KBlab: an equational theorem prover for the Macintosh

Maria Paola Bonacina **Giancarlo Sanna**
Dipartimento di Scienze dell'Informazione
Università degli Studi di Milano

KBlab is a Completion based theorem prover for equational logic, written in the language C and developed on the Macintosh in the MPW (Macintosh Programmer Workshop) programming environment. The core of KBlab is the Knuth–Bendix Completion Procedure (*KB*) [9,7,1], extended to Unfailing Knuth–Bendix (*UKB*) [5], *S–strategy* [5] and inductive theorem proving (*IKB*). *IKB* implements the Huet–Hullot method for inductionless induction [8] and the Fribourg linear strategy [3]. The Knuth–Bendix ordering [9] and both the multiset extension and the lexicographic extension of the recursive path ordering are available.

KBlab couples ease of use, portability and low cost of a small Macintosh application with advanced features for experimenting in automated reasoning. It is possible to edit theories, execute the Completion procedures, store the proof traces and even modify the search strategy, all within the same environment.

Experimentation with search strategies is one of the new features of KBlab. The user chooses the search strategy for selecting axioms during the Completion process. Strategy selection is a key feature for a Completion based theorem prover, since it affects termination and the number of critical pairs generated.

KBlab also allows the user to modify the strategy during a proof session, to direct it towards a positive result. Nine different strategies are available: *FIFO*, *LIFO*, *smallest components + FIFO or LIFO*, *by size + FIFO or LIFO*, *ordered axioms + FIFO or LIFO*, *linear*. The *smallest components* and *by size* strategies are based on counting the number of symbols in the axiom terms. The *ordered axioms* strategy extends to axioms the selected simplification ordering on terms. The FIFO or LIFO strategy solves possible conflicts.

The results of the experimentation performed with KBlab are collected in a small data base of solved problems. We have extensively tested KBlab on problems in combinatorial logic taken from [12]. So far we have succeded in proving about 23% of them, including some rather large examples. The availability of several strategies turned out to be a key feature in solving these problems, since many of them could be solved, or solved in shorter time, by modifying the selected strategy during the run (e.g. es.6 p.97, es.1 p.118, es.13 p.132, es.1 p.151, es.2 p.182). Search strategies interaction allowed also to prove that Grau's three axioms are sufficient to define a ternary Boolean algebra (example 6, page 158 in [2]).

The following table shows results obtained from running KBlab on some problems with different search strategies.

Problem A: completion of the abstract loop axioms [9] by KB.

Problem B: completion of the central groupoid axioms $((X*Y)*(Y*Z)=Y, (X*(X*X))*Y = X*Y)$ by UKB.

Problem C: proving by UKB that in group theory $x^2 = e$ implies commutativity.

Problem D: proving by S–strategy that in group theory $x^3 = e$ implies $h(h(X,Y),Y) = e$ where the commutator h is defined by $h(X,Y) = XYX^{-1}Y^{-1}$.

Problem E: proving by S–strategy that given combinators t, $tXY = YX$, and q, $qXYZ = Y(XZ)$, there exists a combinator b such that $bXYZ = X(YZ)$ [12].

Problem F: proving by S–strategy that given combinators b, $bXYZ = X(YZ)$ and m, $mX = XX$, for all X exists Y such that $XY = Y$, i.e. a fixed point [12,11].

Problem G: proving by S–strategy that given combinators b, $bXYZ = X(YZ)$ and s_2, $s_2XYZ = XZ(YY)$, for all X exists Y such that $XY = Y$, i.e. a fixed point [12,10].

Experiments with search strategies in KBlab

	FIFO	*LIFO*	*smallest components FIFO*	*smallest components LIFO*	*by size FIFO*	*by size LIFO*	*ordered axioms FIFO*	*ordered axioms LIFO*
A KB	*6* *40* *9* *14* *1.20*	*6* *40* *10* *14* *1.20*	*6* *41* *10* *14* *1.20*	*6* *41* *10* *14* *1.20*	*6* *41* *10* *14* *1.20*	*6* *41* *10* *14* *1.20*	*6* *41* *10* *14* *1.35*	*6* *41* *10* *14* *1.32*
B UKB	*2* *891* *43* *6* *101.92*	*2* ↑ 	*2* *320* *29* *6* *22.13*	*2* *342* *37* *6* *24.25*	*2* *320* *29* *6* *23.28*	*2* *342* *37* *6* *26.98*	*2* *709* *40* *6* *78.67*	*2* *318* *25* *6* *26.93*
C UKB	*5* *89* *15* *13* *4.80*	*5* *24* *10* *11* *1.12*	*5* *33* *11* *11* *1.42*	*5* *30* *10* *11* *1.25*	*5* *33* *11* *11* *1.42*	*5* *30* *10* *11* *1.22*	*5* *30* *10* *11* *1.03*	*5* *30* *10* *11* *1.02*
D S	*5* *405* *65* *36* *102.53*	*5* ↑ 	*5* *401* *76* *55* *136.05*	*5* *248* *51* *37* *41.28*	*5* *401* *76* *55* *135.93*	*5* *248* *51* *37* *41.35*	*5* ↑ 	*5* ↑
E S	*3* *119* *101* *105* *352.17*	*3* ↑ 	*3* *55* *49* *53* *51.58*	*3* *35* *35* *39* *25.28*	*3* ↑ 	*3* ↑ 	*3* ↑ 	*3* ↑
F S	*3* *9* *9* *12* *1.33*	*3* ↑ 	*3* *22* *20* *20* *6.78*	*3* ↑ 	*3* *21* *19* *19* *6.65*	*3* ↑ 	*3* *22* *20* *20* *6.87*	*3* *22* *20* *20* *6.90*
G S	*3* *18* *16* *19* *7.38*	*3* *2* *2* *5* *0.13*	*3* *2* *2* *5* *0.20*	*3* *2* *2* *5* *0.15*	*3* ↑ 	*3* *2* *2* *5* *0.13*	*3* ↑ 	*3* ↑

Each entry gives, in order, the number of axioms in the input, the number of critical pairs

generated, the number of non trivial critical pairs generated, the number of axioms in the output and the running time in seconds on the Macintosh II. The ↑ entry means that the prover was interrupted after running without yielding an answer for a much longer time than that required by the same problem with other strategies. The KBO ordering was selected on problems A, C, D, E, F and the RPO ordering on problems B and G.

All but the first problem in the table above turned out to be strongly sensitive to the different strategies. Problem E was the hardest one: only three search strategies lead KBlab to find the solution in a reasonable time. The smallest components with LIFO strategy gave a very good result, whereas the FIFO strategy required a much longer time. The running time of the FIFO strategy on this problem was higher than that on problems B and D although in these two examples KBlab generated many more critical pairs, because equations generated in solving problem E involved very long terms.

The FIFO strategy yielded a result on all the listed problems, but it was always slower than the others strategies with the exception of problem F. The LIFO strategy halted only on problems A, C and G. Both strategies are very useful for experimentation: the former is a safe, exhaustive strategy; the latter is certainly not fair, but when it works it can yield very good results, as shown by examples C and G, where it yielded the fastest running time.

The ordered axioms strategy worked very well on example C but not in cases D, E, F. Moreover its behaviour on problem B was affected by the choice between FIFO and LIFO as strategy to solve conflicts among axioms having the same position in the ordering. The four smallest components and by size strategies behaved very similarly on examples A, B and C. The choice of FIFO or LIFO made a significant difference in problems D and F. Strategies in the LIFO family were faster on problem D, but they did not halt on problem F, because they are not fair. It is interesting to note that, the ordered axioms strategy, the only strategy which orders the terms and the axioms coherently, did not fare as well as we had expected.

Acknowledgements

KBlab has been developed at Dipartimento di Scienze dell'Informazione, Università degli Studi di Milano. Jieh Hsiang suggested several improvements to KBlab and its presentation and gave many interesting problems for testing. The experimentation done with KBlab while the first author was visiting Laboratoire de Recherche en Informatique, Université de Paris Sud at Orsay, contributed to improve it significantly: we thank Jean Pierre Jouannaud and Emmanuel Kounalis.

References

[1] L.Bachmair, N.Dershowitz, J.Hsiang – Orderings for Equational Proofs

In Proceedings 1^{st} Annual IEEE Symp. on Logic in Computer Science, 346–357, Cambridge, MA, June 1986

[2] L. Fribourg – A superposition oriented theorem prover

J. of Theoretical Computer Science, Vol. 35, 129–164, 1985

[3] L.Fribourg – A Strong Restriction to The Inductive Completion Procedure

In Proceedings 13^{th} Int. Conf. on Automata, Languages and Programming, Rennes, France, July 1986, Lecture Notes in Computer Science 226, 1986

[4] Glickfield, R.Overbeek – A Foray Into Combinatory Logic

J. of Automated Reasoning, Vol. 2, No. 4, Dec. 1986

[5] J.Hsiang, M.Rusinowitch – On Word Problems in Equational Theories

In Th.Ottman ed., Proceedings 14^{th} Int. Conf. on Automata, Languages and Programming, Karlsrhue, W.Germany, July 1987, Lecture Notes in Computer Science 267, 1987

[6] J.Hsiang, J.Mzali – SbREVE User's Guide

To appear as Technical Report L.R.I. Universitè de Paris Sud, Orsay, France

[7] G.Huet – A Complete Proof of Correctness of Knuth–Bendix Completion Algorithm

J. of Computer and System Sciences, Vol. 23, 11–21, 1981

[8] G.Huet, J.M.Hullot – Proofs by Induction in Equational Theories with Constructors

J. of Computer and System Sciences, Vol. 25, 239–266, 1982

[9] D.E.Knuth, P.Bendix – Simple Word Problems in Universal Algebras

In J.Leech ed., Proceedings of the Conf. on Computational Problems in Abstract Algebras, Oxford, 1967, Pergamon Press, Oxford, 263–298, 1970

[10] B.McCune, L.Wos – Some Fixed Points Problems in Combinatory Logic

AAR Newsletter

[11] A.Ohsuga, K.Sakai – Refutational Theorem Proving for Equational Systems Based on Term Rewriting

Technical Report COMP86–40, ICOT, 1986

[12] R.Smullyan – How to mock a mocking bird

Alfred A. Knopf, New York 1985