Proceedings of Tableaux'98.
Lecture Notes in Artificial Intelligence No. 1397, Springer: Berlin, May 1998, pp. 27–30.

FaCT and DLP

Ian Horrocks[1,2] and Peter F. Patel-Schneider[3]

[1] Medical Informatics Group, Department of Computer Science,
University of Manchester, Manchester M13 9PL, UK
horrocks@cs.man.ac.uk
[2] IRST, Istituto per la Ricerca Scientifica e Tecnologica, I-38050 Povo TN, ITALY
[3] Bell Labs Research, Murray Hill, NJ, U.S.A.
pfps@research.bell-labs.com

FaCT: The tests were performed using FaCT version 1.2. FaCT is a description logic classifier whose description language is a superset of $\mathbf{K4_{(m)}}$ and whose subsumption reasoning uses a sound and complete tableaux algorithm. FaCT employs a wide range of optimisations, in particular a form of dependency directed backtracking called *backjumping* which can significantly reduce the size of the search space [5]. The FaCT algorithm does not support **KT** and **S4** explicitly, but FaCT includes a preprocessing and encoding optimisation which is also able to apply the standard embedding of **KT** and **S4** in **K** and **K4** respectively: the time taken for preprocessing and embedding is included in the results. Programming language: Common Lisp (compiled).

DLP: The ideas in FaCT are being incorporated into a new generation of Description Logic systems. Initial experiments in this effort have resulted in a modal prover for a superset of $\mathbf{K4_{(m)}}$, which has provisionally been called DLP. The DLP prover has control over several options, including backjumping and caching partial results. Both of these mechanisms have proved to be very useful in the benchmarks, with caching being the more powerful. As an experimental prover, there are essentially no user amenities in DLP, but the final Description Logic system will have a full user interface and other amenities. Programming language: ML (compiled).

The other provers: For comparative purposes the tests for **K** and **KT** were repeated using three other available provers: Crack version 1.0 beta 15 [3], KSAT [4] and Kris [2, 1]. Crack and Kris are also description logic classifiers which use sound and complete tableaux algorithms while KSAT is a $\mathbf{K_{(m)}}$ prover which uses an algorithm based on propositional satisfiability (SAT) testing. None of these systems supports transitive relations so they could not be used for **S4**. The **KT** tests were performed by using the standard embedding of **KT** in **K**: the time taken for the embedding is not included in the results for these systems.

All three systems are programmed in Common Lisp (compiled). It should be pointed out that neither Crack nor Kris are intended as stand-alone **K** provers and for many classes of formula a significant improvement in their performance could be achieved by preprocessing and encoding large formulae, a technique which is used by both FaCT and KSAT. Both Crack and Kris support much richer logics than **K** (for example Crack can reason about converse relations) and can also reason about nominals (individuals).

Availability: The sources for FaCT are available from the first authors home page: *http://www.cs.man.ac.uk/ horrocks*; the DLP prover is currently under development, but the benchmark version and full timing results are also available from the same location. Contacts for information about the other systems are:

Crack — Enrico Franconi, *franconi@irst.itc.it*;
KSAT — Roberto Sebastiani, *rseba@irst.itc.it*;
Kris — H.-J. Burckert, *hjb@dfki.uni-sb.de.*

Advantages: FaCT has been tested using several Common Lisps including GNU Lisp and should thus be highly portable. As well as **K**, **KT** and **S4** it can also deal with **K4**. The implemented logic is significantly more expressive than **S4**: it includes support for a hierarchy of multiple modalities (roles), functional roles and global axioms. DLP should also be highly portable: the ML compiler runs on a variety of platforms and is freely available from several sites, including *http://cm.bell-labs.com/cm/cs/what/smlnj.*

Hardware and Software: For DLP: SPARC clone; main memory 132MB; 150 MHz Ross RT626 CPU; SML-NJ compiler, version 109.32. For the other provers: Sun Ultra 1; main memory 32MB; 147 MHz CPU; Solaris; Allegro CL 4.3.

Results: To demonstrate the effectiveness of the backjumping optimisation the tests were also performed using FaCT with backjumping disabled: the resulting prover is referred to as FaCT*. The results of the tests are given in Tables 1, 2 and 3. Both FaCT and DLP performed reasonably well with all classes of **K** and **KT** formula, trivially solving most of the **K** formulae, and in the case of DLP many of the **KT** and **S4** formulae.

FaCT and DLP significantly outperformed all the other provers, and in many cases they also exhibited a completely different qualitative performance. For example, with *k_dum_p* the other provers all show an exponential increase in solution times with increasing formula size, whereas the times taken by FaCT and DLP increase very little for larger formulae (and FaCT is already 2,000 times faster for the largest formula solved by another system).

The results for FaCT* demonstrate that backjumping accounts for a significant proportion of FaCT's performance advantage over the other systems, particularly with respect to provable formulae, and experiments with DLP suggest that caching is even more effective. However the performance of FaCT* still compares favourably with that of the other systems and it still exhibits a

Table 1. Results for **K**, **KT** and **S4**

	FaCT		**FaCT***		**DLP**		**Crack**		**KSAT**		**Kris**	
Formulae	p	n	p	n	p	n	p	n	p	n	p	n
*k_branch_*	6	4	3	3	13	11	2	1	8	8	3	3
*k_d4_*	>20	8	15	8	>20	>20	2	3	8	5	8	6
*k_dum_*	>20	>20	15	>20	>20	>20	3	>20	11	>20	15	>20
*k_grz_*	>20	>20	8	>20	>20	>20	1	>20	17	>20	13	>20
*k_lin_*	>20	>20	7	>20	>20	>20	5	2	>20	3	6	9
*k_path_*	7	6	5	5	>20	>20	2	6	4	8	3	11
*k_ph_*	6	7	5	6	6	8	2	3	5	5	4	5
*k_poly_*	>20	>20	>20	>20	>20	>20	>20	>20	13	12	11	>20
*k_t4p_*	>20	>20	>20	>20	>20	>20	1	1	10	18	7	5

Table 2. Results for **KT**

	FaCT		**FaCT***		**DLP**		**Crack**		**KSAT**		**Kris**	
Formulae	p	n	p	n	p	n	p	n	p	n	p	n
*kt_45_*	>20	>20	13	>20	>20	>20	0	0	5	5	4	3
*kt_branch_*	6	4	3	3	16	11	2	2	8	7	3	3
*kt_dum_*	11	>20	8	>20	>20	>20	0	1	7	12	3	14
*kt_grz_*	>20	>20	5	>20	>20	>20	0	0	9	>20	0	5
*kt_md_*	4	5	3	5	3	>20	2	4	2	4	3	4
*kt_path_*	5	3	2	2	6	>20	1	5	2	5	1	13
*kt_ph_*	6	7	4	5	7	18	2	2	4	5	3	3
*kt_poly_*	>20	7	>20	6	6	6	1	1	1	2	2	2
*kt_t4p_*	4	2	1	1	3	>20	0	1	1	1	1	7

Table 3. Results for **S4**

	FaCT		**FaCT***		**DLP**	
Formulae	p	n	p	n	p	n
*s4_45_*	>20	>20	8	>20	>20	>20
*s4_branch_*	4	4	2	2	10	8
*s4_grz_*	2	>20	0	>20	9	>20
*s4_ipc_*	5	4	4	4	10	>20
*s4_md_*	8	4	3	4	3	>20
*s4_path_*	2	1	2	1	3	>20
*s4_ph_*	5	4	4	3	7	18
*s4_s5_*	>20	2	2	2	3	>20
*s4_t4p_*	5	3	1	1	>20	>20

different qualitative performance in some cases (e.g. *k_lin_p*). DLP is more effective with non-provable formulae, and for some classes of provable formulae it is outperformed by FaCT; this phenomenon is the subject of continuing research.

A well engineered C code implementation of KSAT is now available, and has been observed to outperform the Lisp version by a significant margin (as much as 100 times). It is likely that significant improvements to the performance of FaCT and DLP could also be achieved by employing more sophisticated software engineering.

References

1. F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.-J. Profitlich. An empirical analysis of optimization techniques for terminological representation systems. In B. Nebel, C. Rich, and W. Swartout, editors, *Principals of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR'92)*, pages 270–281. Morgan-Kaufmann, 1992. Also available as DFKI RR-93-03.
2. F. Baader and B. Hollunder. A terminological knowledge representation system with complete inference algorithms. In *Processing declarative knowledge: International workshop PDK'91*, number 567 in Lecture Notes in Artificial Intelligence, pages 67–86, Berlin, 1991. Springer-Verlag.
3. P. Bresciani, E. Franconi, and S. Tessaris. Implementing and testing expressive description logics: a preliminary report. In Gerard Ellis, Robert A. Levinson, Andrew Fall, and Veronica Dahl, editors, *Knowledge Retrieval, Use and Storage for Efficiency: Proceedings of the First International KRUSE Symposium*, pages 28–39, 1995.
4. F. Giunchiglia and R. Sebastiani. A SAT-based decision procedure for $\mathcal{ALC}$. In L. C. Aiello, J. Doyle, and S. Shapiro, editors, *Principals of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR'96)*, pages 304–314. Morgan Kaufmann, November 1996.
5. I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics.* PhD thesis, University of Manchester, 1997.