Description Logic Inference Technology: Lessions Learned in the Trenches

Volker Haarslev[†] Ralf Möller[‡] and Michael Wessel[‡]

[†]Concordia University, Montreal
[‡]Hamburg University of Technology

Abstract

Optimized description logic systems are now available for quite a long time. Whereas initially, to a large extent only T-box reasoning was used in applications, now more and more applications also rely on A-box reasoning. In this article we summarize our experiences with the description logic reasoner Racer and perform an evaluation of the system with respect to instance retrieval benchmarks. In addition, we report on our experiences with two years of user support for OWL knowledge base development and usage. The article provides an overview over the state of the art in description logic inference technology and derives suggestions for future developments.

1 Introduction

Now as description logic systems such as Racer are used in more and more applications we try to achieve a better understanding about how reasoning facilities are exploited for problems that applications solve. Understanding how description logic systems are actually used is important because, due to our experiences, the software architecture must be tailored to practical usage scenarios in order to achieve adequate performance. On the other hand, facilities offered by description logic inference sytems must be adequately used in applications to guarantee that unnecessary overhead can be avoided right from the beginning.

In this article we summarize our experiences with the description logic reasoner Racer and perform an evaluation of the system with respect to instance retrieval benchmarks with reference to expressive query languages. Instance retrieval requires many optimization techniques and provides a good estimate for the overall performance of a DL reasoner. The article provides an overview over the state of the art in description logic inference technology and derives suggestions for future developments of description logic systems. In addition, we report on our experiences resulting from two years of user support for OWL knowledge base development and usage. This might be of interest for developers whose applications rely on OWL inference technology. Racer covers OWL DL with minimum restrictions (nominals are approximated with standard techniques).

2 Benchmarking nRQL

Racer offers an expressive query language (nRQL, new Racer Query Language, pronounce "nercle") for over a year now (see, e.g., [7]). This language has been used in many application projects since then. In the same spirit as early description logic systems (see, e.g., [10]), nRQL uses the so-called active domain semantics for query answering, i.e., variables in queries are bound to explicitly mentioned individuals in the A-box (and not to arbitrary elements of the universe). With the active domain semantics the nRQL language provides at least the expressivity of nonrecursive datalog with negation (or relational algebra). In addition, binary predicates (roles) can be declared as transitive in the T-box and inverses of roles can be named appropriately if required. The current Racer system (RacerPro 1.8, http://www.racer-systems.com) provides an optimized implementation for this language in the context of description logics. It is obvious that with active domain semantics the expressivity could be extended to disjunctive datalog (with recursion, negation and stratification) in the future without loosing decidability. Note, however, that with the availability of transitive roles in the description logic, recursive datalog rules in the query part are less frequently required in practice and nonrecursive queries are much easier to optimize effectively.

Many users have asked for an overview about the performance of nRQL with respect to practical usage scenarios. In the following we apply Racer to two kinds of benchmarks (Sunfire V880 with 8 64bit processors, 1GHz each, 32GB total).

2.1 Lehigh University Benchmark (LUBM)

The so-called Lehigh University Benchmark (LUBM, [2, 3]) was developed to facilitate the evaluation and comparison of OWL semantic web repositories. An older version of Racer was already successfully evaluated for this kind of benchmark [12]. We add to this work the results for the latest Racer version (RacerPro 1.8), which provides for two different inference modes that allow DL systems to be used in many more practical applications contexts than before.

The LUBM consists of an OWL ontology for modeling universities; i.e., the ontology contains concepts for persons, student, professors, publications, courses etc. as well as appropriate relationships for such a universe of discourse. A benchmark generator written in Java is capable of generating extensional data corresponding to this ontology; i.e., a set of descriptions for specific departments, professors, students, courses, and so on can be generated. Furthermore, in [2, 3] a set of 14 benchmarking queries is defined, ranging from simple queries that can be answered using plain relational look-up techniques to more complicated queries which require dense OWL reasoning techniques in order to be answered in a complete mode. The LUBM queries are conjunctive queries referencing concept and role names from the T-box. All queries can be expressed with nRQL. Please refer to [2, 3] for more information about the queries. Below, LUBM queries 9 and 12 are shown in order to present a flavor of the kind of query answering problems – note that www.University0.edu is an individual, and subOrganizationOf is a transitive role.

```
Q9: (retrieve (?x ?y ?z)
        (and (?x Student) (?y Faculty) (?z Course)
             (?x ?y advisor) (?x ?z takesCourse)
```

```
              (?y ?z teacherOf)))

Q12: (retrieve (?x ?y www.University0.edu)
        (and (?x Chair)
              (?y Department)
              (?x ?y memberOf)
              (?y www.University0.edu subOrganizationOf)))
```

DLDB [2] is a relational database system (Microsoft Access) augmented with a DL-based query rewriting engine. Query rewriting is necessary to cope with (implicit) subsumption relationships between concept names due to T-box axioms. For instance, if in the database a certain individual is declared as a Professor, it will be found to be an instance of the superconcept Faculty as well (see query 9). In [2, 3], the DLDB system is benchmarked according to LUBM and it is observed that result sets for queries 11, 12 and 13 are not complete with the current DLDB implementation, i.e. in its current state of development, DLDB is an incomplete OWL semantic web repository. The main reason for incompleteness is that neither inverse roles or transitive roles nor concept definitions are considered by DLDB. In query 12, for instance, a predicate Chair is mentioned (see above). For the concept Chair there exists a definition in the T-box (a Chair is a Professor who is the HEAD of a Department). There is no individual directly declared to be an instance of Chair or one of its subconcepts. It is the relation HEAD to a Department that matters in this case. Chair is just one example that demonstrates a source of incompleteness in DLDB that occurs if T-boxes are present. Depending on the application, incompleteness may or may not be a problem.

Racer 1.8 can be used in a way that is similar to DLDB, which means that nRQL queries are answered with optimized algorithms known from relational databases with the addition of handling implicit subsumption relationships and inverse as well as transitive roles (answers computed this way are called "cheap" answers). In contrast to this, in the complete mode, Racer uses a tableau-based A-box satisfiability prover (for solving refutation proof problems) [5]. The tableau-based prover is necessary despite all optimization techniques that have been published and integrated into Racer.

We now compare RacerPro 1.8 with the performance of the DLDB system as reported in [3]. In particular, we distinguish between load times, preparation times (times for computing index structures), and query answering times. The performance of Racer is evaluated in two modes. In mode A only "cheap" answers are retrieved, in mode B (the complete mode) all answers are derived. Note that neither mode A nor mode B requires a process that is known as A-box realization in the literature (much too expensive for all benchmarks discussed in this article). Furthermore, in mode A no A-box consistency test is performed.

Univs	Load	Prep.	Inds	DLDB Size (MB)	Approx. Racer Size (MB)
1	24	56	17174	16	220
10	1065	7065	207426	184	1500
20	5300	42279	437555	388	3500

Table 1: Runtimes (in secs) for loading and preparing the tests in mode A.

In Table 1 statistics for 1, 10, and 20 universities are indicated. It can be seen that load and preparation times must not be underestimated as more individuals

Univs	Q9 (DLDB)	Q9 (Racer)	Q12 (DLDB)	Q12 (Racer)
1	0.6	2.6	0.062	0.2
10	20	33	0.1	0.3
20	57	66	0.3	0.5

Table 2: Runtimes (in secs) for instance retrieval in mode A.

have to be considered. Table 2 indicates that query execution times for Racer in mode A are comparable with DLDB (DLDB can handle larger datasets, though).

Univs	Preparation	Q9	Q12
1	2258	4.6	1011

Table 3: Runtimes (in secs) for instance retrieval in mode B.

In mode B, the complete mode, one university can be handled by Racer (Table 3). In contrast to DLDB, Racer runs in main memory, which is a clear disadvantage if huge amounts of data are to be processed. In fact, in [3] up to 50 universities are considered. This is beyond the scope of Racer and other description logic engines (reasoning requires dealing with role assertions).

Approaches such as DLDB completely fail if completeness matters. Note that even for information retrieval scenarios "expensive" answers might be "interesting" and relevant answers. For query 12, DLDB as well as Racer in mode A return 0 results whereas Racer mode B returns all results (15 chairs) since in the given LUBM benchmark, university 1 consists of 15 departments, each associated with a Professor that is the HEAD.

In mode A, Racer finds more solutions than DLDB (due to handling inverse and transitive roles appropriately even in mode A). Runtimes of Racer are within the range of DLDB. Thus, it is shown that it is possible to build a DL system that is comparable with database systems w.r.t. query answering times, if memory requirements and transactions are no issue. It becomes clear that with respect to runtimes in mode A, only minimal advances can be expected in the near future. So, from a scientific point of view, memory issues and preparation times should be the focus of new DL research projects.

There is much room for improvement wrt. space requirements if description logic reasoners are to be used as database-like semantic web repositories. Memory requirements do not allow for 50 universities to be handled by Racer (this is no problem for DLDB). Even for one university, memory consumption can be seen as a problem in mode A (220MB in the case of Racer but 16MB in the case of DLDB). In mode B Racer requires approx. 2 GBs for performing the tests.

In the complete mode (mode B), query answering times can dramatically increase for some queries (see the results on query 12) whereas for others, execution times are similiar to those in mode A. This might not be expected by many users. There is room for further optimization techniques to be developed for tableau-based A-box reasoning systems. In particular, large numbers of role assertions reveal a source of inefficiency in Racer.

With version 1.8, Racer supports incremental query answering. Thus, rather than retrieving the whole solution set in a single step (set-at-a-time approach), client applications can load elements from the solution set in a way that proceeds tuple by tuple (tuple-at-a-time iterator approach). Racer also allows for the retrieval of bundles (see also [1]). In the incremental mode, Racer 1.8 supports a query function that indicates when the last of the "cheap" answers (computed in mode A) is returned. After this point, Racer switches to mode B and retrieves the remaining tuples. However, after this point much more computational resources are required (see the discussion above). It is up to the application to decide if this is worth the effort.

2.2 KAON Benchmark

The KAON benchmarks have been investigated in [11]. They are artificial benchmarks that rely on a T-box whose taxonomy forms symmetric concept tree (SCT) of some depth d and branching factor b. The A-box part contains concept assertions such that for each concept name in the taxonomy there are n instances.

d	b	n	Inds.	Prep. (A)	Retr. (A)	Prep. (B)	Retr. (B)
3	5	20	3100	1	0.095	1	0.150
3	5	30	4650	1.6	0.15	1.6	0.2
4	5	10	7800	3.3	0.2	3	0.3
4	5	30	23400	12.3	0.7	9.7	1.1
5	5	10	39050	31.6	1.2	24	7.8

Table 4: Runtimes (in secs) for SCT (mode A and mode B, respectively).

The (single) query for each benchmark is an instance retrieval query with some concept name at level 1 of the taxonomy. Thus, the result set is quite large. In the first setting, there are no role assertions provided. Table 4 lists the evaluation results for Racer in mode A (cheap answers) and mode B (complete). For both modes, preparation times for setting up index structures and retrieval times are indicated. As we can seen, preparation times and retrieval times do not increase drastically in mode B. Due to optimization techniques based on individual pseudo models [6, 5], query answering does not require expensive A-box satisfiability tests. This changes when role assertions are involved as can be seen with the next KAON benchmark. It is interesting to note that with respect to SCT benchmarks, Racer is complete even in mode A.

d	b	n	r	Inds.	Roles	Prep. (A)	Retr. (A)	Prep. (B)	Retr. (B)
3	5	10	3	1550	1549	0.6	0.02	0.9	0.07
4	5	10	3	7800	7799	4.1	0.02	11	1.6
5	5	10	3	39050	39049	40	0.3	362	> 1300

Table 5: Runtimes (in secs) for SCT with role assertions.

In the same way as above, Table 5 lists preparation and retrieval times for a knowledge base with relation assertions. All individuals are connected to form a ring which consists of r different roles (randomly selected). Retrieval times reveal that Racer is currently not really suited to deal with A-boxes with these kinds of role assertions. Since the result sets in the KAON benchmarks are quite large compared to the number of candidates, optimization techniques such as binary partitioning [5] are not effective in this case (on the contrary).

The KAON benchmarks also clearly indicate some deficiencies in the current architecture of Racer. The problems are due to the underlying implementation of the tableau algorithm. In the presence of many role assertions, linear search for the applicability of tableau rules is a tremendous source of inefficiency. Note that certain kinds of A-box transformations useful for fast index computation (so-called contractions [4]) are not possible due to the ring structure of A-box individuals and roles.

d	b	n	r	Inds.	Roles	Prep. A	Retr. A	Prep. B	Retr. B
3	5	10	10	1550	1549	0.7	0.05	0.7	> 1300

Table 6: Runtimes (in secs) for DCT benchmarks.

It is interesting to see that [11] also defines benchmarks which allow for fast preparation time even in mode B but show deteriorating performance for instance retrieval (see Table 6). The reason is that Racer is able to absorb hard T-box axioms using special transformation techniques. In this case, axioms are represented in a form that allows for lazy unfolding: $\neg A \sqsubseteq B \sqcup \forall RC$. In this kind of benchmark these axioms with "lurking" disjunctions and possible cycles occur very frequently in the T-box (hence the name, DCT, disjunctive concept tree). T-box reasoning is easy because model merging is very effective and unfolding does not occur. Since there are no negative assertions $i : \neg A$ in the initial A-box, Racer also has no problem with the initial A-box consistency test. But, since in the Racer architecture, complete instance retrieval requires the negation of the query concept to be claimed for individuals (refutation proof), axioms such as those discussed above suddenly have to be unfolded and, due to the large number of applicable axioms of this kind and their cyclic nature, combinatorial explosion occurs. At the current state, we cannot see how this can be avoided in a tableau-based complete reasoner.

In mode A, Racer is complete with respect to SCT. Query answering times do not increase dramatically if larger numbers of individuals are present in an A-box. Space requirements have not been measured here in detail but are known as large as well. In mode B, Racer suffers from a tremendous increase in query execution time due to role axioms and the specific form of cyclic T-box axioms for some query problems (e.g., DCT). However, SCT and DCT are artificial benchmarks.

3 RDF/OWL: Two Years of User Support

The Lehigh benchmark of the previous section indicates that for practical problems, A-box reasoners are not really outperformed by database systems. However, due to our experiences, in some applications A-boxes are (mis)used to represent database-like information that requires single-model reasoning (model checking). Nevertheless, current inference technology in principle includes many sources of overhead to support multi-model reasoning in order to be complete. With Racer users have the choice: They can benefit from fast query answering (called mode A in this article) and from complete query answering (mode B). Incremental query answering strategies even provide a means to quickly retrieve "cheap" answers before the "expensive" answers are derived.

In practice, the situation with A-boxes becomes even more complicated since with taking T-boxes off-the-shelf as suggested by the semantic web initiative, user tend to use most expressive T-box languages. Often, these T-boxes are more expressive than required.

3.1 Beware of Cyclic Axioms

Domain and range restrictions can be absorbed in many cases (even qualified domain restrictions are absorbed in Racer). However, as discussed in the KAON benchmark, in some cases domain or range restrictions have the consequence that cycles occur in the T-box and blocking is required. Users usually ignore Racer's warnings about cycles, and wonder about performance penalities due to higher typical-case complexity caused by blocking.

3.2 Avoid Inverse Roles

If blocking is considered, inverse roles add to average-case inference complexity enormously. Although optimizations concerning blocking [8] are also included in Racer (to some extent), inverse roles as well as cyclic knowledge bases are known to be hard. There is currenly no caching supported if inverse roles are present. If hard T-boxes with inverse roles and blocking required are referred to by A-boxes, performance for instance retrieval queries in mode B usually deteriorates. Unfortunately, users do not see why this occurs. Indeed, users might benefit from some (system-specific) feedback about the "hardness" of the inference problems that they specify.

3.3 Datatypes Instead of Nominals

In the current version of Racer, nominals are only approximated by atomic concepts. Racer will soon be updated to appropriately deal with nominals according to [9]. We assume that nominals will be used extensively in applications. However, in case of nominals being present in T-boxes important optimization techniques can no longer be used effectively (without substatial changes to the architecture). Due to our experiences, nominals are not always required but just datatypes can be used for solving practical problems. At least with Racer, this will allow for much better average-case performance.

3.4 Annotation Properties for Storing Data

Retrieval functions for datatype values provided by nRQL return only "told" information. Thus, there is actually no way to compute specific concrete values due to constraint solving right now. Interestingly, this does not seem to be a problem in practice. However, if reasoning is not the point but just data storage, the question is whether OWL's annotation properties could be used more extensively in applications. Racer deals with annotation properties as effectively as database systems do (fillers of annotation properties are not treated as constraints but just values, whereas, in order to be complete, fillers of datatype properties must be treated as constraints).

4 Summary

We have seen that Racer can handle large numbers of individuals quite effectively if services similar to those offered by database-like systems are requested. Mode A is possible with RacerPro 1.8. We also see that completeness (mode B) currently comes with the disadvantage of memory consumption and main-memory based algorithms. The benchmarks indicate that role assertions are a source of inefficiency in the current implementation of Racer. This will be solved in the near future. However, a sound and complete description logic inference engine that does not rely on main memory for languages as expressive as OWL DL really constitutes a new research endeavor. Perhaps, data retrieval should be the starting point for new architectures, with reasoning added on top.

References

[1] R. Fikes, P. Hayes, and I. Horrocks. OWL-QL - a language for deductive query answering on the semantic web. Technical Report KSL-03-14, Knowledge Systems Lab, Stanford University, CA, USA, 2003.

[2] Y. Guo, J. Heflin, and Z. Pan. Benchmarking DAML+OIL repositories. In *Proc. of the Second Int. Semantic Web Conf. (ISWC 2003)*, number 2870 in LNCS, pages 613–627. Springer Verlag, 2003.

[3] Y. Guo, Z. Pan, and J. Heflin. An evaluation of knowledge base systems for large OWL datasets. In *Proc. of the Third Int. Semantic Web Conf. (ISWC 2004)*, LNCS. Springer Verlag, 2004.

[4] V. Haarslev and R. Möller. An empirical evaluation of optimization strategies for ABox reasoning in expressive description logics. In *Proc. Workshop on Description Logics*, 1999.

[5] V. Haarslev and R. Möller. Optimization techniques for retrieving resources described in OWL/RDF documents: First results. In *Proc. of the 9th Int. Conf. on the Principles of Knowledge Representation and Reasoning (KR'04, Whister, BC, Canada, June 2-5)*, pages 163–173, 2004.

[6] V. Haarslev, R. Möller, and A.-Y. Turhan. Exploiting pseudo models for TBox and ABox reasoning in expressive description logics. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, volume 2083 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, 2001.

[7] V. Haarslev, R. Möller, and M. Wessel. Querying the semantic web with Racer + nRQL. In *Proc. Workshop on Applications of Description Logics*, 2004.

[8] I. Horrocks and U. Sattler. Optimised reasoning for SHIQ. In *Proc. of the 15th European Conference on Artificial Intelligence*, 2002.

[9] I. Horrocks and U. Sattler. A tableaux decision procedure for SHOIQ. In *Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI 2005)*, 2005.

[10] R. MacGregor. The evolving technology of classification-based knowledge representation systems. In John F. Sowa, editor, *Principles of Semantic Networks*, pages 385–400. Morgan Kaufmann, Los Altos, 1991.

[11] B. Motik, R. Volz, and A. Maedche. Optimizing query answering in description logics using disjunctive deductive databases. In *Proceedings of the 10th International Workshop on Knowledge Representation Meets Databases (KRDB-2003)*, pages 39–50, 2002.

[12] Z. Zhang. Ontology query languages for the semantic web: A performance evaluation. Master's thesis, University of Georgia, 2005.