The FaCT System

Ian Horrocks

Medical Informatics Group, Department of Computer Science,
University of Manchester, Manchester M13 9PL, UK
horrocks@cs.man.ac.uk
http://www.cs.man.ac.uk/~horrocks

Abstract. FaCT is a Description Logic classifier which has been implemented as a test-bed for a highly optimised tableaux satisfiability (subsumption) testing algorithm. The correspondence between modal and description logics also allows FaCT to be used as a theorem prover for the propositional modal logics **K**, **KT**, **K4** and **S4**. Empirical tests have demonstrated the effectiveness of the optimised implementation and, in particular, of the dependency directed backtracking optimisation.

1 Introduction

FaCT [1] is a Description Logic (DL) classifier which has been implemented as a test-bed for a highly optimised tableaux satisfiability/subsumption testing algorithm. The underlying logic, $\mathcal{ALCH}_{R^+}$, is a superset of the $\mathcal{ALC}$ DL, and this means that FaCT can be used as a theorem prover for the propositional modal logic $\mathbf{K_{(m)}}$ (**K** with multiple modalities) by exploiting the well known correspondence between the two logics [17]. Because $\mathcal{ALCH}_{R^+}$ supports transitive relations, FaCT can also be used as a prover for $\mathbf{K4_{(m)}}$, and it extends the range of logics it can deal with to include $\mathbf{KT_{(m)}}$ and $\mathbf{S4_{(m)}}$ by embedding formulae in $\mathbf{K_{(m)}}$ and $\mathbf{K4_{(m)}}$ respectively.

In order to make the FaCT system usable in realistic DL applications, a wide range of optimisation techniques are used in the implementation of the $\mathcal{ALCH}_{R^+}$ satisfiability testing algorithm. Although some of these techniques were designed to take advantage of the structure of a DL knowledge base (KB), and the repetitive nature of the satisfiability problems encountered when classifying a KB, some of the optimisations are also effective in improving FaCT's performance with respect to single satisfiability problems.

2 Description Logics and Modal Logics

Description Logics support the logical description of concepts and roles (relationships) and their combination, using a variety of operators, to form more complex descriptions. The $\mathcal{ALC}$ DL [18] allows descriptions to be formed using

[1] **F**ast **C**lassification of **T**erminologies.

standard logical connectives as well as both universally and existentially quantified relational operators: if C is a concept and R is a role then an $\mathcal{ALC}$ concept expression is of the form $C \mid \top \mid \bot \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C$. A Tarski style model theoretic semantics is used to interpret expressions [3].

Table 1 shows how propositional $\mathbf{K_{(m)}}$ formulae correspond to $\mathcal{ALC}$ concept expressions. Note that the modal operators $\Box$ and $\Diamond$ correspond to $\exists R.C$ and $\forall R.C$ expressions, with different roles corresponding to distinct modalities or accessibility relations. Standard modal $\mathbf{K}$ ($\mathbf{K}_{\mathbf{(1)}}$) has only one modality, so modal $\mathbf{K}$ formulae correspond to $\mathcal{ALC}$ concept expressions containing a single role. The correspondence can be extended to $\mathbf{K4_{(m)}}$ simply by making all roles transitive.

Table 1. The correspondence between modal $\mathbf{K_{(m)}}$ and $\mathcal{ALC}$

$\mathbf{K_{(m)}}$	$\mathcal{ALC}$	$\mathbf{K_{(m)}}$	$\mathcal{ALC}$
True	$\top$	False	$\bot$
ϕ	C	$\neg\phi$	$\neg C$
$\phi \wedge \varphi$	$C \sqcap D$	$\phi \vee \varphi$	$C \sqcup D$
$\Box_i \phi$	$\forall R_i.C$	$\Diamond_i \phi$	$\exists R_i.C$

FaCT also supports $\mathbf{KT_{(m)}}$ and $\mathbf{S4_{(m)}}$ by embedding formulae in $\mathbf{K_{(m)}}$ and $\mathbf{K4_{(m)}}$: $\Box_i \phi$ becomes $\phi \wedge \Box_i \phi$ and $\Diamond_i \phi$ becomes $\phi \vee \Diamond_i \phi$.

3 The $\mathcal{ALCH}_{R^+}$ Tableaux Algorithm

The tableau algorithm for $\mathcal{ALCH}_{R^+}$ is extended from an algorithm for the $\mathcal{ALC}_{R^+}$ DL described in [16]. The full algorithm, along with a proof of its soundness and correctness, is given in [14].

The main features of the algorithm are:

1. it uses a "single pass" tableau construction and search method as is usual in DL tableaux algorithms where logics generally have the finite model property;
2. transitive roles are dealt with simply by propagating $\Box_i \phi$ terms along i relations;
3. termination is ensured by "blocking"—checking for cycles in the tableau construction [7, 1].

4 Optimisations

To improve the performance of the $\mathcal{ALCH}_{R^+}$ satisfiability testing algorithm, a range of optimisations have been employed. These include lexical normalisation and encoding, semantic branching search and dependency directed backtracking.

4.1 Normalisation and Encoding

In DL terminologies, large and complex concepts are seldom described monolithically, but are built up from a hierarchy of named concepts whose descriptions are less complex. The tableaux algorithm can take advantage of this structure by trying to find contradictions between concept names before substituting them with their definitions and continuing with the tableau expansion: we will call this strategy *lazy unfolding*. In fact it has been shown (in the Kris system) that a significant improvement in performance can be obtained simply by not deleting names when they are lazily unfolded [2]. This is because obvious contradictions can often be detected earlier by comparing names rather than unfolded definitions.

FaCT takes this optimisation to its logical conclusion by lexically normalising and encoding all formulae and, recursively, their sub-formulae, so that:

1. All formulae are named; e.g., $\Diamond_i(\phi \wedge \varphi)$ would be encoded as $\Diamond_i \Phi$, where $\Phi = \phi \wedge \varphi$.
2. All formulae are in a standard form; e.g., all $\Diamond$ formulae are converted to $\Box$ formulae, so $\Diamond_i \phi$ would be normalised to $\neg\Box_i\neg\phi$. The encoded sub-formulae in conjunctions and disjunctions are also sorted.

Adding normalisation (step 2) allows lexically equivalent formulae to be recognised and identically encoded; it can also lead to the detection of formulae which are trivially satisfiable or unsatisfiable.

4.2 Semantic Branching Search

Standard tableaux algorithms use an inherently inefficient search technique for the non-deterministic expansion of disjunctive formulae—they choose an unexpanded disjunction and check the different tableaux obtained by adding each of the disjuncts [11]. As the alternative branches of the search are not disjoint, there is nothing to prevent the recurrence of unsatisfiable disjuncts.

FaCT deals with this problem by using a semantic branching technique adapted from the Davis-Putnam-Logemann-Loveland procedure (DPL) commonly used to solve propositional satisfiability (SAT) problems [8, 10]. Instead of choosing an unexpanded disjunction, a single disjunct ϕ is chosen from the set of unexpanded disjunctions, and the two possible tableaux obtained by adding either ϕ or $\neg\phi$ are then searched.

During the DPL search, FaCT also performs boolean constraint propagation (BCP) [9], a technique which maximises deterministic expansion, and thus pruning of the search via contradiction detection, before performing non-deterministic expansion. BCP works by deterministically expanding disjunctions which present only one expansion possibility, and detecting a contradiction when there is a disjunction which no longer has any expansion possibilities. In effect, BCP applies the inference rule $\frac{\neg\phi,\, \phi\vee\varphi}{\varphi}$ to disjunctive formulae encountered in the tableau expansion, or in other words, performs some localised propositional resolution.

4.3 Dependency Directed Backtracking

Inherent unsatisfiability concealed in sub-formulae can lead to large amounts of unproductive backtracking search known as thrashing. For example, expanding the formula $(\phi_1 \vee \varphi_1) \wedge \ldots \wedge (\phi_n \vee \varphi_n) \wedge \Diamond_i(\phi \wedge \varphi) \wedge \Box_i \neg \phi$ could lead to the fruitless exploration of 2^n possible expansions of $(\phi_1 \vee \varphi_1) \wedge \ldots \wedge (\phi_n \vee \varphi_n)$ before the inherent unsatisfiability of $\Diamond_i(\phi \wedge \varphi) \wedge \Box_i \neg \phi$ is discovered.

This problem is addressed by adapting a form of dependency directed backtracking called *backjumping*, which has been used in solving constraint satisfiability problems [5]. Backjumping works by labeling formulae with a dependency set indicating the branching choices on which they depend. When a contradiction is discovered, the dependency sets of the contradictory formulae can be used to identify the most recent branching point where exploring an alternative branch might alleviate the cause of the contradiction. The algorithm can then jump back over intervening branching points *without* exploring any alternative branches. A similar technique was employed in the HARP theorem prover [15].

5 Performance

FaCT's performance as a modal logic theorem prover has been tested using both randomly generated formulae, a test method described in [12] and derived from a widely used procedure for testing SAT decision procedures [10], and a corpus of carefully designed benchmark formulae [13].

FaCT performs well in tests using randomly generated formulae [14], but its advantages are more clearly demonstrated by the benchmark formulae, and in particular by the provable formulae.[2] This is because the hardness of these formulae often derives from hidden unsatisfiability, a phenomenon which rarely occurs in the randomly generated formulae where hardness is simply a feature of the problem size. Figure 1, for example, shows CPU time plotted against problem size for 2 classes of formulae from the **K** benchmark suite, *k-dum-p* and *k-grz-p*. FaCT's performance is compared with that of the Crack DL [6], the KSAT $\mathbf{K_{(m)}}$ theorem prover [11] and the Kris DL [4]. The performance of FaCT with the backjumping optimisation disabled is also shown, indicated in the graphs by FaCT*. All the systems use compiled Lisp code (Allegro CL 4.3), and the tests were performed on a Sun Ultra 1 with a 147 MHz CPU and 64MB of RAM.

It can be seen from the graphs that FaCT not only outperforms the other systems, but that it exhibits a completely different qualitative performance: solution times for all the other systems increase exponentially with increasing formula size, whereas those for FaCT increase only very slowly. Extrapolating the results for the other systems suggests that FaCT would be several orders of magnitude faster for the largest problems in the test sets. The results for FaCT* demonstrate that, for these formulae, backjumping accounts for FaCT's performance advantage.

[2] Note that a formula is proved by showing that its negation is unsatisfiable.

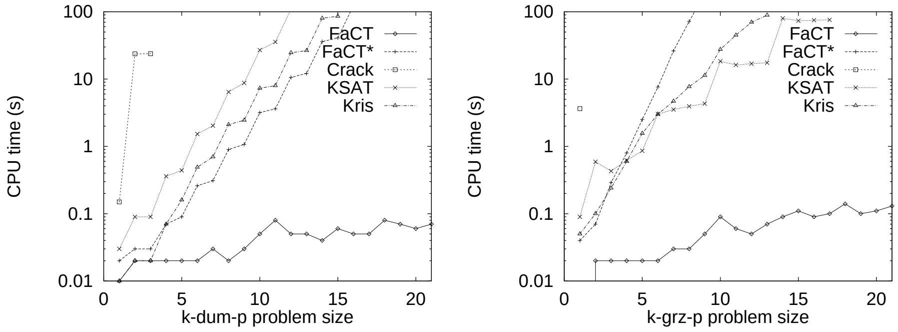

Fig. 1. Solving **K** satisfiability problems

6 Conclusion

Although it was designed for subsumption testing in a DL classifier, FaCT's optimised $\mathcal{ALCH}_{R^+}$ satisfiability testing algorithm also performs well as a propositional modal logic theorem prover, and enables FaCT to outperform the other systems with which it has been compared. Backjumping has been shown to be particularly effective, changing both the quantitative and the qualitative performance of the algorithm for some classes of hard unsatisfiable formulae.

References

1. F. Baader, M. Buchheit, and B. Hollunder. Cardinality restrictions on concepts. *Artificial Intelligence*, 88(1–2):195–213, 1996.
2. F. Baader, E. Franconi, B. Hollunder, B. Nebel, and H.-J. Profitlich. An empirical analysis of optimization techniques for terminological representation systems or: Making KRIS get a move on. 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.
3. F. Baader, H.-J. Heinsohn, B. Hollunder, J. Muller, B. Nebel, W. Nutt, and H.-J. Profitlich. Terminological knowledge representation: A proposal for a terminological logic. Technical Memo TM-90-04, Deutsches Forschungszentrum für Künstliche Intelligenz GmbH (DFKI), 1991.
4. F. Baader and B. Hollunder. Kris: Knowledge representation and inference system. *SIGART Bulletin*, 2(3):8–14, 1991.
5. A. B. Baker. *Intelligent Backtracking on Constraint Satisfaction Problems: Experimental and Theoretical Results.* PhD thesis, University of Oregon, 1995.
6. 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.

7. M. Buchheit, F. M. Donini, and A. Schaerf. Decidable reasoning in terminological knowledge representation systems. *Journal of Artificial Intelligence Research*, 1:109–138, 1993.
8. M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5:394–397, 1962.
9. J. W. Freeman. *Improvements to propositional satisfiability search algorithms*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, Philadelphia, PA, USA, 1995.
10. J. W. Freeman. Hard random 3-SAT problems and the Davis-Putnam procedure. *Artificial Intelligence*, 81:183–198, 1996.
11. F. Giunchiglia and R. Sebastiani. Building decision procedures for modal logics from propositional decision procedures—the case study of modal K. In Michael McRobbie and John Slaney, editors, *Proceedings of the Thirteenth International Conference on Automated Deduction (CADE-13)*, number 1104 in Lecture Notes in Artificial Intelligence, pages 583–597. Springer, 1996.
12. 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.
13. A. Heuerding and S. Schwendimann. A benchmark method for the propositional modal logics k, kt, s4. Technical report IAM-96-015, University of Bern, Switzerland, October 1996.
14. I. Horrocks. *Optimising Tableaux Decision Procedures for Description Logics*. PhD thesis, University of Manchester, 1997.
15. F. Oppacher and E. Suen. HARP: A tableau-based theorem prover. *Journal of Automated Reasoning*, 4:69–100, 1988.
16. U. Sattler. A concept language extended with different kinds of transitive roles. In G. Görz and S. Hölldobler, editors, *20. Deutsche Jahrestagung für Künstliche Intelligenz*, number 1137 in Lecture Notes in Artificial Intelligence, pages 333–345. Springer Verlag, 1996.
17. K. Schild. A correspondence theory for terminological logics: Preliminary report. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91)*, pages 466–471, 1991.
18. M. Schmidt-Schauß and G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48:1–26, 1991.