Graphical Interfaces for Racer: Querying DAML+OIL and RDF documents

Ralf Möller[†], Ronald Cornet[‡], and Volker Haarslev[§]

[†]University of Applied Sciences, Wedel
[‡]Academic Medical Center – University of Amsterdam
[§]Concordia University, Montreal

Abstract

In this paper, we introduce RICE, a graphical application for interacting with the description logic inference server Racer. Comparing RICE with OilEd, we address the problem of visualizing and querying A-boxes w.r.t. predefined T-boxes. We discuss examples with T-boxes and A-boxes that are derived from DAML+OIL and RDF documents, respectively. Thus, the visualization tools discussed in this paper also apply to semantic web representation languages.

1 Motivation

Graphical interfaces are very important for practical work with description logic (DL) inference systems. However, very few graphical interfaces are available for existing DL systems. A notable exception is OilEd [1]. With tools such as OilEd, ontologies can be interactively built and they can be stored, for example, as DAML+OIL documents [7]. In addition, OilEd can be used to develop RDF documents for describing information about individuals with respect to DAML+OIL ontologies. Applications using these RDF and DAML+OIL documents require an inference engine that supports reasoning about individuals. Indeed, OilEd can be configured to use Racer [5] as an inference engine for classifying ontologies and for answering queries about individuals.

In this paper, we introduce another graphical interface for the description logic inference system Racer. The interface is called Racer Interactive Client Environment (RICE). RICE complements ontology building tools such as OilEd and, for instance, addresses the problem of querying A-boxes w.r.t. predefined T-boxes. In addition, A-box structures maintained by a Racer Server can be visualized using RICE. We do not claim that there is something entirely new about the techniques we describe. The goal of the paper is to demonstrate that state-of-the-art description logic systems freely available for research purposes no longer suffer from missing graphical interfaces, and missing integration into industry-based distributed system infrastructures.

2 RDF, DAML+OIL, and OWL

State-of-the-art description logic systems such as, for instance, Racer allow for interpreting DAML+OIL ontology documents as T-boxes and RDF data descriptions as

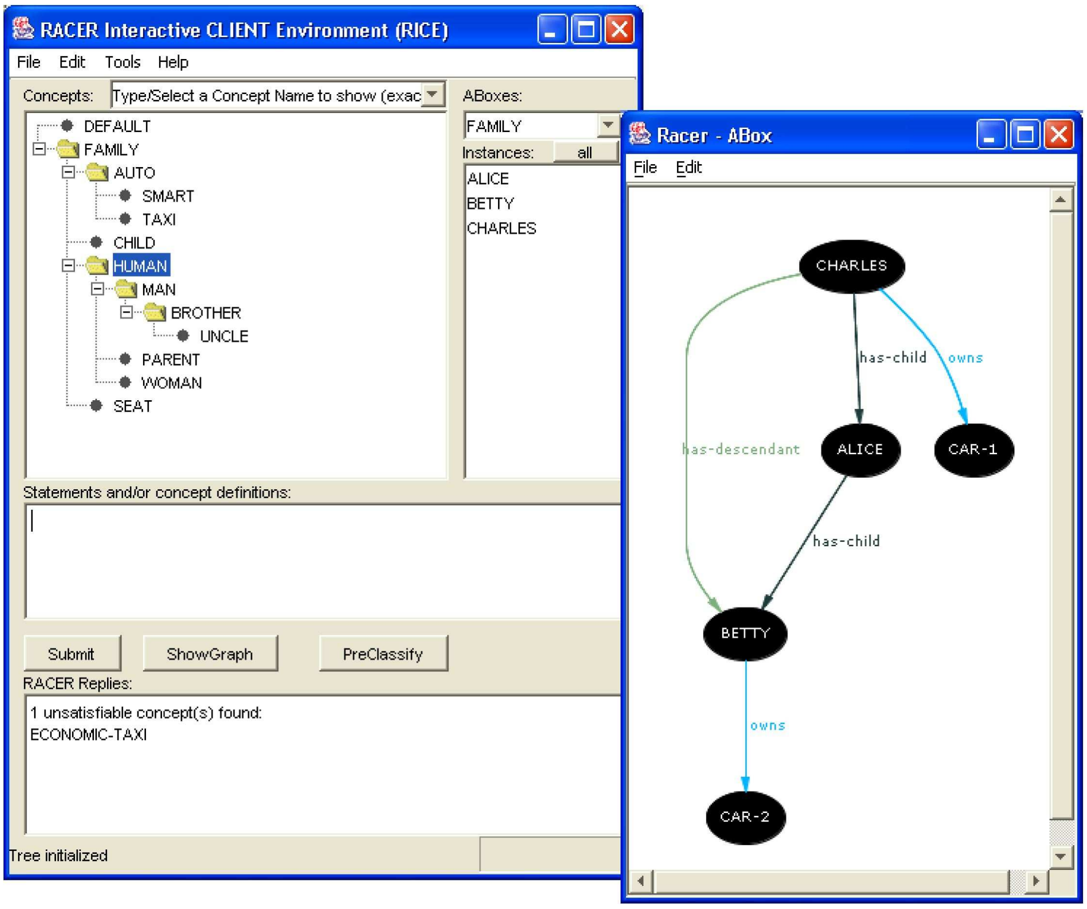
A-boxes. Racer also accepts the so-called OWL DL subset [6] with some additional restrictions (such as approximated reasoning for nominals (see [4] for details), unique name assumption, and role equality restricted to declare inverses). DAML+OIL documents are interpreted with the same restrictions as manifested in OWL DL (the sets of classes and instances are disjoint, no reified statements, no treatment of class meta-objects etc.). Descriptions in RDF documents (with OWL DL restrictions) are represented as A-boxes by the Racer System (for details see the Racer User's Guide [4]). Viewing RDF documents as A-boxes provides for a query language with the means of description logic inference systems. Furthermore, powerful graphical interfaces for description logic inference systems can be used to inspect RDF documents wrt. DAML+OIL or OWL ontologies. Before we discuss this with some examples, a more detailed introduction to the Racer system is appropriate.

3 The Racer Server and Racer Proxy

The Racer Server is an application program which can read DAML+OIL and OWL knowledge bases either from local files or from remote Web servers (i.e., a Racer Server is also an HTTP client). In turn, other client programs that need inference services can communicate with a Racer Server via TCP-based protocols. OilEd can be seen as a specific client that uses the DIG protocol [2] for communicating with a Racer Server, whereas RICE is another client that uses a more low-level TCP protocol.

The DIG protocol is a XML- and HTTP-based standard for connecting client programs to description logic inference engines (Fact or Racer right now). 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 [2]. 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 [3]). 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.

In the context of multiple graphical interfaces and client programs (e.g., agents), a DL server such as Racer will be used by more than one client. For dealing with multiple clients, the Racer Server includes a subsystem called the Racer Proxy. The Racer Proxy is started as a front-end to an associated Racer Server. It is configured to use a port number for external client access in the same way as a Racer Server. The port number of the associated Racer Server must be specified at proxy startup time. Then, a Racer Proxy is accessed just like a Racer Server (see above), and it just forwards requests to a corresponding Racer Server. The task of the Racer Proxy is to provide locks for inference services of the Racer Server(s) that it "manages" such that instructions and queries of multiple clients are properly coordinated.



```
1 unsatisfiable concept(s) found:
ECONOMIC-TAXI
```

Figure 1: Snapshot of the RICE interface displaying the example knowledge base.

4 RICE

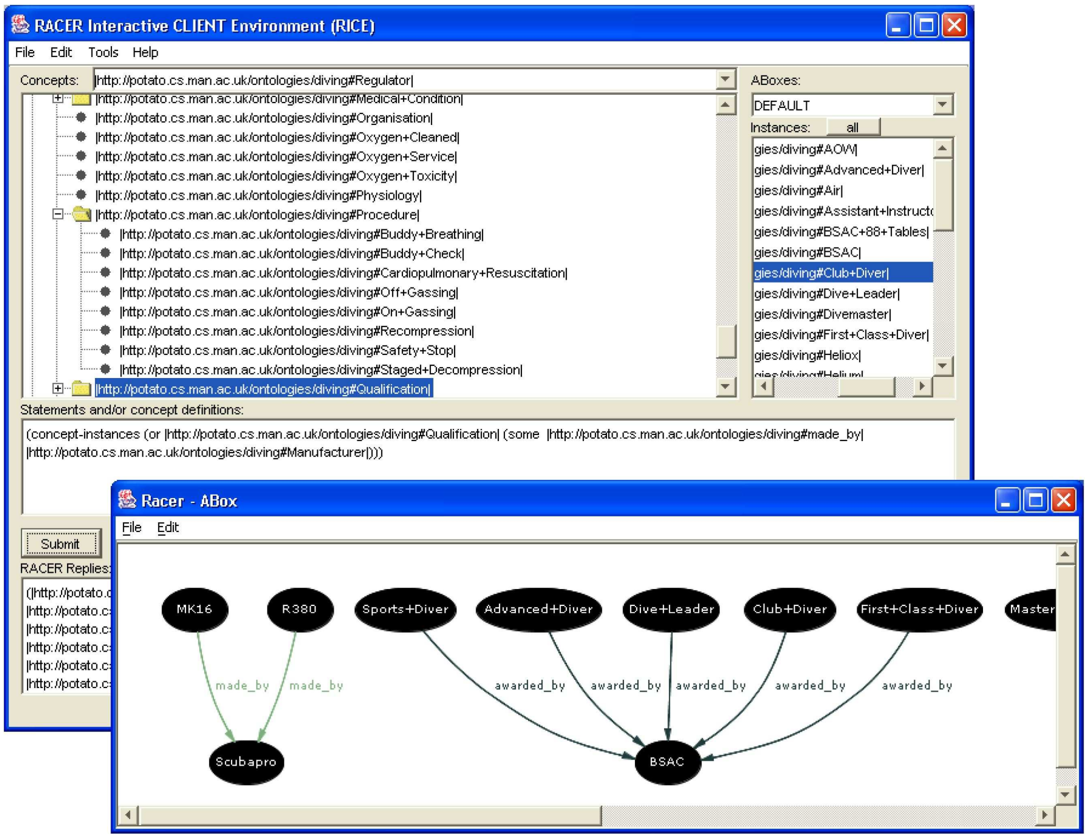
RICE visualizes taxonomies and A-box structures and enables users to interactively define queries using these visualizations. RICE is started as an application program and can be configured to connect to a Racer Server by giving a host name and a port. When RICE connects to a Racer Server it retrieves all T-boxes and displays them in an unfoldable tree view (in a similar way as OilEd does). In the next sections we assume that the following T-box and A-box have been classified by Racer.

$transitive(has\text{-}descendant)$, $domain(has\text{-}child, PARENT)$,
$range(has\text{-}child, CHILD)$, $has\text{-}child \sqsubseteq has\text{-}descendant$,
$MAN \sqsubseteq HUMAN$, $WOMAN \sqsubseteq HUMAN$, $MAN \sqsubseteq \neg WOMAN$,
$TAXI \sqsubseteq Auto \sqcap (\geq 4\, has\text{-}seats.SEAT)$, $SMART \sqsubseteq Auto \sqcap (\leq 2\, has\text{-}seats.SEAT)$,
$PARENT \equiv HUMAN \sqcap \exists has\text{-}child.HUMAN$, $UNLCE \equiv MAN \sqcap \exists has\text{-}sibling.PARENT$,
$BROTHER \equiv MAN \sqcap \exists has\text{-}sibling.HUMAN$, $ECONOMIC\text{-}TAXI \equiv TAXI \sqcap SMART$

$CHARLES : MAN$, $(CHARLES, ALICE) : \textit{has-child}$, $ALICE : WOMAN$, $(ALICE, BETTY) : \textit{has-child}$, $(CHARLES, CAR\text{-}1) : \textit{owns}$, $(BETTY, CAR\text{-}2) : \textit{owns}$

4.1 Querying Knowledge Bases Graphically

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 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 (compare with the assertions given in Section 1). Clicking on individuals is interpreted as posing queries for the direct types of the individuals. The taxonomy is automatically unfolded such that all concept names which are direct types can be seen as highlighted nodes. Graphical attributes (e.g., color, shape) for displaying A-boxes can be (interactively) specified as appropriate.



|http://potato.cs.man.ac.uk/ontologies/diving#Regulator|

```
|http://potato.cs.man.ac.uk/ontologies/diving#Medical+Condition|
|http://potato.cs.man.ac.uk/ontologies/diving#Organisation|
|http://potato.cs.man.ac.uk/ontologies/diving#Oxygen+Cleaned|
|http://potato.cs.man.ac.uk/ontologies/diving#Oxygen+Service|
|http://potato.cs.man.ac.uk/ontologies/diving#Oxygen+Toxicity|
|http://potato.cs.man.ac.uk/ontologies/diving#Physiology|
|http://potato.cs.man.ac.uk/ontologies/diving#Procedure|
    |http://potato.cs.man.ac.uk/ontologies/diving#Buddy+Breathing|
    |http://potato.cs.man.ac.uk/ontologies/diving#Buddy+Check|
    |http://potato.cs.man.ac.uk/ontologies/diving#Cardiopulmonary+Resuscitation|
    |http://potato.cs.man.ac.uk/ontologies/diving#Off+Gassing|
    |http://potato.cs.man.ac.uk/ontologies/diving#On+Gassing|
    |http://potato.cs.man.ac.uk/ontologies/diving#Recompression|
    |http://potato.cs.man.ac.uk/ontologies/diving#Safety+Stop|
    |http://potato.cs.man.ac.uk/ontologies/diving#Staged+Decompression|
|http://potato.cs.man.ac.uk/ontologies/diving#Qualification|
```

```
gies/diving#AOW|
gies/diving#Advanced+Diver|
gies/diving#Air|
gies/diving#Assistant+Instructo
gies/diving#BSAC+88+Tables|
gies/diving#BSAC|
gies/diving#Club+Diver|
gies/diving#Dive+Leader|
gies/diving#Divemaster|
gies/diving#First+Class+Diver|
gies/diving#Heliox|
```

```
(concept-instances (or |http://potato.cs.man.ac.uk/ontologies/diving#Qualification| (some  |http://potato.cs.man.ac.uk/ontologies/diving#made_by|
|http://potato.cs.man.ac.uk/ontologies/diving#Manufacturer|)))
```

```
(|http://potato.c
|http://potato.c
|http://potato.c
|http://potato.c
|http://potato.c
|http://potato.c
```

Figure 2: Using RICE on the fly to visualize an RDF document interactively defined with OilEd.

4.2 Using OilEd and RICE in combination

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 Sever. 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 if the Racer Proxy is installed appropriately (see above). If the RICE user selects the knowledge base stemming from OilEd (in Figure 2 we used one of the OilEd example files) and presses "Show Graph", the A-box part is shown in a graph display (see Figure 2).

As a summary, we compare OilEd and RICE. OilEd supports DIG, which makes it useful for more reasoners (e.g., FaCT), 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.

5 Conclusion

The paper demonstrates that state-of-the-art description logic systems come with various kinds of graphical editors and interaction tools which can effectively cooperate. Description logic systems freely available for research purposes now provide for industry-oriented software integration as can be expected in the context of the semantic web with its XML-based representation languages (RDF, DAML+OIL, OWL).

References

[1] 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.

[2] S. Bechhofer, R. Möller, and P. Crowther. The DIG description interface. In *Proc. International Workshop on Description Logics – DL'03*, 2003.

[3] V. Haarslev and R. Möller. Optimization stategies for instance retrieval. In *Proc. International Workshop on Description Logics – DL'02*, 2002.

[4] V. Haarslev and R. Möller. The Racer user's guide and reference manual, 2003.

[5] Volker Haarslev and Ralf Möller. RACER system description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, 2001.

[6] F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. OWL web ontology language reference, 2003.

[7] F. van Harmelen, P.F. Patel-Schneider, and I. Horrocks (Editors). Reference description of the DAML+OIL (march 2001) ontology markup language, 2001.