Well-Behaved Search and the Robbins Problem

William McCune*

Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, Illinois, 60439, U.S.A.

Introduction

The Robbins problem was solved in October 1996 [7] by the equational theorem prover EQP [6]. Although the solution was automatic in the sense that the user of the program did not know a solution, it was *not* a simple matter of giving the conjecture and pushing a button. The user made many computer runs, observed the output, adjusted the search parameters, and made more computer runs. The goal of this kind of iteration is to achieve a *well-behaved search*. Several of the searches were successful.

The purpose of this presentation is to convey some of the methods that have led to well-behaved searches in our experiments and to speculate on automating the achievement of well-behaved search. First, I give some background on the Robbins problem and its solution.

Boolean Algebra

The following set of equations, presented by E.V. Huntington in 1933 [4, 3], is a well-known basis for Boolean algebra.

$$x + y = y + x, \qquad \text{(commutativity)}$$
$$(x + y) + z = x + (y + z), \qquad \text{(associativity)}$$
$$\overline{\overline{x} + \overline{y}} + \overline{\overline{x} + y} = y. \qquad \text{(Huntington equation)}$$

The operations $t_1 + t_2$ and $\overline{t}$ are the ordinary addition and complementation operations of Boolean algebra. Existence of *zero* and *one* can be proved, and multiplication, if defined in the obvious way, has the appropriate properties.

The Robbins Problem

Shortly after Huntington's basis appeared, Herbert Robbins posed the problem of whether the Huntington equation can be replaced with the following equation.

$$\overline{\overline{\overline{x} + y} + \overline{x + y}} = y. \qquad \text{(Robbins equation)}$$

* Supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

Since then the problem has frequently been stated as follows: *Is every associative commutative (AC) algebra satisfying the Robbins equation a Boolean algebra?* A proof of the statement

$$\textit{Robbins} \Longrightarrow_{\text{AC}} \textit{Huntington}.$$

would clearly solve the problem and, furthermore, would show further that the operations in the Robbins equation are the ordinary addition and complementation operations of Boolean algebra.

The Winker Lemmas

While working on the Robbins problem in the early 1980s, Steve Winker formulated and proved (by hand) the following lemmas [9].

$$\left\{\begin{array}{l} \textit{Robbins} \\ \exists c \exists d (c + d = c) \end{array}\right\} \Longrightarrow_{\text{AC}} \textit{Huntington}.$$

$$\left\{\begin{array}{l} \textit{Robbins} \\ \exists c \exists d (\overline{c + d} = \overline{c}) \end{array}\right\} \Longrightarrow_{\text{AC}} \textit{Huntington}.$$

Attack on the Robbins Problem with EQP

I focused on trying to prove the statement

$$\textit{Robbins} \Longrightarrow_{\text{AC}} \left(\begin{array}{l} \exists c \exists d (c + d = c) \;\; \vee \\ \exists c \exists d (\overline{c + d} = \overline{c}) \end{array}\right)$$

with EQP, a theorem prover for equational logic [6]. EQP's main features are the following.

- The basic search method is similar to unfailing Knuth-Bendix completion.
- Associative-commutative unification and matching are available.

EQP, including the search parameters mentioned in the next section, is presented in more detail in [7] and [6].

Well-Behaved Search

In the experiments that led to the solution of the Robbins problem, I made many searches, observing and comparing the sequences of derived and retained equations and the statistics presented by the program. Many of the studies were quantitative, for example, running two searches for a fixed time and comparing the sets of retained equations, or running two searches until each retained 100 equations and comparing the times and the sets that were retained. Other studies were more qualitative, for example, finding a search that exhibited more variety in size or structure of retained equations, or avoiding searches that appeared to be wasting time in unproductive operations. Retaining completeness of the proof procedure was not a concern.

In the rest of this section, I shall go through some of EQP's important search parameters and indicate how they relate to well-behaved search.

Length Limit. The user can specify a limit on the length of equations. After a paramodulant is derived and simplified, if its length is greater than the limit, it is simply discarded. This parameter (which does not change during the search) is one of the most important in achieving well-behaved search. If the value is too great, too many of the retained equations will never enter the search, wasting memory and, more important, if added to the set of demodulators, wasting time required to simplify derived equations. However, it is not a matter of finding the smallest value that allows the search to progress, because larger equations can lead to quicker proofs. In fact, this was the case in the Robbins experiments [7].

With an initial length limit of 36, several combinations of parameters were tried; in each case, fewer than 300 equations were retained. Then the search continued for up to three days, with few or no additional equations. Although the search spaces were not exhausted, a greater length limit clearly was required. Values of 40, 50, 60, 70, and 80 were then used. The first proof was found with a value of 70; then a shorter proof, in longer time, was found with a value of 60.

Paramodulation Restrictions. I had available three restrictions on paramodulation inferences: *blocked paramodulation* [5], *basic paramodulation* [1, 8], and *the super-0 restriction* on AC unifiers [7].

Blocked paramodulation prevents paramodulation when the unifier is reducible by the current set of demodulators. The early experiments (and some previous experiments [6]) with blocked paramodulation indicated that for the Robbins domain, the benefit of reduced redundancy was small, the search space was not greatly changed, and the time required for the check was substantial. I therefore abandoned this restriction for the remainder of the Robbins experiments.

Basic paramodulation prevents paramodulation into terms that arise from substitutions in previous inference steps. (Note that in the EQP proof below, most of the paramodulation steps are into the Robbins equation; this is likely a result of the "basic" restriction.) Basic paramodulation does not require expensive checks, but it substantially reduces the number of derived equations. Two searches were run with a length limit of 36: one with basic paramodulation and one without. Approximately the same set of equations was retained, but the basic search retained it faster. This result led me to use basic paramodulation in later searches.

The super-0 restriction on AC unifiers is a heuristic that causes the program to use only simple AC unifiers. As with basic paramodulation, the check is inexpensive, and it greatly reduces the number of derived equations. A pair of searches with length limit 36, with and without the super-0 strategy, produced results similar to the basic paramodulation results, but more pronounced: nearly the same set of equations was retained, but much faster. Note that this is an incomplete restriction, with apparently severe and arbitrary pruning of the search.

A search with both basic paramodulation and the super-0 restriction (with length limit 36) showed that the combination would likely be more valuable than either method alone.

Size and Order of Inference Tasks. The user can choose from two methods to control the size of inference tasks: at each iteration of the main loop (1) select a *given clause* and generate all paramodulants with previous given clauses, or (2) select a *pair* of equations for application of paramodulation. To order the inference tasks, the user can specify the ratio of shortest-first selection and breadth-first selection for application of paramodulation. My decisions about these parameters were based more on previous experience than on experiments in the Robbins domain. With respect to size, I used method (2), which gives smaller inference tasks, because the number of AC unifiers can be enormous. For ordering the search, I mostly used a ratio of 1:1, that is, equal parts shortest-first and breadth-first. The important point is to *not* use strictly shortest-first selection or strictly breadth-first selection. Experiments run after the Robbins problem was solved showed that a ratio of 4:1 (shortest-first:breadth-first) would have been a better choice [7].

Extension Equations and Term Ordering. Given equation $t_1 = t_2$ in which t_1 has an AC symbol $+$ at its root, the extension equation $t_1 + x = t_2 + x$ is required in some situations. Extension equations can be disabled or delayed during the search. Several simple term-ordering methods are available. These two issues are ordinarily important to the search, but they were nearly irrelevant to the Robbins experiments. Almost all of the equations had the complement operation at the root of the larger side of the equation; extensions are irrelevant for such equations because the root is not AC. With respect to term ordering, almost all equations had the property that one side of the equation is a subterm (modulo AC) of the other side. All reasonable simplification orderings orient such equations in the obvious way.

(Of course, Robbins algebra, i.e., Boolean algebra, has many equations with $+$ at the root and without the subterm property. However, in all cases I have seen, the vast majority of the search to show that the Robbins equation is sufficient does not have any equations with $+$ at the root or without the subterm property. Such equations, for example, $x + x = x$ and $x + \overline{x} = y + \overline{y}$, are derived only after the key lemma $\exists c \exists d (c + d = c)$ is proved. See the EQP proof below.)

A Parameter on the Wish List. The EQP proof below contains only two simplification steps, yet the program spent more than 95% of its time simplifying equations. This seems like quite a waste of time (and an ill-behaved search). One of the main benefits of simplification is to control redundancy; however, I believe that adding *fewer* equations to the set of rewrite rules is an attractive approach, because less time will be spent rewriting, and other restriction methods such as the length limit will work to control redundancy. Unfortunately, EQP does not currently have any methods, aside from orientation, for selectively adding equations to the set of rewrite rules.

The Solution

The first successful search gave a proof of the second Winker lemma. The search parameters were a length limit of 70, basic paramodulation, the super-0 strategy, the pair selection method, and a selection ratio of 1:1. During the search, 49548 equations (i.e., critical pairs) were derived; during simplification of those derived equations, rewriting was attempted on 2612977 terms, and 5981 terms were rewritten. Of the simplified equations, 17663 were retained, all of which were oriented and became demodulators. Of the total search time, 0.1% was spent deriving equations (including all AC unification time), 74% was spent simplifying equations, 22% was spent finding existing equations that could be simplified with newly-adjoined demodulators, and 3% was spent deciding whether simplified equations were subsumed by existing equations. The search required about 30 megabytes of memory and 8 days on a (Sparc-5 class) UNIX computer.

Theorem. *Robbins* $\Longrightarrow_{\mathrm{AC}}$ $\exists c \exists d(\overline{c+d} = \overline{c})$.

Proof. I abbreviate $x{+}x$ as $2x$, $x{+}x{+}x$ as $3x$, and so on. The justification "$m \to n$" indicates paramodulation from the left-hand side of m into the left-hand side of n, and "simp:n" indicates simplification with n.

3	$\overline{\overline{(\overline{x}+y)}+\overline{(x+y)}} = y$	[Robbins]
5	$\overline{\overline{(\overline{(x+y)}+\overline{x}+y)}+y} = \overline{x+y}$	[3 → 3]
6	$\overline{\overline{(\overline{(\overline{x}+y)}+x+y)}+y} = \overline{\overline{x}+y}$	[3 → 3]
24	$\overline{\overline{(\overline{(\overline{x}+y)}+x+2y)}+\overline{(\overline{x}+y)}} = y$	[6 → 3]
47	$\overline{\overline{(\overline{(\overline{(\overline{x}+y)}+x+2y)}+\overline{(\overline{x}+y)}+z)}+\overline{(y+z)}} = z$	[24 → 3]
48	$\overline{\overline{(\overline{(\overline{x}+y)}+\overline{(\overline{x}+y)}+x+2y)}+y} = \overline{\overline{x}+y}$	[24 → 3]
146	$\overline{\overline{(\overline{(\overline{x}+y)}+\overline{(\overline{x}+y)}+x+3y)}+\overline{(\overline{x}+y)}} = y$	[48 → 3]
250	$\overline{\overline{(\overline{(\overline{(\overline{x}+y)}+x+2y)}+\overline{(\overline{x}+y)}+\overline{(y+z)}+z)}+z} = \overline{y+z}$	[47 → 3]
996	$\overline{\overline{(\overline{(\overline{(\overline{(\overline{x}+y)}+x+2y)}+\overline{(\overline{x}+y)}+\overline{(y+z)}+z)}+z+u)}+\overline{(\overline{(y+z)}+u)}} = u$	[250 → 3]
16379	$\overline{\overline{(\overline{(\overline{x}+x)}+3x)}+x} = \overline{\overline{x}+x}$	[5 → 996,simp:3]
16387	$\overline{\overline{(\overline{(\overline{(\overline{x}+x)}+3x)}+x+y)}+\overline{(\overline{(\overline{x}+x)}+y)}} = y$	[16379 → 3]
16388	$\overline{\overline{(\overline{(\overline{x}+x)}+4x)}+\overline{(\overline{x}+x)}} = x$	[16379 → 3]
16393	$\overline{\overline{(\overline{(\overline{x}+x)}+\overline{(\overline{x}+x)}+4x)}+x} = \overline{\overline{x}+x}$	[16388 → 3]
16426	$\overline{\overline{(\overline{(\overline{(\overline{x}+x)}+\overline{(\overline{x}+x)}+4x)}+x+y)}+\overline{(\overline{(\overline{x}+x)}+y)}} = y$	[16393 → 3]
17547	$\overline{\overline{(\overline{(\overline{(\overline{x}+x)}+\overline{(\overline{x}+x)}+4x)}+\overline{(\overline{(\overline{x}+x)}+3x)}+x)}+x} = \overline{\overline{(\overline{x}+x)}+\overline{(\overline{x}+x)}+4x}$	[146 → 16387]
17666	$\overline{\overline{(\overline{x}+x)}+\overline{(\overline{x}+x)}+4x} = \overline{\overline{(\overline{x}+x)}+3x}$	[24 → 16426,simp:17547]

Equation 17666 asserts the existence of an object c, namely $\overline{(\overline{x}+x)}+3x$, and an object d, namely $\overline{(\overline{x}+x)}+x$, such that $\overline{c+d} = \overline{c}$. Q.E.D.

Later searches gave proofs of $\exists c \exists d(c+d=c)$ and of the Huntington equation [7].

Automating the Quest for Well-Behaved Search

All of the search adjustments I have described have been by hand and ad hoc, but I believe that much of the process can be automated.

At the March 1995 Seminar on Deduction at Dagstuhl [2], I gave a short talk on automated deduction in algebraic geometry in which I mentioned some of the issues of well-behaved search. Afterward, Lincoln Wallen contributed some insightful comments on the need to formalize the problem of well-behaved search. Several other talks touched on the problem; and later in the seminar, Lincoln led a panel discussion on the topic. Such analysis and formalization of proof search are clearly important steps toward automating the achievement of well-behaved search.

Larry Wos's goal of a *self-analytical* automated reasoning system [10, 11], which can monitor its progress and dynamically adjust the search parameters, is also closely related to the topic of well-behaved search.

Input Files, Proofs, and EQP

Many input files and computer proofs from my Robbins experiments are available on the Web at the following location.

`http://www.mcs.anl.gov/home/mccune/ar/robbins/`

In particular, one can find there a graduated set of lemmas, including the Winker lemmas, that are useful for experimenting with well-behaved search. Also, the program EQP can be found at that location.

References

1. L. Bachmair, H. Ganzinger, C. Lynch, and W. Snyder. Basic paramodulation and superposition. In D. Kapur, editor, *Proceedings of the 11th International Conference on Automated Deduction, Lecture Notes in Artificial Intelligence, Vol. 607*, pages 462–476. Springer-Verlag, 1992.
2. W. Bibel, K. Furukawa, and M. Stickel, editors. *Dagstuhl Seminar on Deduction (9512), Report 110*, Schloss Dagstuhl, D-66687 Warden, Gemany, March 1995. IBFI GmbH.
3. E. V. Huntington. Boolean algebra. A correction. *Trans. AMS*, 35:557–558, 1933.
4. E. V. Huntington. New sets of independent postulates for the algebra of logic, with special reference to Whitehead and Russell's *Principia Mathematica*. *Trans. AMS*, 35:274–304, 1933.
5. D. Kapur, D. Musser, and P Narendran. Only prime superpositions need be considered in the Knuth-Bendix completion procedure. *J. Symbolic Computation*, 6:19–36, 1988.

6. W. McCune. 33 basic test problems: A practical evaluation of some paramodulation strategies. In Robert Veroff, editor, *Automated Reasoning: Essays in Honor of Larry Wos*, chapter 5. MIT Press, 1997. To appear.
7. W. McCune. Solution of the Robbins problem. *J. Automated Reasoning*, 1997. To appear.
8. R. Niewenhuis and A. Rubio. Theorem proving with ordering and equality constrained clauses. *J. Symbolic Computation*, 19(4):321–351, 1995.
9. S. Winker. Robbins algebra: Conditions that make a near-Boolean algebra Boolean. *J. Automated Reasoning*, 6(4):465–489, 1990.
10. L. Wos. *Automated Reasoning: 33 Basic Research Problems.* Prentice-Hall, Englewood Cliffs, NJ, 1988.
11. L. Wos. *The Automation of Reasoning: An Experimenter's Notebook with Otter Tutorial.* Academic Press, New York, 1996.