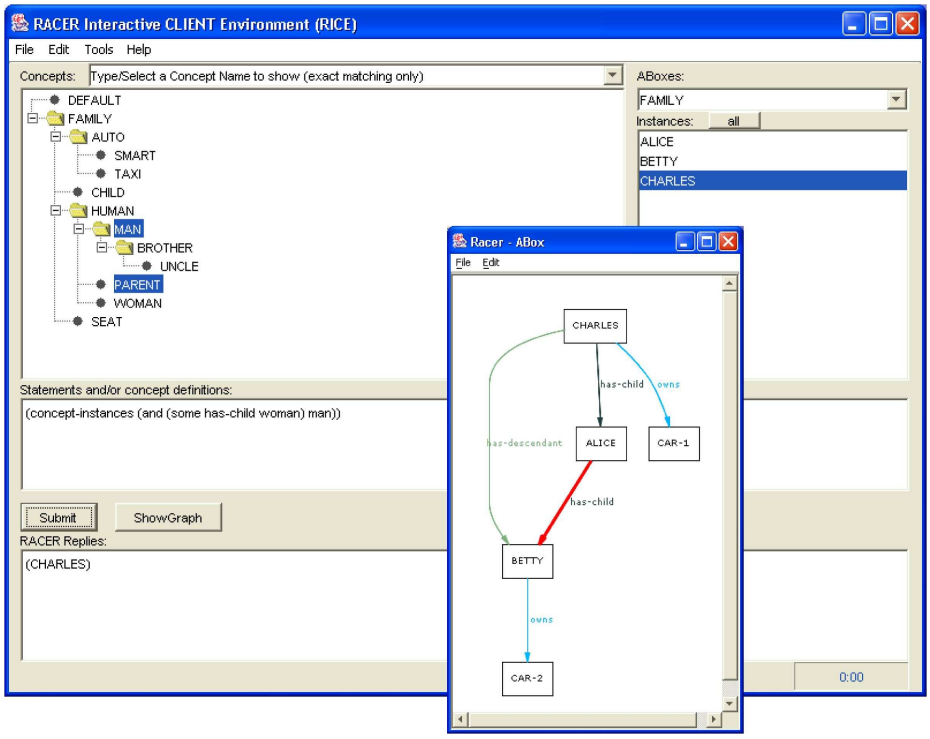

Fig. 2. Snapshot of RICE showing the results of a direct types query and an instance retrieval query.

if the Racer Proxy is installed appropriately (see [10]). If the RICE user selects the knowledge base stemming from OilEd (in Figure 3 we used one of the OilEd example files) and presses "Show Graph", the A-box part is shown in a graph display (see Figure 3).

As a summary, we compare OilEd and RICE. OilEd supports DIG, which makes it useful for more reasoners, but is limited to what DIG supports. Furthermore, OilEd provides a graphical means for displaying definitions of concepts and instances. This makes it easy to see what properties are defined and which ones are inherited. OilEd presents unsatisfiable concepts in the taxonomy, whereas they are not shown in RICE. RICE can connect to a Racer server that has already loaded a model, and retrieve its taxonomy (this is not supported by DIG). RICE can add individual DL statements to Racer (although this currently requires full classification of the model involved). RICE can be used to pose queries on Racer (either interactively or with a textual specification), and shows a graphical representation of relations in an A-box. RICE can also deal with multiple T-boxes and associated A-boxes. In particular, it can show instances of a concept and concepts (direct types) of instances.

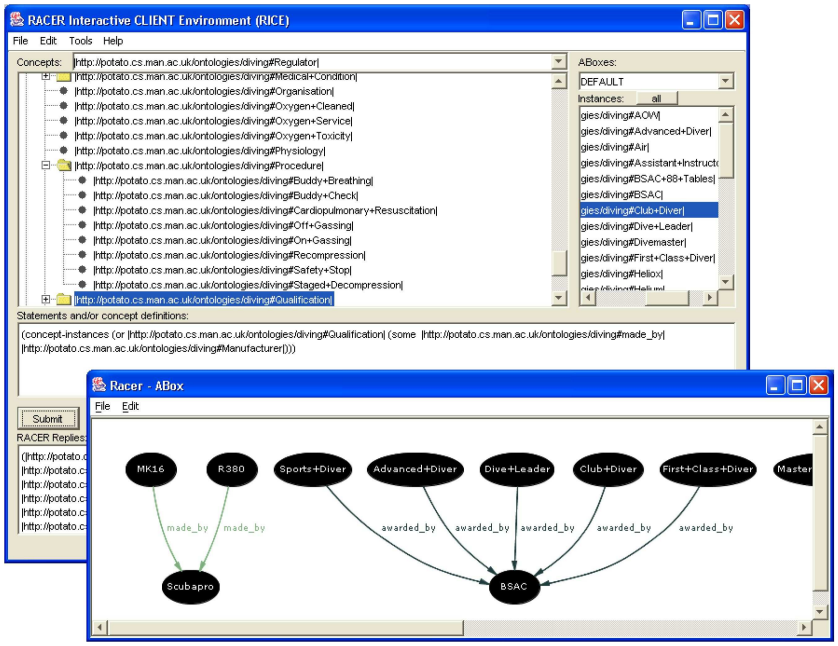

Fig. 3. Using RICE to visualize a RDF document interactively defined with OilEd.

5 Reasoning Beyond OWL: Constraints on Data Types

For various practical reasons OWL also includes so-called data types based on XML-Schema. Data types in XML-Schema are inspired by a storage-oriented characterization of values and are taken from programming languages. For instance, data types encompass `integer`, `short`, `long`, `boolean`, `string` as well as various kinds of specializations for strings.

For an ontology representation language, a semantic characterization for data types might have been more appropriate in our opinion. Thus natural numbers, integers, reals, or complex numbers might have been selected as data types rather than `long` or `short` etc. because for knowledge representation languages the storage format should not be of top-most concern.

Based on XML-Schema in DAML+OIL or OWL it is possible to specify subtypes of, for instance, `integer` by defining a minimum or maximum value [15]. However, OWL does not support so-called constraints between data values. In many practical applications, for instance, linear polynomial inequations with order relations are appropriate. In description logics and databases, these kinds of constraints have a long tradition (see [1, 11]). In the following we will adopt the description logic perspective: concrete domains [2].

Racer supports concrete domain reasoning over natural numbers ($\mathbb{N}$), integers ($\mathbb{Z}$), reals ($\mathbb{R}$), complex numbers ($\mathbb{C}$), and strings. For different sets, different kinds of predicates are supported:

– $\mathbb{N}$: linear inequations with order constraints and integer coefficients
– $\mathbb{Z}$: interval constraints
– $\mathbb{R}$: linear inequations with order constraints and rational coefficients
– $\mathbb{C}$: nonlinear multivariate inequations with integer coefficients
– Strings: equality and inequality

For convenience, rational coefficients can be specified in floating point notation. They are automatically transformed into their rational equivalents (e.g., 0.75 is transformed into $\frac{3}{4}$). In the following we will use the names on the left-hand side of the table to refer to the corresponding concrete domains.

The following example uses the concrete domains $\mathbb{Z}$ and $\mathbb{R}$. For sake of brevity, we use Racer's Lisp syntax [10].

```
(in-tbox family)
(signature
   :atomic-concepts (... teenager)
   :roles (...)
   :attributes ((integer age)))
...
(equivalent teenager (and human (min age 16)))
(equivalent old-teenager (and human (min age 18)))
```

Asking for the children of teenager reveals that `old-teenager` is a `teenager`. A further extensions demonstrates the usage of reals as concrete domain.

```
(signature
   :atomic-concepts (... teenager)
   :roles (...)
   :attributes ((integer age)
                (real temperature-celsius)
                (real temperature-fahrenheit)))
...
(equivalent teenager (and human (min age 16)))
(equivalent old-teenager (and human (min age 18)))
(equivalent human-with-feaver (and human (>= temperature-celsius 38.5))
(equivalent seriously-ill-human (and human (>= temperature-celsius 42.0)))
```

Obviously, Racer determines that the concept `seriously-ill-human` is subsumed by `human-with-fever`. For the Reals, Racer supports linear equations and inequations. Thus, we could add the following statement to the knowledge base in order to ensure the proper relationship between the two attributes `temperature-fahrenheit` and `temperature-celsius`.

```
(implies top (= temperature-fahrenheit
                (+ (* 1.8 temperature-celsius) 32)))
```

If a concept `seriously-ill-human-1` is defined as

```
(equivalent seriously-ill-human-1
            (and human (>= temperature-fahrenheit 107.6)))
```

Racer recognizes the subsumption relationship with `human-with-fever` and the synonym relationship with `seriously-ill-human`.

In an A-box, it is possible to set up constraints between single individuals. This is illustrated with the following examples.

```
(signature
   :atomic-concepts (... teenager)
   :roles (...)
   :attributes (...)
   :individuals (eve doris)
   :objects (temp-eve temp-doris))
...
(constrained eve temp-eve temperature-fahrenheit)
(constrained doris temp-doris temperature-celsius)
(constraints
   (= temp-eve 102.56)
   (= temp-doris 39.5))
```

For instance, this states that the individual `eve` is related via the attribute `temperature-fahrenheit` to the object `temp-eve`. The constraint `(= temp-eve 102.56)` specifies that the object `temp-eve` is equal to 102.56.

Now, asking for the direct types of eve and doris reveals that both individuals are instances of `human-with-fever`. In the following A-box there is an inconsistency since the temperature of 102.56 Fahrenheit is identical with 39.5 Celsius.

```
(constrained eve temp-eve temperature-fahrenheit)
(constrained doris temp-doris temperature-celsius)
(constraints
   (= temp-eve 102.56)
   (= temp-doris 39.5)
   (> temp-eve temp-doris))
```

An additional kind of query is possible for concrete domains: Check if certain concrete domain constraints are entailed by an A-box and a T-box. For instance, in the above-mentioned example, the following query returns true.

```
(constraint-entailed? (= temp-eve temp-doris))
```

6 Conclusion

This paper briefly described Racer and demonstrated that Racer can cooperate with various kinds of ontology editors and visualization tools. Racer can be considered as one of the fastest OWL DL reasoners based on sound and complete algorithms that is currently freely available. It is still unique in its highly optimized reasoning support for A-boxes and constraint-based data types (as demonstrated in the previous sections). Racer also includes optimization techniques supporting the classification of very large knowledge bases (KBs). For

instance, a set of KBs could be classified in a few hours [7] that were derived from the Unified Medical Language System (UMLS) and contain up to 200,000 concept introduction axioms (OWL classes) and up to 50,000 hierarchical roles (OWL object properties).

Acknowledgements

We gratefully acknowledge the work of Ronald Cornet, who developed RICE.

References

1. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, 2002. In print.
2. F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Twelfth International Joint Conference on Artificial Intelligence, Darling Harbour, Sydney, Australia, Aug. 24-30, 1991*, pages 452–457, August 1991.
3. F. Baader, I. Horrocks, and U. Sattler. Description logics as ontology languages for the semantic web. In D. Hutter and W. Stephan, editors, *Festschrift in honor of Jörg Siekmann*. LNAI. Springer-Verlag, 2003.
4. S. Bechhofer, I. Horrocks, and C. Goble. OilEd: a reason-able ontology editor for the semantic web. In *Proceedings of KI2001, Joint German/Austrian conference on Artificial Intelligence, September 19-21, Vienna*. LNAI Vol. 2174, Springer-Verlag, 2001.
5. S. Bechhofer, R. Möller, and P. Crowther. The DIG description interface. In *Proc. International Workshop on Description Logics – DL'03*, 2003.
6. D. Brickley and R.V. Guha. RDF vocabulary description language 1.0: RDF Schema, http://www.w3.org/tr/2002/wd-rdf-schema-20020430/, 2002.
7. V. Haarslev and R. Möller. High performance reasoning with very large knowledge bases: A practical case study. In B. Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI-01, August 4-10, 2001, Seattle, Washington, USA*, pages 161–166, August 2001.
8. V. Haarslev and R. Möller. Racer system description. In *International Joint Conference on Automated Reasoning, IJCAR'2001, June 18-23, 2001, Siena, Italy.*, 2001.
9. V. Haarslev and R. Möller. Optimization stategies for instance retrieval. In *Proc. International Workshop on Description Logics – DL'02*, 2002.
10. V. Haarslev and R. Möller. The Racer user's guide and reference manual, 2003.
11. G. Kuper, L. Libkin, and J. Paredaens (Eds.). *Constraint Databases*. Springer-Verlag, 1998.
12. 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, 1999.
13. R. Möller, R. Cornet, and V. Haarslev. Graphical interfaces for Racer: querying DAML+OIL and RDF documents. In *Proc. International Workshop on Description Logics – DL'03*, 2003.
14. F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL web ontology language reference, 2003.
15. F. van Harmelen, P.F. Patel-Schneider, and I. Horrocks (Editors). Reference description of the DAML+OIL (march 2001) ontology markup language, 2001.