Benchmark Analysis with FaCT

Ian Horrocks

University of Manchester
Manchester, UK
horrocks@cs.man.ac.uk

Abstract. FaCT (**Fa**st **C**lassification of **T**erminologies) is a Description Logic (DL) classifier that can also be used for modal logic satisfiability testing. The FaCT system includes two reasoners, one for the logic $\mathcal{SHF}$ and the other for the logic $\mathcal{SHIQ}$, both of which use optimised implementations of sound and complete tableaux algorithms. FaCT's most interesting features are its expressive logic (in particular the $\mathcal{SHIQ}$ reasoner), its optimised tableaux implementation (which has now become the standard for DL systems), and its CORBA based client-server architecture.

1 The FaCT System

The logics implemented in FaCT are both based on $\mathcal{ALC}_{R^+}$, an extension of $\mathcal{ALC}$ to include transitive roles [13]. For compactness, this logic has be called $\mathcal{S}$ (due to its relationship with the proposition multi-modal logic $\mathbf{S4_{(m)}}$ [14]). $\mathcal{SHF}$ extends $\mathcal{S}$ with a hierarchy of roles and functional roles (attributes), while $\mathcal{SHIQ}$ adds inverse roles and fully qualified number restrictions.

The $\mathcal{SHIQ}$ reasoner is of particular interest, both form a theoretical and a practical viewpoint. Adding inverse roles to $\mathcal{SHF}$ (to give $\mathcal{SHIF}$) already leads to the loss of the finite model property, and this has necessitated the development of a more sophisticated *double dynamic* blocking strategy that allows the algorithm to find finite representations of infinite models while still guaranteeing termination [7]. Moreover, when $\mathcal{SHIF}$ is generalised to $\mathcal{SHIQ}$, it is necessary to restrict the use of transitive roles in number restrictions in order to maintain decidability [8]. $\mathcal{SHIQ}$ is also of great practical interest as it is powerful enough to encode the logic DLR, and can thus be used for reasoning about conceptual data models, e.g., Extended Entity-Relationship (EER) schemas [2, 4].

2 Implementation

FaCT is implemented in Common Lisp, and has been run successfully with several commercial and free lisps, including Allegro, Liquid (formerly Lucid), Lispworks and GNU. Binaries (executable code) are now available (in addition to the source code) for Linux and Windows systems, allowing FaCT to used without a locally available Lisp.

In order to make the FaCT system usable in realistic applications, a wide range of optimisation techniques are used in the implementation of the satisfiability testing algorithms. These include axiom absorption, lexical normalisation, semantic branching search, simplification, dependency directed backtracking, heuristic guided search and caching [6]. The use of these (and other) optimisation techniques has now become standard in tableaux-based DL implementations [10, 5].

The current implementation of $\mathcal{SHIQ}$ is a relatively naive modification of the $\mathcal{SHF}$ reasoner: it does not use the more efficient form of double blocking described in [8], it does not include any special optimisations to deal with inverse roles (or take advantage of their absence), and some optimisations that would require modification in the presence of inverse roles are instead simply disabled. As a result, performance with $\mathcal{SHIQ}$ is significantly worse than with $\mathcal{SHF}$, even w.r.t. $\mathcal{SHF}$ problems. These issues are being addressed in a new implementation of the $\mathcal{SHIQ}$ reasoner.

Work is also underway on the development of Abox reasoning for the FaCT system: an $\mathcal{SHF}$ Abox has recently been released [15] and a full $\mathcal{SHIQ}$ Abox is being developed [9].

3 Special Features

In addition to the standard KRSS functional interface [11], FaCT can also be configured as a classification and reasoning server using the Object Management Group's Common Object Request Broker Architecture (CORBA) [1]. This approach has several advantages: it facilitates the use of FaCT by non-Lisp client applications; the API is defined using CORBA's Interface Definition Language (IDL), which can be mapped to various target languages; a mechanism is provided for applications to communicate with the DL system, either locally or remotely; and server components can be added/substituted without client applications even being aware of the change. This has allowed, for example, the successful use of FaCT's reasoning services in a (Java based) prototype EER schema integration tool developed as part of the DWQ project [3].

4 Performance Analysis

FaCT's $\mathcal{SHIQ}$ reasoner (version 2.13.14) was used with those problems involving inverse roles; in all other cases the $\mathcal{SHF}$ reasoner (version 2.13.3) was used. The tests were run on two machines, one with a 450MHz Pentium III and 128Mb of RAM, the other with a 433MHz Celeron and 256Mb of RAM. In both cases Allegro CL Enterprise Edition 5.0 was used with Red Hat Linux. For the purposes of these tests the difference in performance between the two machines is small enough to be ignored.

As far as FaCT's performance is concerned, the current implementation is beginning to show its age: the system has been used as a testbed for new algorithms and optimisation techniques, and after nearly five years of "evolution" a major overhaul is long overdue. This is particularly true of the $\mathcal{SHIQ}$ reasoner.[1] It is therefore unlikely that FaCT's performance will be competitive with that of younger and leaner systems whose designs reflect the experience gained with the FaCT system. Moreover, FaCT's optimisations are specifically aimed at improving the system's performance when classifying realistic knowledge bases (KBs), and perform less well with the kinds of randomly generated data used in these tests. In particular, the partial model caching technique used by FaCT relies for its effectiveness on the fact that, in realistic KBs, there are typically large numbers of different roles (modalities). In contrast, most of the TANCS test data uses only a single role. Moreover, the data used in the TANCS test is very susceptible to optimisation by caching the satisfiability status of sets of formulae [6]. This can be seen in Table 4, which shows the results of running the "final" set of TANCS QBF tests using both FaCT's standard $\mathcal{SHF}$ reasoner and a modified version (denoted FaCT†) that includes satisfiability status caching instead of partial model caching. The results for satisfiable and unsatisfiable tests are separated and in each case the number of instances solved (i), median time (m) and worst case time (w) is given. Times are in seconds, with a timeout of 1,000s. There were a total of 64 instances in each test, and all unsolved instances were the result of timeout rather than memory failure.

Table 1. Results of "final" TANCS tests for FaCT and FaCT†

	FaCT						FaCT†					
	SAT			UNSAT			SAT			UNSAT		
Test	i	m	w	i	m	w	i	m	w	i	m	w
C45-V8-D4	4	606.57	717.39	23	106.43	813.78	32	5.26	11.64	32	1.07	10.58
C45-V8-D5	0	–	–	2	547.02	774.59	32	139.97	721.68	32	20.52	259.09
C55-V8-D6	0	–	–	3	595.26	631.98	28	89.04	328.41	36	13.85	215.70

It is interesting to compare these results with the the times taken to classify a large realistic KB (the Galen medical terminology KB [12]). In this case satisfiability status caching is actually less effective than partial model caching: FaCT takes ≈41s to classify the KB, whereas FaCT† takes ≈50s.

The standard $\mathcal{SHF}$ reasoner was also tested on the Periodic Satisfiability (Global PSpace) reference problems. With the standard (CNF) encoding, these were all very easy (most problems were solved in times too short to by accurately measured), but with the K encoding the PSat problems became *much* harder, and few were solved within the timeout.[2] This effect is much less pronounced with the QBF problems, probably because absorption is only relevant with global axioms. A more detailed analysis was performed using harder PSat problems

[1] As mentioned above, these (and other) issues are being addressed in a new implementation.

[2] This causes of this effect are being investigated—it is probably related to the absorption optimisation.

with the standard encoding (12 variables, depth 1, 24–192 clauses, 30 instances per data point, 600s timeout), and the results are shown in Figure 1 (left).

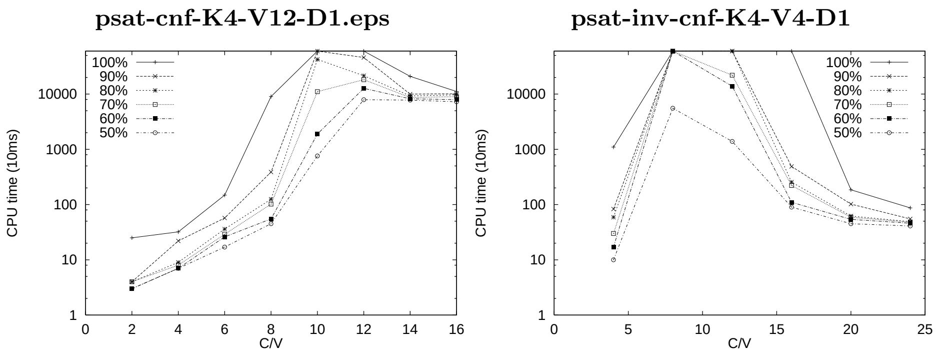

Fig. 1. Percentile satisfiability times for two problem sets

The $\mathcal{SHIQ}$ reasoner was tested using the Modal QBF (Modal PSpace) and Periodic Satisfiability (Global PSpace) problems with inverse. The performance of the reasoner in these tests was quite poor, with most tests ending in either timeout or memory failure. The results of those tests where at least one instance was solved are given in Table 4, using the same format as Table 4, but with the addition of the number of tests resulting in a timeout (T) or memory failure (M). The reason for the poor performance is probably the lack of satisfiability status caching which, as demonstrated above, is a crucially important optimisation with this kind of test data. A more detailed analysis was performed using easier PSat problems with the standard encoding (4 variables, depth 1, 16–96 clauses, 30 instances per data point, 600s timeout), and the results are shown in Figure 1 (right).

5 Availability

FaCT and iFaCT are available (under the GNU general public license) via the WWW at `http://www.cs.man.ac.uk/~horrocks`.

References

1. S. Bechhofer, I. Horrocks, P. F. Patel-Schneider, and S. Tessaris. A proposal for a description logic interface. In *Proc. of DL'99*, pages 33–36, 1999.
2. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Source integration in data warehousing. In *Proc. of DEXA-98*, pages 192–197, 1998.

Table 2. Results of tests using FaCT's $\mathcal{SHIQ}$ reasoner

	SAT			UNSAT				
Test	i	m	w	i	m	w	T	M
p-qbf-inv-cnfSSS-K4-C10-V4-D4	2	11.19	32.70	0	–	–	0	6
p-qbf-inv-cnfSSS-K4-C20-V4-D4	1	36.43	36.43	2	70.37	155.62	0	5
p-qbf-inv-cnfSSS-K4-C30-V4-D4	1	112.48	112.48	1	275.13	275.13	0	6
p-qbf-inv-cnfSSS-K4-C40-V4-D4	0	–	–	5	176.97	376.15	0	3
p-qbf-inv-cnfSSS-K4-C50-V4-D4	0	–	–	8	11.23	61.44	0	0
p-qbf-inv-cnfSSS-K4-C20-V4-D6	0	–	–	1	1.58	1.58	0	7
p-qbf-inv-cnfSSS-K4-C40-V4-D6	0	–	–	1	50.72	50.72	0	7
p-qbf-inv-cnfSSS-K4-C50-V4-D6	0	–	–	6	123.88	341.35	0	2
p-psat-inv-cnf-K4-C20-V4-D1	8	1.59	45.61	0	–	–	0	0
p-psat-inv-cnf-K4-C30-V4-D1	3	11.58	58.59	0	–	–	0	5
p-psat-inv-cnf-K4-C40-V4-D1	4	64.32	106.05	1	2.56	2.56	3	0
p-psat-inv-cnf-K4-C50-V4-D1	5	1.81	17.98	1	1.79	1.79	2	0
p-psat-inv-cnf-K4-C20-V8-D1	7	0.05	39.06	0	–	–	0	1
p-psat-inv-cnf-K4-C30-V8-D1	5	14.03	108.67	0	–	–	0	3
p-psat-inv-cnf-K4-C20-V8-D2	2	0.04	0.14	0	–	–	2	4

3. D. Calvanese, G. De Giacomo, M. Lenzerini, D. Nardi, and R. Rosati. Use of the data reconciliation tool at telecom italia. DWQ deliverable D.4.3, Foundations of Data Warehouse Quality (DWQ), 1999.
4. E. Franconi and G. Ng. The i•com tool for intelligent conceptual modelling. Submitted to VLDB-2000.
5. V. Haarslev, R. Möller, and A.-Y. Turhan. Implementing an $\mathcal{ALCRP}(\mathcal{D})$ abox reasoner – progress report. In *Proc. of DL'98*, pages 82–86, 1998.
6. I. Horrocks and P. F. Patel-Schneider. Optimising description logic subsumption. *Journal of Logic and Computation*, 9(3):267–293, 1999.
7. I. Horrocks and U. Sattler. A description logic with transitive and inverse roles and role hierarchies. *Journal of Logic and Computation*, 9(3):385–410, 1999.
8. I. Horrocks, U. Sattler, and S. Tobies. Practical reasoning for expressive description logics. In *Proc. of LPAR'99*, pages 161–180. Springer-Verlag, 1999.
9. I. Horrocks, U. Sattler, and S. Tobies. Reasoning with individuals for the description logic $\mathcal{SHIQ}$. In *Proc. of CADE-2000*, 2000. To appear.
10. P. F. Patel-Schneider. DLP system description. In *Proc. of DL'98*, pages 87–89, 1998.
11. P. F. Patel-Schneider and B. Swartout. Description logic specification from the KRSS effort, June 1993.
12. A. L. Rector, W A Nowlan, and A Glowinski. Goals for concept representation in the Galen project. In *Proc. of SCAMC'93*, pages 414–418, 1993.
13. U. Sattler. A concept language extended with different kinds of transitive roles. In *20. Deutsche Jahrestagung für Künstliche Intelligenz*, pages 333–345, 1996.
14. K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proc. of IJCAI-91*, pages 466–471, 1991.
15. S. Tessaris and G. Gough. Abox reasoning with transitive roles and axioms. In *Proc. of DL'99*, 1999.