ARGONNE NATIONAL LABORATORY
9700 South Cass Avenue
Argonne, Illinois 60439

Experiments with the Hot List Strategy

by

Larry Wos

Mathematics and Computer Science Division

Technical Memorandum No. 232

October 1997

This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Comptuational and Technology Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

Contents

Experiments with the Hot List Strategy*

by

Larry Wos

Abstract

Experimentation strongly suggests that, for attacking deep questions and hard problems with the assistance of an automated reasoning program, the more effective paradigms rely on the retention of deduced information. A significant obstacle ordinarily presented by such a paradigm is the deduction and retention of one or more needed conclusions whose complexity sharply delays their consideration. To mitigate the severity of the cited obstacle, I formulated and feature in this report the *hot list strategy*. The hot list strategy asks the researcher to choose, usually from among the input statements, one or more clauses that are conjectured to play a key role for assignment completion. The chosen clauses—conjectured to merit revisiting, again and again—are placed in an input list of clauses, called the *hot list*. When an automated reasoning program has decided to retain a new conclusion *C*—before any other clause is chosen to initiate conclusion drawing—the presence of a nonempty hot list (with an appropriate assignment of the input parameter known as *heat*) causes each inference rule in use to be applied to *C* together with the appropriate number of members of the hot list. Members of the hot list are used to *complete* applications of inference rules and not to *initiate* applications. The use of the hot list strategy thus enables an automated reasoning program to briefly consider a newly retained conclusion whose complexity would otherwise prevent its use for perhaps many CPU-hours. To give evidence of the value of the strategy, I focus on four contexts: (1) dramatically reducing the CPU time required to reach a desired goal; (2) finding a proof of a theorem that had previously resisted all but the more inventive automated attempts; (3) discovering a proof that is more elegant than previously known; and (4) answering a question that had steadfastly eluded researchers relying on an automated reasoning program. I also discuss a related strategy, the *dynamic hot list strategy* (formulated by my colleague W. McCune), that enables the program during a run to augment the contents of the hot list. In the Appendix, I give useful input files and interesting proofs. Because of frequent requests to do so, I include for consideration challenge problems and questions whose answers are unknown, commentary on my approach to experimentation and research, and suggestions to guide one in the use of McCune's automated reasoning program OTTER.

1. Background, Motivation, and the Basic Problem

Of the areas of research, in my view, that which offers the greatest promise for increasing the power of automated reasoning programs focuses on strategy. My preference is for strategy that restricts reasoning, but clearly strategy that directs reasoning is also vital. This report features a strategy that perhaps belongs in neither category, namely, the *hot list strategy*; see [Wos97c] for relevant material. Its use rearranges the order in which conclusions are drawn. To enable researchers to estimate the potential of the hot list strategy, I present (in Section 3) the results of various experiments in lattice ordered groups, in Robbins algebra, and in logic calculi. Almost all of the data I provide is gathered from using a SPARCstation-10 to permit one to compare the successes and the failures. Some of the experiments rely heavily on the hot list strategy; some rely on it in combination with one or more other strategies; and, to permit comparisons to be made, some ignore its use entirely. To encourage researchers—especially mathematicians and logicians—to use the hot list strategy (as well as other strategies), I include various input files that are acceptable to the powerful automated reasoning program OTTER [McCune90,McCune91,McCune94], and I include interesting proofs as well. Because the experiments were conducted over a period of many months, different versions of OTTER were used. Where I recall, I shall note which version was used. As is almost inevitable, the results obtained with one version will not necessarily agree precisely with those obtained with a later version; an example of such concerns proof length, and even the proof itself may differ.

I note with great satisfaction that one of my long-term goals has finally been reached: Well-respected mathematicians and logicians who are *not* primarily interested in automated reasoning are nevertheless successfully using an automated reasoning program in their research. For example, K. Kunen (University of Wisconsin) has profitably studied aspects of group theory [Hart95,Kunen95]; R. Padmanabhan (University of Manitoba) with my colleague McCune has answered various questions focusing on algebraic geometry combined with universal algebra and is currently studying quasilattices and Boolean-like algebras [McCune96a]; and T. Jech (Penn State University) has obtained satisfying results in the foundations of mathematics [Jech94] and in combinatory logic.

This use by mathematicians and logicians nicely complements the use by some researchers who study deep questions (some of which are open) not only because the questions are intriguing, but also to increase the power of reasoning programs. Such a study can culminate in a breakthrough in the design of a program (as in the case of McCune's use of discrimination trees in his program OTTER) or can result in the formulation of a new inference rule or a new strategy. As noted, the focus in this report is on the hot list strategy.

Also in this report (in Section 4), I respond to the frequent request to offer challenge problems and questions whose answers are unknown. To provide some insights about how I have approached such problems and questions, I offer (in Section 3) a detailed treatment of various studies I have made with OTTER. To give one some insight into the choices I make from among various options offered by OTTER, I include commentary with the details of my experimentation. Finally, to increase the likelihood of success when using McCune's program, I include (in Section 5) diverse hints.

1.1. The Problem in Brief

The use of automated reasoning programs would almost certainly increase sharply if *simple* theorems were almost always proved quickly and if *deep* theorems were more in reach. In both contexts, one obstacle that is encountered is the need to use a conclusion as a hypothesis for drawing further conclusions, where the needed conclusion is quite complex. The complexity of the needed conclusion far too frequently causes a significant delay in the program's focusing on it in order to draw additional vital conclusions. The delay results from (1) the choosing as the focus of attention retained conclusions by less complex first and (2) the absence of any effective means for looking ahead and identifying good choices from among very complex conclusions. Overcoming the cited obstacle, or reducing its severity, is precisely the basic problem of concern in this report.

To address this basic problem, I formulated (in the mid-1980s) the *hot list strategy*. As it turned out, no implementation existed for many years, other than that in the program ITP [Lusk84}. For the record, I conducted little experimentation with ITP for various reasons, not the least of which was its being menu driven rather than file driven. (McCune, on November 2, 1993, installed in a new version of OTTER, OTTER 3.0, access to the hot list strategy.) This strategy asks the researcher to choose, from among the statements characterizing the question under attack, those that might be especially useful and merit revisiting again and again. As will be discussed here, with the hot list strategy, the chosen statements are automatically and immediately consulted by the program repeatedly, consulted for *completing* applications of inference rules and not for *initiating* applications.

1.2. Two Paradigms for the Automation of Reasoning

To set the stage for featuring the hot list strategy and for presenting evidence of its value, I first note that two distinctly different paradigms exist for automating logical reasoning. In the context of the basic problem to be addressed here, the key difference between the two paradigms concerns whether or not to accrue new deduced conclusions. Perhaps the most effective approaches where *no new information* is retained are based on Prolog technology. Among the more effective approaches in which *new information is accrued* are the *computational logic paradigm* and the *clause language paradigm* [Wos87,Wos92,Wos96a]. The former provides the basis for the Boyer-Moore program [Boyer79,Boyer98] (mainly used for program verification), and the latter (in the Argonne variant) provides the basis for McCune's OTTER.

Primarily because of my own experiments spread over more than thirty years, but also supported by the exchange of comments with researchers in automated reasoning, I currently have little doubt that

an automated attack on deep questions and hard problems virtually requires the accrual of information. Indeed, at this time (1997), the answer to the question or solution to the problem under consideration sometimes requires the retention of more than 200,000 deduced facts, relations, and lemmas. Perhaps the future will offer such powerful strategies for a reasoning program to use that far, far fewer conclusions must be retained to reach whatever goal is being sought; the set of support strategy has proved to be an important step in that direction. To possibly bring that future closer, I include various statistical information gleaned from the experiments on which this report is based, some without interpretation or conclusion, and some accompanied by conjecture. Without such data, I have always found it extremely difficult to evaluate new ideas.

With the retention of deduced conclusions, which in the Argonne paradigm are stored as *clauses*, the program needs a means for selecting where next to focus its attention. Although one occasionally finds it useful to instruct OTTER to conduct a breadth-first or first come first serve search (technically, a level-saturation search), far more often effectiveness is sharply increased by instructing the program to use *weighting* to decide where next to focus its attention. Use of the weighting strategy causes a program to choose as the next clause on which to focus that with the lowest weight (highest priority), where the weight of a clause is computed based on symbol patterns (*weight templates*) and assigned values supplied by the researcher or, if no template applies, by symbol count.

Whether the choice is level saturation or the weighting strategy, one encounters a significant obstacle to overcome or, at least, to reduce in severity. Specifically, a needed conclusion may have been retained but, because of its complexity or because of the lateness it was retained, the program has no means for choosing it to initiate applications of the inference rules in use.

1.3. A Motivating Example

If a deduced conclusion is retained late in a level saturation search or if it has high weight, and if consideration of that conclusion is crucial to reaching the desired goal (such as a proof of a theorem), then (quite likely) the automated reasoning program in use will either take an inordinate amount of CPU time to reach the goal or totally fail in that regard. The formidable, and sometimes insurmountable, obstacle presented by this situation is the large number of retained conclusions that must be considered *before* the key conclusion is finally chosen as the focus of attention to be used to initiate an application of an inference rule to in turn yield a vital step. Although weighting is usually far more effective than level saturation, the obstacle is still formidable even when the weighting strategy is in use, for, before the key conclusion is considered, many additional conclusions with smaller weight are retained and considered.

To spur my research that culminated in the formulation of the hot list strategy, the cited obstacle provided more than enough motivation at the general level. However, a particular theorem from algebra also provided motivation, at the specific level. The theorem in question asserts that commutativity can be proved in rings in which, for every element x, the cube of x equals x. I had repeatedly attempted (with various colleagues) to obtain a proof of the theorem by using one of the Argonne automated reasoning programs, under the condition that little or no guidance be provided by the user. (In the late 1970s, R. Veroff did in fact reach the sought-after goal by using some procedures designed for that purpose.) As a target, I had in mind a proof shown to me by S. Winker, a proof relying on various deduced lemmas including that asserting that $6x = 0$. The following mathematical proof and its counterpart using paramodulation illustrates well the basic problem under attack.

The mathematical proof begins with $xxx = x$ and substitutes the square of $v+w$ for x; in other words, instantiation is employed, an inference rule that is *not* offered by OTTER. (For the curious, I note that instantiation should not be offered by a program, for currently no means is known for wisely applying it; in other words, one encounters an important difference between mathematics and logic on one hand and automated reasoning on the other.) The left side becomes the cube of $v+w$, and the right side is simply $v+w$; call this equation (1). After expanding and simplifying with the hypothesis $xxx = x$, the left side becomes $v+w+vvw+vwv+vww+wvv+wvw+wwv$. Call the resulting equation (2). Equation (3) is obtained from equation (2) by subtracting $v+w$ from both sides. For equation (4), set $v = w$ and simplify with the hypothesis $xxx = x$, and one has the sought-after lemma $0 = 6x$.

From the viewpoint of automated reasoning with paramodulation as the inference rule, the corresponding proof (in clause notation) begins by paramodulating (*into* the clause equivalent of) $x(xx) = x$, with the focus on the *into term* xx, *from* the clause equivalent of left distributivity, with the focus on the left-hand argument. After appropriate demodulation, the result is the clause equivalent of equation (2). No clause equivalent of equation (1) is produced, other than the intermediate result obtained by applying the unification to the *into clause* of the preceding paramodulation. Of the various ways to obtain the clause equivalent of equation (3)—one of which will be seen (in Section 1.7) to be the use of the hot list strategy—assume that an extended form of cancellation is used by the program, perhaps through the use of demodulation at the literal level. At this point, one has an illustration of the obstacle under discussion: The clause equivalent of equation (3) has weight equal to 37, if measured purely in symbol count. Such a ''heavy'' clause will be delayed from consideration—if ever—for a substantial amount of CPU time, for many, many clauses will almost certainly be retained with weight less than 37. Therefore, the clause equivalent of equation (4), the desired lemma, will not be adjoined to the growing database of deduced conclusions for far, far too long.

The illustration just completed is neither rare nor unusual. Experiments repeatedly encounter the obstacle of the program's needing to consider some retained conclusion C whose complexity (weight) is so great that it remains (in the Argonne paradigm) in what is called list(sos) for far too long—possibly forever—before being chosen as the focus of attention. When this occurs, in too many cases, the program is prevented from access to those conclusions that would otherwise be deduced. The same can occur with a level-saturation approach, where the level of a clause is the problem rather than its complexity. (Of course, occasionally, another path to the ''children'' of C is found.) To address this basic problem of conclusion complexity resulting in the consumption of far too much CPU time before being chosen as the focus of attention, and motivated by the specific illustration of the lemma asserting that $6x = 0$, in the mid-1980s I introduced the concept of a *hot list*.

For the researcher who, like me, immediately wishes to know what happened when the hot list strategy was finally accessible to be applied to the $6x = 0$ lemma, I cite the following to provide a small taste of the type of experiment featured in this report. To avoid encountering my biases, I took an input file (supplied by my colleague Veroff) used to study rings in which $xxx = x$. When it was used to seek a proof of the $0 = 6x$ lemma *without* the hot list strategy, approximately 17 CPU-seconds sufficed. When the file was modified by adding the use of the strategy, approximately .3 CPU-seconds sufficed. For a second data point, I turned to the related problem in ring theory in which $xxx = x$ is replaced with $xxxxx = x$. A valuable lemma for proving commutativity in that context asserts that $30x = 0$. After the required change in the cited input file, *with* the hot list strategy, a proof of the lemma was found in approximately .8 CPU-seconds. When the strategy was omitted, no proof was found after approximately 408 CPU-seconds; the run was terminated by exhausting 20 megabytes of memory.

1.4. The Original Notion

As originally proposed, the hot list strategy was to be used when paramodulation was chosen as the inference rule; later, as discussed, the strategy was generalized by McCune to apply to all inference rules. (I have not reported this aspect of my research in print until now, for I lacked hard evidence of its value; only in late 1993 was the hot list strategy finally available for experimentation in OTTER, thanks to McCune.) My notion was to have the researcher choose equations thought to be crucial and place them in a special list, to be called the hot list. Then, when a clause C was deduced by applying paramodulation and the decision was to retain the clause, automatically and immediately paramodulation would be applied to C and each of the elements in the hot list. Thus, especially if the weight (complexity) of C were great, the program would still have access to some of the clauses descended from C, those deducible from C and (depending on the inference rule being applied) one or more members of the hot list.

For clauses to be included in the (input) hot list, I recommended (in the mid-1980s, and still do) those corresponding to the so-called special hypothesis, those clauses that are part of the hypothesis of the theorem but are not among the basic axioms. For example, in the case of the $6x = 0$ lemma, the clause equivalent of $xxx = x$ would be included in the (input) hot list. Then, immediately upon the decision to retain the clause equivalent of equation (3), that clause together with the clause equivalent of $xxx = x$ would be considered by paramodulation, and the program would deduce the clause

equivalent of equation (4). In other words, access to a hot list used as recommended would enable a program to overcome the obstacle of coping with a clause whose weight (complexity) equals 37, namely, equation (3). I also recommend for inclusion in the (input) hot list clauses that are conjectured to merit revisiting, again and again (to complete applications of inference rules, rather than to initiate applications).

1.5. Relation to Earlier Research

One perhaps sees in my original conception of the hot list strategy some of the aspects of the set of support strategy and some of the aspects of demodulation. Quite likely, in the following sense, these two concepts prepared the way for formulating the notion of a hot list. I focus first on the possible relevance (including the historical) of the set of support strategy to the hot list strategy.

The motivation for my formulation of the set of support strategy was also the study of a single (and very simple to prove) theorem: Commutativity can be proved for groups of exponent 2, those in which the square of x (for every element x) is the identity e. My notion (regarding the set of support strategy) was to restrict the applications of the inference rules in use and to force the search to key on information chosen by the researcher. To implement the strategy, the researcher places the *key information* in an input list, the initial set of support. Such is also the case for implementing the hot list strategy: The researcher places what is conjectured to be key information in an input list, the hot list. New conclusions for which the set of support strategy plays a role are recursively traceable to the initial set of support; new conclusions for which the hot list strategy plays a role are recursively traceable to the initial hot list. (The latter is not precisely the case for the dynamic hot list strategy; one must drop the word ‘‘initial’’.)

For a second way in which the two strategies are related, I have always recommended that the *special hypothesis* (clauses) be included in the initial set of support, and I typically recommend that such information also be placed in the initial hot list. For a third similarity, just as the newly retained conclusions in which the set of support strategy plays a role are added to the set of support list, so also can clauses be added to the hot list if the dynamic version of the hot list strategy, due to W. McCune, is in use (see Section 2 for details). Finally, the conclusions that are retained and that are traceable to the initial set of support can be used, without violating the set of support strategy, to deduce additional conclusions; such is also the case for clauses deduced with the hot list strategy, depending on the value assigned to the *heat* parameter.

Of the cited similarities between the two strategies, perhaps the most important concerns keying the program’s search on information selected by the researcher.

As for a key difference, a member from the set of support is chosen to *initiate* an application of an inference rule (when the set of support strategy is in use); in contrast, the members of the hot list come into play only *after* a clause has been chosen as the focus of attention to drive the program’s reasoning. Indeed, the members of the hot list are used only to *complete* an application of an inference rule. For a second important difference, the main object of the set of support strategy is to *restrict* the program’s reasoning; on the other hand, the object of the hot list strategy is to *rearrange* the order in which conclusions are drawn. Put another way, the hot list strategy (in a sense that is discussed in Sections 1.6, 1.7, 3.1, 3.4, and 5) enables a program to ‘‘look ahead’’.

Regarding the possible relevance of demodulation (which is the mechanism Argonne’s programs and many other programs use for simplification and canonicalization) to the hot list strategy, I had noted a dramatic improvement in efficiency due directly to automatically applying various equalities (demodulators) to each deduced conclusion. I finally conjectured that the automatic consideration by paramodulation of each newly retained clause with each member of a chosen set of equalities might also prove to be a powerful move. Of course, two constraints on the actions of the program must be relaxed: (1) rather than requiring that no instantiation of variables be permitted in the *into clause* (as in demodulation), full two-way unification must be permitted; and (2) rather than having the program wait until the newly retained clause is chosen as the focus of attention, the program must be allowed to immediately use it as one of the parents in the attempt to draw additional conclusions. Indeed, with the hot list strategy, members of the hot list are automatically and immediately considered by paramodulation, if in use, and by any other inference rule in use (as McCune suggested by way of a generalization)

with each newly retained clause, *before* another conclusion is chosen from list(sos) to be the focus of attention to drive the program's reasoning.

1.6. Intuitive View of the Hot List Strategy

In the following sense, one sees that the use of the hot list strategy enables an automated reasoning program to ''look ahead''. Purely for pedagogical reasons, I make various assumptions. Assume that paramodulation is the only inference rule in use and that a clause *A* has just been selected to be considered (by paramodulation) with each of the various clauses that have already been chosen as the focus of attention. Let *H* be a member of the hot list, and assume that the assignment to the appropriate input parameter (called *heat*) permits consideration of *H* with each new clause that the program decides to retain. Also assume that the program is choosing where next to focus its attention based purely on symbol count and that the weight (number of symbols) of *A* is 12. Let *C* be a clause with weight 25 that is deduced from *A* and some earlier-considered clause such that the program decides to retain *C*. Finally, assume that, prior to the decision to retain *C*, the consideration of *A* has resulted in the retention of ten new clauses each with weight 15.

Ordinarily, without the intervention of the hot list strategy, the program would not consider applying paramodulation to *C* and another clause until after focusing first on the ten newly retained clauses each with weight 15. Quite likely the consideration of *C* would be delayed further, for the focus on the weight 15 clauses would probably result in the retention of additional clauses of weight less than 25.

However—and here is how the use of the hot list strategy enables the program to look ahead—*before* another clause is chosen as the focus of attention, paramodulation is applied to *C* and *H*. If the application yields a clause *D* with weight 10 that is retained, then the program will have almost immediate access to the use of *D* for initiating applications of inference rules rather than waiting for *C* to be chosen as the focus of attention—if *C* is ever chosen. Indeed, as soon as *A* has completed its role as the focus of attention, *D* will be chosen to initiate applications of inference rules if all eligible clauses have weight greater than 10. In addition to the deduction of *D*, other low-weight clauses might be retained whose parents are *C* and *H*, and still others from *C* and some other member of the hot list.

If the size of the hot list is small, the researcher need not in general worry about the program being forced to cope with an avalanche of clauses of the type under discussion. In effect, the program looks ahead to deduce just those immediate descendants of *C* whose other parent is a member of the hot list.

Consistent with the formal definition of heat level given in Section 2, such a deduced clause *D* has *heat level* equal to 1. If the value the researcher assigns to the (input) heat parameter is 2, then *after* the program decides to retain a clause *D* with heat level 1 but *before* another clause is chosen as the focus of attention (in this case) paramodulation is applied to *D* and each member of the hot list, *before* that newly retained clause is used in its fullest to initiate applications of inference rules. Any clause that is so deduced has heat level equal to 2. Clauses of heat level 2 are immediate descendants of immediate descendants of a newly retained clause.

1.7. The Hot List Strategy for Replacing Other Procedures

In a limited way, one can use the hot list strategy in place of associative-commutative unification. For example, by including in the hot list a clause for associativity and a clause for commutativity and assigning the heat parameter the value 2, from each newly retained clause *before* another clause is chosen from list(sos) to be the focus of attention, the program will automatically apply paramodulation to the new clause together with that for associativity and also apply the inference rule to the new clause together with that for commutativity. For each newly retained clause, the two clauses that are so deduced will have heat level 1, and they will be retained depending on the other input parameters and subsumption and such. Then, because of the assignment of the value 2 to the heat parameter, those heat level-1 clauses that are retained will each be considered by paramodulation with associativity and also with commutativity. In the case under discussion, a limited form of associative-commutative unification is used. Also deduced at heat level 2 are clauses to which associativity has been applied twice and to which commutativity has been applied twice.

Use of the hot list strategy in the described manner can produce clauses early in a run that, because of the reassociation and commuting of terms, admit further canonicalization. By choosing the appropriate assignment of the heat parameter, one has control over the amount of AC-unification that is used. I find this alternative to AC-unification appealing, for I have always been wary of a general and full use of that form of unification. Indeed, a full use of AC-unification can drown a program in unwanted conclusions; in general, for practical considerations, restrictions must be imposed.

If associative unification without commutativity is desired, a clause for associativity is included in the hot list, and no clause for commutativity is included.

Use of the hot list strategy can also (in effect) partially substitute for access to linked inference rules [Veroff92,Wos84b]. For the example, the focus is on *linked paramodulation*, an inference rule that (from what I know) has not yet been implemented in any automated reasoning program. Consider the case in which the clause *C* is deduced, the decision is to retain *C*, *C* has high weight, and, were it not for the fact that a particular term *t* in *C* was left associated, paramodulation would apply to *C* and a clause corresponding to the special hypothesis with the *into term* being the right association of *t*. With linked paramodulation, one could use associativity as a link to right associate *t* in *C* to then permit the result to unify appropriately with the special hypothesis clause. If one was using the hot list strategy with associativity and the special hypothesis clause in the hot list, and if the heat parameter was assigned the value 2, then the decision to retain *C* would immediately trigger the application of paramodulation to *C* and the clause for associativity. Then, if the decision was to retain the result, immediately paramodulation would consider the reassociated version of *C* with the special hypothesis clause.

Linked paramodulation does not work precisely as does the hot list strategy. Among the differences is that concerning so-called intermediate clauses. Specifically, once the program begins an application of a linked inference rule, the weight of a clause that is *temporarily* deduced on the way to the linked conclusion is ignored. Linked paramodulation cannot produce an intermediate clause whose weight prevents the continuation of the application of linked paramodulation. In particular, in the example just discussed, the reassociation of the clause *C* because of using associativity as a link cannot prevent completion of the application of the linked inference rule because of the weight of the reassociated clause. In contrast, with the hot list strategy, the reassociated *C* must be retained in order to permit its consideration with the special hypothesis clause. (I am curious about the possible usefulness of considering each clause with members of the hot list *before* the decision concerning retention is made.) A second difference concerns the possible use of demodulation. The intermediate clauses resulting from a partial application of a linked inference rule are *not* subject to demodulation, but their correspondents *are* with the hot list strategy.

Regarding the earlier reference in Section 1.3 to the hot list strategy in the context of extended cancellation, one could proceed in the following manner. Clauses that function as nuclei and that capture various types of cancellation can be placed in the (input) hot list, and hyperresolution can be used as one of the inference rules. Then, when the program decides to retain a new clause, before another clause is chosen as the focus of attention to initiate applications of inference rules, the clause will be processed with cancellation of the type present in the hot list. One might find this alternative more attractive than either (1) waiting for the clause to be chosen as the focus of attention to then be considered with included clauses for cancellation or (2) using demodulation in some usual form or some nonstandard form.

One thus sees that the hot list strategy might prove useful in various ways to a researcher using an automated reasoning program. One naturally thinks of experiments that focus on obtaining a proof that is difficult to obtain, where the difficulty lies with apparent inaccessibility to some conclusion that is quite complex (if measured in symbol count). Of course, if the researcher has a conjecture that identifies such a conclusion, then an appropriate weight template can be included to cause the program to assign a small weight to the conclusion. In many cases, however, especially with an open question or a deep theorem, one has no good guess about the form of the needed proof steps. Of a slightly different nature is the experiment that concerns a proof in which two or more complex proof steps occur in succession. Perhaps the hot list strategy will prove useful in that context.

Regarding level saturation in place of weighting to direct the program's reasoning, the hot list strategy also enables the program to look ahead. Indeed, depending on the value assigned to the heat

parameter, use of the hot list strategy permits clauses of level $j+k$ to be deduced during the deduction of clauses of level j, where k is the value assigned to the heat parameter. Therefore, the problem of inaccessibility to clauses of too high a level can sometimes be overcome by appropriate use of the hot list strategy.

To begin to measure the power of the hot list strategy, I considered four contexts: (1) to sharply reduce the length of CPU time to complete a proof; (2) to produce a proof of a theorem that previously had been virtually out of reach; (3) to find proofs more elegant than previously known; and (4) to answer a question whose answer had steadfastly eluded researchers relying on an automated reasoning program. In Section 3 and its subsections, I shall present some relevant data drawn from experiments with OTTER. What is next in order at this time is some formalism and a discussion of parameters and options.

2. Formalism, Parameters, and Options

In this section, I employ one or more of the notations acceptable to McCune's program OTTER. When I use the phrase ''clause or its equivalent'', I am *not* restricting the definitions and terminology to OTTER-like programs. Rather, I intend that the appropriate adjustment(s) be made when the program in use requires input in some other form. Often, I use the term ''clause'' to mean ''clause or its equivalent''.

Definition, initial and extended hot list. The initial hot list is a (possibly empty) set of clauses (or their equivalent) selected by the researcher and included in the input. Depending on the inclusion of appropriate options, the initial hot list can be extended by adjoining new members to it during a run. Each member of the hot list (initial and extended) is eligible for automatic and immediate consideration to *complete* applications of the inference rules in use, with the requirement that the application of one of those inference rules be *initiated* by focusing on a clause (or its equivalent) that the program has decided to retain. In particular, no inference rule is permitted to apply to a set of clauses all of which are members of the hot list.

Definition, hot list strategy. The hot list strategy, which relies on the hot list, enables the researcher to specify certain facts as possibly key to success, causing the program to revisit them repeatedly in the context of completing the application of an inference rule.

With certain reservations, a glance at the definition of an initial hot list correctly suggests that the actions of such a hot list are somewhat reminiscent of the actions of an input set of demodulators—especially when paramodulation is in use; see Section 1.5. One key difference rests with permission to apply full (two-way) unification (when a hot list element is involved), in contrast to the requirement that demodulation allows variable instantiation only in the demodulators. A second difference concerns the consideration of members of the hot list only *after* the program decides to retain a new conclusion, in contrast to the consideration of input demodulators for each deduced conclusion *before* any decision to retain or not to retain is made. A third important difference focuses on the ''eligibility'' of members of the hot list, which implies that an additional condition (that I immediately discuss) must be met; demodulation is not so constrained. For the additional condition, I need the following definition and the introduction of an input parameter that governs the use of the hot list.

Definition, heat level. The heat level of a clause is 0 if and only if no clauses of the hot list participate in the application of an inference rule; the heat level of a clause is 1 if and only if (1) clauses from the hot list participate and (2) the heat level of the clause initiating the application of the inference rule is 0; for $n \geq 2$, the heat level of a clause is n if and only if the heat level of the clause that initiates the application of the inference rule is $n-1$.

Regarding the eligibility of the members of the hot list, the (input) parameter known as *heat* must be assigned a value greater than or equal to 1 for members to be eligible for use. In other words, permission must be given to deduce clauses with heat level greater than or equal to 1. To instruct OTTER to attempt to deduce clauses with heat level 1, one adds a single command to the input file. If the value in the command of the following type is equal to 1, *after* OTTER decides to retain a newly generated clause A but *before* another clause is chosen as the focus of attention (to drive the program's reasoning), each inference rule in use is applied to A (as if A were the focus of attention) and the appropriate number of clauses H in the hot list, where that number is determined by the inference rule being

applied.

```
assign(heat,1).
```

The default of the parameter heat is 1.

For example, if paramodulation is being applied, then *A* is considered with each *H* in the hot list; if hyperresolution is being applied, then, consistent with the requirements of nucleus and satellites, all subsets of clauses from the hot list are considered with *A*. Any clause *B* deduced from *A* and one or more clauses *H* is treated as all deduced clauses are treated (with regard to subsumption, weighting, demodulation, and the like). If such a clause *B* is used in a proof, the proof will show for that clause (heat=1), meaning that its heat level is 1.

To enable OTTER to deduce and possibly use clauses whose heat level equals 2, one modifies the preceding command to be the following.

```
assign(heat,2).
```

With this modified command, *after* OTTER decides to retain a newly generated clause *B* whose heat level equals 1 but *before* another clause is chosen as the focus of attention, each inference rule in use is applied to *B* (as if *B* were the focus of attention) and the appropriate number of clauses *H* in the hot list, where that number is determined by the inference rule being applied. As expected, any clauses that are deduced whose heat level equals 2 are treated as all deduced clauses are treated. Of course, one can assign to the heat parameter values greater than 2. An assignment of the value 0 instructs the program *not* to consult the hot list.

Before continuing the discussion of parameters and options that govern the use of the hot list strategy, I pause for a brief illustration. The example concerns Meredith's axiom (given shortly) and the theorem that says it is a single axiom for two-valued sentential (or propositional) calculus; see [Meredith53]. (I give Meredith's proof in the Appendix.) Later in this report (in Section 4.1), I revisit this theorem and offer it as a challenge problem for researchers; the theorem has never been proved by an automated reasoning program without substantial guidance. The following is taken from one of my numerous unsuccessful attempts to prove the theorem in a single run and without guidance from the researcher. In the input file I give later for this challenge problem one finds appropriate conclusions to deduce to show that Meredith's axiom does indeed axiomatize (by itself) this area of logic. Specifically, I use as the goal known axiom systems for two-valued sentential calculus. For notation that is accepted by OTTER, I use ''|'' for logical **or**, ''-'' for logical **not**, and the predicate *P* that can be interpreted as ''provable''. The function *i* can be interpreted as ''implication'' and the function *n* as ''negation''. (When a line contains a ''%'', the characters from the first ''%'' to the end of the line are treated by the program as a comment.)

```
assign(heat,3).
list(hot).
% Following is for condensed detachment.
-P(i(x,y)) | -P(x) | P(y).
% Following is Meredith's single axiom.
P(i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x)))).  % CN-CAM
% Following were proved in temp.otter3.meredith.hot.out
P(i(x,i(y,x))).
P(i(i(i(x,y),z),i(y,z))).
P(i(x,i(n(x),y))).
P(i(n(n(x)),x)).
P(i(n(n(x)),x)).
P(i(i(i(x,y),z),i(n(x),z))).
P(i(x,n(n(x)))).
P(i(x,n(n(x)))).
P(i(i(n(x),x),x)).
P(i(i(x,i(x,y)),i(x,y))).
end_of_list.
```

As one sees (from the comment, signified by the use of ‘‘%’’), the inference rule used in the study is condensed detachment, used by Kalman in his landmark study of equivalential calculus [Kalman78,Kalman83], captured (for this area of logic) by the cited clause together with the use of hyperresolution. No other inference rule is used in the study. Typically, in such investigations, the only clauses used to initiate applications of an inference rule are unit clauses. Therefore, for the hot list strategy to be usable, one must include at least one nucleus in the hot list; the cited nucleus is the only such clause in the study. After all, as commented earlier, with the hot list strategy, all but the initiating clause for the application of each inference rule must be members of the hot list. Similarly, because the nucleus contains two negative literals, the hot list must also contain at least one other positive clause to permit the use of the hot list strategy to complete applications of condensed detachment. In the given example, I included in the hot list Meredith's axiom and various members of known axiom systems, each of which had been proved in a prior experiment. The assignment of the value 3 to the heat parameter instructs OTTER to attempt to deduce clauses with heat level less than or equal to 3. Whether any clauses that are thus deduced are retained depends, of course, on other parameters, such as the max_weight parameter (which places an upper bound on the weight of a retained clause). Having completed the illustration, I now return to parameters and options relevant to the hot list strategy.

In addition to choosing (if any) which clauses to place in list(hot) and which value to assign to the heat parameter, with a command of the following type, one can instruct OTTER to attempt to dynamically adjoin new clauses to the hot list *during the run*.

```
assign(dynamic_heat_weight, 20).
```

With the command as given as part of the input, OTTER will, during a run, adjoin to list(hot) any clause that (1) the program has decided to retain and (2) the program has assigned a weight less than or equal to 20. To answer a natural question, clauses adjoined to list(hot) during a run must have an assigned weight less than or equal to the max_weight currently in use. (Incidentally, members of the hot list are not subject to subsumption or, for that matter, to demodulation and the like.)

Regarding the dynamic hot list strategy in conjunction with other strategies, I offer the following for another means for attacking questions: If one is using the *resonance strategy* [Wos95a,Wos96c] (briefly reviewed in Section 3.2) with the *resonators* assigned, say, a weight of 2, I have found that an assignment to dynamic_heat_weight of the value 2 often proves advantageous. Briefly, a resonator is a formula or equation that, by treating its variables as indistinguishable from each other, provides patterns to direct a program's search; a resonator is *not* an added fact or lemma and does not take on any truth value.

This aspect of OTTER (of being able to dynamically adjoin new clauses to the hot list) has some of the flavor of self-analyticity (see Section 13.4 of [Wos96a]), and some of the flavor of the set of support strategy (see Section 1.5). Regarding the latter (somewhat vague) similarity, one expects or intends that clauses dynamically adjoined to list(hot) play a key role in a program's attack on the question or problem under study, just as one wishes, ideally, that clauses dynamically adjoined to list(sos) play a key role. Of course, as experimentation repeatedly shows, the ideal case (for the set of support strategy) is not even approximated: Typically, a few CPU-minutes suffices to produce a large and growing list(sos), many of whose members will *never* be chosen as the focus of attention to drive the program's reasoning. The most common cause for a clause not being chosen as the focus of attention is its high weight or complexity. Perhaps in the future, this deficiency will be sharply reduced by having the program automatically (and possibly self-analytically) move certain clauses from list(sos) to list(usable) *before* they are chosen as the focus of attention; see Section 13.4 of [Wos96a]. List(sos) is the name of the list of clauses that have *not* yet been chosen as the focus of attention but are recursively traceable to the initial set of support or were in the initial set of support. List(usable) consists of the input clauses that were part of the problem description but not placed in the initial set of support and also the clauses that were selected from list(sos) to be the focus of attention to drive the program's reasoning. However, an immediate move of a clause to list(usable) will permit its use only for inference rule completion, not for inference rule initiation.

As for proven recommendations and relevant data—the latter given in the next section—I can give less of the former than of the latter, for I have only begun to heavily experiment with the hot list strategy. I recommend that an input clause placed in list(hot) also be placed in some other list

(although OTTER does not require this move). (Among the exceptions is that discussed in the context of cancellation; see Section 1.7.) For example, if an input clause would ordinarily be included to *complete* the application of an inference rule rather than *initiate* the application, then I recommend that, if the clause is placed in list(hot), it also be placed in list(usable). As an aside, clauses that one suspects are best used to *complete* rather than *initiate* inferences belong, in my view, in list(usable). As another aside, consistent with my view, I conjecture that the effectiveness of an automated reasoning program would be increased if, when such a (completion) clause is retained, it were immediately placed in list(usable) rather than being placed in list(sos); this option is not offered by OTTER or, for that matter, from what I know by any program.

I would place the following clause (used to capture condensed detachment in the presence of the inference rule hyperresolution, when for example studying two-valued sentential calculus) both in list(usable) and in list(hot), were I using the hot list strategy.

-P(i(x,y)) | -P(x) | P(y).

The cited clause is best used to complete applications of an inference rule, and not to initiate them. Were I using OTTER to apply the hot list strategy to study rings in which the cube of (every element) *x* is *x*, I would place the clause equivalent of *xxx* = *x* both in list(sos) and in list(hot). I would take this action even though such a clause is best used to initiate applications of an inference rule, rather than complete an application.

For a global recommendation for using the hot list, I suggest including in it the clauses that correspond to the *special hypothesis* of the proposed theorem under attack; such clauses also, in my view, are wisely placed in list(sos). For a related global recommendation, I suggest the hot list consist of those equations from the input set of support having eight or fewer symbols (ignoring parentheses and commas) whose right-hand argument is a single symbol, constant or variable. Of course, I have in mind that predicates, functions, and the like are represented with single letters. The hot list can also be augmented with all similar ''short and simple'' clauses taken from the usable list. For inclusion in the (input) hot list, I also recommend clauses that one conjectures merit revisiting again and again.

Regarding recommendations for assignments for the respective values for the heat and the dynamic_heat_weight parameters, I can only suggest experimentation and a glance at some of the experiments I feature next in this report; see also [Wos95b,Wos96a,Wos96b,Wos96c] and especially [Wos97a,Wos97b].

3. Successes, Failures, and Experiments

At this point, I present evidence of the power of the hot list strategy by discussing various experiments, comparing results *with* and *without* the use of this strategy. As one should virtually demand, because of the complexity of mathematics and logic, not all of the evidence is on the positive side. I do not adhere strictly to a historical account. Nor do I offer what amounts to an experimenter's notebook—although some such aspects are in fact present in this report. Instead, I offer a small taste of what one might expect with the use of the hot list strategy, focusing on the four contexts cited in Section 1.7. Some of the experiments I present here focus on combining the use of the hot list strategy with other strategies of interest.

I also provide some insight into my approach to experimentation and research and illustrate some of the versatility of McCune's program OTTER. One sees from the data how complicated experimentation is. For example, sometimes an experiment strongly suggests that the hot list strategy is just what is needed, and sometimes the experiment indicates the strategy offers little. Rather than disappointment or puzzlement, I conclude for now that far more understanding and insight are needed regarding wise choices from among the numerous options offered by OTTER. Also, just for the record, I believe that the depth and complexity of mathematics preclude the existence of *any* uniformly powerful approach to attacking theorems, conjectures, and open questions—by a reasoning program, or by a person. Indeed, one should expect that the results will vary widely when one fixes the combination of strategies to use and then selects the problems for study from diverse areas.

3.1. Lattice Ordered Groups

The focus in this subsection is on a theorem concerned with lattice ordered groups. The theorem was brought to my attention by I. Dahn at the 1994 QED workshop. The theorem (which I denote by *LOGT*1 for Lattice Ordered Groups Theorem 1) asks one to prove that, for each element *x* in a lattice ordered group, *x* is equal to the product of its *positive part pp*(*x*) and its *negative part np*(*x*), where 1 is the identity of the group, *pp*(*x*) = the union of *x* and 1, and *np*(*x*) = the intersection of *x* and 1. The following (in yet another notation acceptable to OTTER) gives the axioms, needed definitions, and various members of a complete set of reductions for groups.

Regarding the notation, in addition to the preceding, *i* denotes inverse, *u* union, and *n* intersection. (The anomaly of having associativity of both union and intersection expressed as they are results from the manner in which I was given the problem; OTTER, when processing the input, interchanges their respective arguments so that the left-associated argument is on the left.) The significance of the line of dashes in the following will become clear when I focus on the set of support strategy.

```
x = x.
(x*y)*z = x* (y*z).
1*x = x.
x*1 = x.
i(x)*x = 1.
x*i(x) = 1.
i(1) = 1.
i(i(x)) = x.
i(x*y) = i(y)*i(x).
n(x,x) = x.
u(x,x) = x.
n(x,y) = n(y,x).
u(x,y) = u(y,x).
n(x,n(y,z)) = n(n(x,y),z).
u(x,u(y,z)) = u(u(x,y),z).
-----------------------------------
u(n(x,y),y) = y.
n(u(x,y),y) = y.
x*u(y,z) = u(x*y,x*z).
x*n(y,z) = n(x*y,x*z).
u(y,z)*x = u(y*x,z*x).
n(y,z)*x = n(y*x,z*x).
pp(x) = u(x,1).
np(x) = n(x,1).
pp(a)*np(a) ! =  a.
```

I had learned from Dahn that the theorem had been proved in 30 CPU-seconds on a SPARCstation-10 by a program called Discount. (Actually, in addition to the SPARCstation-10, simultaneously two other computers were used, each a SPARCstation-2; after 30 CPU-seconds, Discount announced the theorem proable, but an additional 90 CPU-seconds was required to return the proof.) I also learned that Discount relies on parallel processing, where the various processors compare notes, functioning as a ‘‘team’’. Especially since I had never attempted to study lattice ordered groups, the problem was particularly appealing. Because using OTTER is rather like having access to an assemblage of eager and sometimes erudite graduate students—this program offers numerous approaches from which to choose—I was not at all sure about how precisely to start the attack with the goal of eventually discovering an effective approach.

For my study of *LOGT*1 with OTTER, my colleague McCune in part paved the way; he chose a Knuth-Bendix approach with the following symbol ordering.

```
lex([1,a,u(_,_),n(_,_),*(_,_),i(_),pp(_),np(_)]).
```

McCune assigned max_weight the value 15 (for a bound on the complexity of retained conclusions,

measured in symbol count) and assigned the value 4 to the *pick_given_ratio*. By assigning the pick_given_ratio the value 4, OTTER is instructed to focus on four conclusions (from among those retained) based on their complexity (the weighting strategy), one by first come first serve (level saturation), then four, then one, and the like. (McCune is the author of the *ratio strategy* [McCune94,Wos96a], which relies, of course, on the pick_given_ratio and which we both used to attack the theorem under discussion.) Finally, possibly influenced by the approach OTTER takes when instructed to use the autonomous mode, McCune placed all clauses in list(sos), including that for reflexivity. By doing so, he (in effect) instructed OTTER *not* to use the set of support strategy. In all of the experiments I report in this subsection, I included in list(passive) the negations of the deduced steps of McCune's proof, 32 steps not counting the first step produced by demodulating the input clause corresponding to assuming the theorem false. I did so to monitor the program's progress; elements of this list are consulted by the program only for forward subsumption and for unit conflict.

I began the attack (with Experiment 1) by following McCune's lead, except that (through accident) I assigned the value 6 to the pick_given_ratio. The object was to begin gathering evidence pertinent to the possible strength and the possible weakness of the hot list strategy. So that I would have a target, I ran all of my experiments on a SPARCstation-10, as was the case for Dahn's use of Discount. Experiment 1 succeeded, producing a 45-step proof of level 12, where the computation of length and level includes (if such occurs) demodulation of input clauses but excludes the binary-resolution step that produces $F (**false**) or the empty clause. The proof required approximately 8607 CPU-seconds to complete, relying on (retained) clause (32673). To obtain unit conflict, the program chose as the focus of attention 5283 clauses. The run was not terminated immediately upon the completion of the sought-after proof but, instead, by completing the preassigned number of proofs, namely, 36. At the termination of the run, after choosing as the focus of attention 5990 clauses, 18,111 clauses were retained and 1,567,363 were generated.

By way of comparison, McCune's success (with the pick_given_ratio assigned the value 4) required 19,280 CPU-seconds (on a SPARCstation-2, roughly half as fast as a SPARCstation-10). His proof (which I give later, in the Appendix) has length 33 and level 10, and was completed with the choice as the focus of attention of 5603 clauses, with the retention of 17,645, and with the generation of 14,170,591. (Note that, in general, a smaller value assigned to the pick_given_ratio results in the selection as the focus of attention of more clauses with higher weight, or complexity, which may explain the reduction in the proof length and the increase in CPU time.)

For Experiment 2, I made a single change from Experiment 1: I used the set of support strategy. Of the clauses given (earlier in this subsection) to characterize the theorem under attack, I placed in the (initial) set of support those positive clauses found after the line of dashes. The choice of the cited clauses was based on many years of experience and on the conjecture, made in the late 1960s, that paramodulation often benefits from placing in list(sos) interesting axioms. My reason for not immediately introducing the use of the hot list strategy rests with two factors: first, the conjecture that a sharp reduction in CPU time to complete the desired proof would most likely require the use of the set of support strategy; and second, a preliminary evaluation of the power of the hot list strategy suggests the need for data *with* and *without* its use. Experiment 2 did complete a proof in approximately 8450 CPU-seconds with the retention of clause (31961), with length 46 and level 14. The proof was completed after choosing as the focus of attention 5145 clauses. Again the run was terminated by reaching the assigned limit (36) of completed proofs. When the run was terminated, after choosing as the focus of attention 5148 clauses, 16,405 were retained and 1,531,447 were generated.

The rather small (measured as a percentage) reduction in CPU time when comparing Experiment 1 with Experiment 2 gave me added incentive to move in Experiment 3 immediately to the use of the hot list strategy. I therefore had to choose the assignment of the (input) heat parameter and also choose which input clause to place in list(hot). I rejected basing my choices on an examination of proofs found in Experiments 1 and 2, for I had the goal of increasing the likelihood of objectivity and the goal of illustrating how a researcher rather new to the use of an automated reasoning program might profitably proceed. No doubt biased by more than thirty years of experimentation, my intuition weakly suggested that the attack would benefit from (in effect) recursing some when a newly deduced clause was considered by the hot list. I thus chose 3 as the assigned value for the heat parameter; other assignments will be discussed later. Regarding which clauses to place in list(hot), I reasoned that if I

ordinarily recommend the inclusion of the clauses that correspond to the special hypothesis, then all of the clauses in the (initial) set of support (except, of course, the negative clause) would be an interesting choice for the contents of list(hot).

Experiment 3 produced a proof in approximately 3142 CPU-seconds with the retention of clause (23157), with length 34 and level 16. The proof was completed after choosing as the focus of attention 3093 clauses. Again the run was terminated by reaching the assigned limit (36) of completed proofs. When the run was terminated, after choosing as the focus of attention 3096 clauses, 11,979 (of which 1772 were hot) were retained and 1,439,175 (of which 55,341 were hot) were generated. (A ''hot clause'' is a clause in which its history contains at least one clause in the hot list.) I was encouraged regarding the possible power of the hot list strategy.

For Experiment 4, in the spirit of imitating AC-unification, I chose to expand the contents of list(hot) by including the two clauses for commutativity and the three for associativity of the various functions. Experiment 4 produced a proof in approximately 9596 CPU-seconds with the retention of clause (45310), with length 45 and level 17. The proof was completed after choosing as the focus of attention 5288 clauses. Again the run was terminated by reaching the assigned limit (36) of completed proofs. When the run was terminated, after choosing as the focus of attention 5288 clauses, 17,527 (of which 14,179 were hot) were retained and 13,099,457 (of which 194,476 were hot) were generated.

My reaction to the apparent setback was to conjecture that either (1) because of the nature of its arguments, associativity in the hot list causes too many terms to expand or (2) additional input clauses adjoined to the hot list would remedy the situation. To decide which new clauses to adjoin, I either behaved like a good researcher, or I cheated—depending on one's viewpoint. Specifically, I began by briefly examining the proof obtained by McCune, discovering that the last step was obtained from that preceding it and the clause for left inverse. I next noted that the next to the last step of the proof he obtained has weight 14 (measured in symbol count), which would likely delay its being chosen as the focus of attention to deduce the last step. I then turned to an examination of the proof produced by Experiment 1, noting that more than 6800 CPU-seconds elapsed between the completion of a proof of the next to the last step of McCune's proof and a completion of the desired proof. That examination also showed that the correspondent of the next to the last step together with the correspondent of *right* inverse was considered by paramodulation. Based on the various observations and discoveries just cited, I decided for Experiment 5 to include in list(hot) the clauses for left and also for right inverse, and otherwise proceed as in Experiment 4.

Experiment 5 produced a proof in approximately 1981 CPU-seconds with the retention of clause (23157), with length 59 and level 17. The proof was completed after choosing as the focus of attention 2252 clauses. Again the run was terminated by reaching the assigned limit (36) of completed proofs. When the run was terminated, after choosing as the focus of attention 2394 clauses, 13,000 (of which 10,538 were hot) were retained and 1,285,846 (of which 99,503 were hot) were generated. My research was again making progress, and it was also producing additional evidence of the possible power of the hot list strategy.

For Experiment 6, because of the suspicion that associativity (for any function) might be far more expensive than it was worth, I decided to remove the three clauses for that law from the hot list, and otherwise proceed as in Experiment 5. My suspicion rested on the observation that the nature of the arguments of associativity encourages terms to expand into longer terms. Experiment 6 produced a proof in approximately 379 CPU-seconds with the retention of clause (7539), with length 44 and level 16. The proof was completed after choosing as the focus of attention 973 clauses. The run was terminated by entering the interactive mode and ''killing'' the run; 35 proofs had been completed. When the run was terminated, after choosing as the focus of attention 1559 clauses, 10,738 (of which 8451 were hot) were retained and 1,269,753 (of which 97,870 were hot) were generated.

Because I was at this point quite satisfied with my choice of the members of the hot list, I decided to investigate what would occur if I made a single change from Experiment 6 to Experiment 7, that of assigning the value 2 rather than the value 3 to the heat parameter. Experiment 7 produced a proof in approximately 347 CPU-seconds with the retention of clause (7739), with length 42 and level 18. When in Experiment 8 the heat parameter was changed to 1 (with all else unchanged from Experiment 6), a proof was obtained in approximately 370 CPU-seconds with the retention of clause (7696);

the length is 59 and the level is 18. For Experiment 9, I took the input for Experiment 8 and added to the hot list the associative laws for multiplication, union, and intersection. Here I had minute evidence of the possible value (from the viewpoint of CPU time and more) of the inclusion in the hot list of associativity: A proof was found in approximately 364 CPU-seconds with the retention of clause (7712), where the length is 58 and the level is 18. Perhaps the cost of including clauses for associativity depends dramatically on the value chosen for the heat parameter.

To examine the possibility of giving the inexperienced researcher a simple rule to follow for choosing clauses for the hot list (as was done in Experiment 3) and for choosing an assignment for the heat parameter, I conducted Experiment 10. I chose for the hot list precisely the positive clauses in the initial (input) set of support and assigned the heat parameter the value 1. Experiment 10 produced a proof in approximately 3148 CPU-seconds with the retention of clause (23565), with length 32 and level 13. The proof was completed after choosing as the focus of attention 3120 clauses. Again the run was terminated by reaching the assigned limit (36) of completed proofs. When the run was terminated, after choosing as the focus of attention 3122 clauses, 12,187 (of which 1714 were hot) were retained and 4,670,281 (of which 61,980 were hot) were generated. I experienced satisfaction at the bit of evidence supporting the use of a simple rule for choosing the hot list clauses (those positive clauses in the initial set of support) and for choosing an assignment for the heat parameter (namely, 1). I was more than pleased at discovering that OTTER had found a proof shorter than any I had seen, specifically, a proof of length 32 (given in the Appendix).

To gain some data on the need for the set of support strategy and to determine, in particular, whether the use of the hot list strategy would suffice by itself to produce the cited improvement in effectiveness (measured in CPU time), I conducted six additional experiments, 11 through 16. In all six, I placed all clauses in the initial set of support, (in effect) thus avoiding the use of this powerful restriction strategy. In Experiments 11 through 13, the hot list consisted of the members of the set of support used in Experiment 10 (the positive clauses found after the line of dashes near the beginning of this subsection) augmented by the clauses for commutativity of union and of intersection and the clauses for inverse. For Experiments 14 through 16, I adjoined to the hot list (used in Experiments 11 through 13) the clauses for associativity of union and of intersection. In order and not with careful planning, I assigned the heat parameter the value 3, 2, 1, 2, 1, and 3. Therefore, in a cursory manner, I was also studying some of the same aspects as studied in the first ten reported experiments.

Experiment 11 produced a proof in approximately 1523 CPU-seconds with the retention of clause (21879), with length 44 and level 15. Experiment 12 produced a proof in approximately 385 CPU-seconds with the retention of clause (7880), with length 51 and level 17. Experiment 13 produced a proof in approximately 400 CPU-seconds with the retention of clause (7624), with length 48 and level 18. Experiment 14 produced a proof in approximately 583 CPU-seconds with the retention of clause (11670), with length 47 and level 15. Experiment 15 produced a proof in approximately 462 CPU-seconds with the retention of clause (7711), with length 48 and level 16. Experiment 16 produced a proof in approximately 1689 CPU-seconds with the retention of clause (21542), with length 58 and level 17.

A glance at the results of Experiment 11 through 16 suggests that the use of the hot list strategy can sharply increase the effectiveness (measured in CPU time) of an automated reasoning program. Indeed, the set of support strategy, often so vital to assignment completion, was not used, and yet the most unsatisfying result (that of Experiment 16) was far superior to the result produced by Experiment 1 in which the hot list strategy was *not* used.

In no way am I implying that the combination of the set of support strategy and the hot list strategy offers little; indeed, as shown in some of the reported experiments, I did in fact use this combination of strategy profitably. If confusion is present, it rests on the results where success was obtained with the hot list by itself. Future research will, I hope, produce some guidelines for deciding when which combination of strategies is most effective.

Nor do I imply that either strategy can in general replace the other. After all, the set of support strategy *restricts* the application of inference rules, and the hot list strategy *rearranges* the order in which conclusions are drawn. I conjecture with almost certainty, therefore, that many situations exist in which these two strategies together provide far more power than either does alone. Rather than seeking

one of these situations, I instead turn in the next subsection to a concrete example where the hot list strategy (in its dynamic incarnation) combines well with another useful strategy, namely, the *resonance strategy*. In the example, either strategy alone proved far less effective.

Before I leave the subject of lattice ordered groups, I focus on an intriguing example, a theorem I call *LOGT*2. The intrigue rests in part with its illustration of the complexities present in the coupling of the mechanisms of reasoning. Part of its intrigue also rests with the fact that Dahn informed me that he was unable to obtain a proof with any of the reasoning programs he was using. I thank Dahn for bringing this theorem to my attention, especially since my initial attempts to prove it with OTTER brought me in direct contact with a new set of complexities.

Dahn informed me that *LOGT*2 is a generalization of *LOGT*1. One is asked in *LOGT*2 to prove a relation among inverse, intersection, and union, a relation whose negation is the following, where *i* denotes inverse, *n* denotes intersection, and *u* denotes union.

i(n(a,b)) ! = u(i(a),i(b)).

As the problem was proposed to me, one is permitted to use essentially the entire underlying theory. Prompted by the knowledge of Dahn's lack of success with any reasoning program, my experience with experimentation immediately suggested discarding all nonunit clauses and all new (not in the input for *LOGT*1) equalities but two, the following.

u(x,n(y,z)) = n(u(x,y),u(x,z)).
n(x,u(y,z)) = u(n(x,y),n(x,z)).

I added in list(sos) the two positive equalities to the input for *LOGT*1 (of course, omitting the denial of its conclusion) and added the negative equality to list(passive).

Strongly motivated by the objective of obtaining a proof in reasonable CPU time, for my first attempt, I chose to rely on the resonance strategy, using as resonators the positive steps of a proof of *LOGT*1. (As noted in Section 2, a resonator is a formula or equation that, by treating its variables as indistinguishable from each other, provides patterns to direct a program's search; a resonator is *not* an added fact or lemma and does not take on any truth value.) Instantly, I was presented with a problem to solve. Of the various proofs I had found, which one should I choose? The objective of minimizing CPU time led me to focus on the proof (of *LOGT*1) obtained in the least amount of CPU time. I assigned max_weight the value 15 and the pick_given_ratio the value 6 because those values had been sufficient for proving *LOGT*1. I succeeded: In approximately 2753 CPU-seconds, OTTER produced a proof of length 30 and level 9, with retention of clause (17381). Although finding *any* proof was significant, I was not totally delighted with the CPU time. Naturally, I wondered whether the resonance strategy harmed more than helped OTTER, and I wondered what would be the result of using the hot list strategy. I therefore conducted various appropriate experiments—of which the results of the first few puzzled me greatly.

In each of the puzzling experiments, the run was terminated with ''Search stopped because sos empty''. How could that occur, for my main action was either to delete the use of the resonators or to include the use of the hot list strategy under various conditions? Indeed, at least on the surface, I felt that OTTER should still find some proof, even if a great deal of CPU time was required. After all, I had not changed the max_weight, and each of the resonators has weight (in symbol count) less than or equal to 15. Some thought and some consultation with colleagues led to the following conjectured explanation for exhausting all available clauses *without* finding a proof. The resonators being present markedly changes the order in which clauses are chosen as the focus of attention, which then changes the order in which demodulators are adjoined, which in turn may allow a clause with more than fifteen symbols to be appropriately demodulated to a clause with fifteen or fewer symbols, a clause that is crucial to finding a proof—directly or indirectly. The absence of the resonators may prevent the retention of a clause because of its weight through insufficient demodulation. To answer the justified question concerning my lack of certainty, I note that approximately 10,000 clauses are retained and more than 4,000,000 generated before the set of support goes empty without finding the desired proof.

As supporting evidence for my explanation, I cite two additional experiments whose goal was that of proving *LOGT*2. The two experiments also illustrate further the complexity of mathematics and logic and automated reasoning, and demonstrate the versatility of OTTER. In both experiments, I chose

as the search strategy one I use infrequently, namely, that of level saturation (breadth first). To instruct OTTER to make such a search, one includes the following command.

```
set(sos_queue).
```

Since the search is breadth first in the presence of this command, one does *not* use the ratio strategy and, therefore, removes or comments out the pick_given_ratio. The two experiments differed in that the second used the hot list strategy and the first did not; both used the resonance strategy, where the resonators were taken from an earlier proof, each assigned the value 2. In the first, OTTER produced no proof. In the second, OTTER produced a proof in approximately 1826 CPU-seconds with length 37 and level 15, with retention of clause (6698). For that experiment, I assigned the value 2 to the heat parameter and used the following hot list.

```
list(hot).
n(x,y) = n(y,x).
u(x,y) = u(y,x).
i(x)*x = 1.
x*i(x) = 1.
u(n(x,y),y) = y.
n(u(x,y),y) = y.
x*u(y,z) = u(x*y,x*z).
x*n(y,z) = n(x*y,x*z).
u(y,z)*x = u(y*x,z*x).
n(y,z)*x = n(y*x,z*x).
u(x,n(y,z)) = n(u(x,y),u(x,z)).
n(x,u(y,z)) = u(n(x,y),n(x,z)).
end_of_list.
```

With level saturation and the heat parameter assigned the value 2, when a new clause is retained at, say, level 4, the hot list strategy will first immediately generate clauses of level 5 (and heat level 1) and then, if any of them are retained, use them to immediatedly generate clauses of level 6 (and heat level 2). So, in the sense just described, the use of the hot list strategy with a breadth-first search enables a program to look ahead.

Thus, if one wishes to avoid raising the max_weight or making other changes to cope with termination because of sos empty, one might try level saturation alone or in combination with one or more other strategies. To further test the value of level saturation in the context of studying lattice ordered groups and to in general see what would happen, I conducted two additional experiments (run on a SPARCstation-2). In each, I removed the two positive equalities that were added to study *LOGT*2 and placed their respective negations in list(passive). The targets consisted of the negation of the conclusion of *LOGT*2 and the two just-cited negations, the former to prove *LOGT*2 *without* the use of the two additional equalities, and the latter to show that these two equalities are dependent axioms. In the first of the two experiments, I did not use the hot list strategy, and in the second I did use it. The first experiment proved *LOGT*2, in approximately 9601 CPU-seconds without the hot list strategy; in approximately 18 CPU-hours the second experiment failed to prove any of the three target theorems.

The proof obtained in the first experiment has length 32 and level 8 and completed with retention of clause (9685). As evidence that the hot list strategy helps little in this context, I conducted a third experiment, similar to the second except that the search strategy was based on weighting rather than on level saturation. In approximately 28,582 CPU-seconds, OTTER produced a proof of *LOGT*2 of length 39 and level 13, with retention of clause (33373).

Rather than simply turning to another topic, I mention here one additional set of experiments concerning *LOGT*2. The results of the experiments nicely illustrate how narrow can be the window of opportunity to answer a difficult question and how intertwined various procedures often are. Whereas one of the experiments yielded the shortest proof (given in the Appendix) of *LOGT*2 of which I know—a proof of length 22—the other experiments yielded *no* proof of any type. The 22-step proof was found by dropping the use of level saturation, assigning the value 10 rather than 6 to the pick_given_ratio, assigning the value 3 rather than 2 to the heat parameter, and using a hot list consistent with the recommendation (given in Section 2) concerning ‘‘short and simple’’ clauses that occur

in the input set of clauses. The hot list consisted of the following ten clauses.

1*x = x.
x*1 = x.
i(x)*x = 1.
x*i(x) = 1.
i(1) = 1.
i(i(x)) = x.
n(x,x) = x.
u(x,x) = x.
u(n(x,y),y) = y.
n(u(x,y),y) = y.

The other options and assignments were the same as were used in the cited successful level-saturation run for *LOGT*2. Although some of the other experiments from the set failed to yield a proof, they were each most valuable, for their respective failure provides evidence of how narrow is the window of opportunity. In one of the experiments that failed to yield any proof, except for dropping the use of level saturation, the experiment was identical to that which yielded the 37-step proof; in other words, the hot list consisted of the elements used in the level-saturation experiment that succeeded. In another experiment that failed, the hot list consisted of just the following two clauses.

i(x)*x = 1.
x*i(x) = 1.

In my view, the narrowness of the window of success is not a weakness of the hot list strategy; rather, it simply reflects the depth of mathematics and the fact that no panacea exists.

3.2. Combining Strategies

In the preceding subsection, I provided evidence of the value of using the hot list strategy to sharply reduce the CPU time required to complete an assignment, context (1) of Section 1.7. Here, my focus is more on the use of this strategy to bring within range theorems whose proof resisted various automated attempts, context (2) of Section 1.7. The area is two-valued sentential (or propositional) calculus. The inference rule is condensed detachment, captured (for this area of logic, as noted in Section 2) by the following clause together with hyperresolution.

-P(i(x,y)) | -P(x) | P(y).

Where the function *i* can be interpreted as ‘‘implication’’ and the function *n* as ‘‘negation’’, the theorems under consideration each have as axiom system that due to Lukasiewicz [Lukasiewicz63], the following (in clause notation), which are respectively called *L*1, *L*2, and *L*3.

P(i(i(x,y),i(i(y,z),i(x,z)))).
P(i(i(n(x),x),x)).
P(i(x,i(n(x),y))).

As in my study of other strategies, I returned to an attempt to prove the following 68 theorems (or, as Lukasiewicz called them, theses) originally brought to my attention by D. Scott. (I note that my colleague McCune points out correctly that the observation that the 68 theses are tightly coupled does not do justice to the situation. Rather, one might better view the ensemble of the theses as a set of proofs, in the sense that a later thesis is provable from some subset of earlier theses; therefore, viewing the theses in any way as separate theorems can indeed be misleading.) Theses 01, 02, and 03 are, respectively, *L*1, *L*2, and *L*3.

(thesis_04) i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))
(thesis_05) i(i(x,i(y,z)),i(i(u,y),i(x,i(u,z))))
(thesis_06) i(i(x,y),i(i(i(x,z),u),i(i(y,z),u)))
(thesis_07) i(i(x,i(i(y,z),u)),i(i(y,v),i(x,i(i(v,z),u))))
(thesis_08) i(i(x,y),i(i(z,x),i(i(y,u),i(z,u))))
(thesis_09) i(i(i(n(x),y),z),i(x,z))
(thesis_10) i(x,i(i(i(n(x),x),x),i(i(y,x),x)))

(thesis_11) i(i(x,i(i(n(y),y),y)),i(i(n(y),y),y))
(thesis_12) i(x,i(i(n(y),y),y))
(thesis_13) i(i(n(x),y),i(z,i(i(y,x),x)))
(thesis_14) i(i(i(x,i(i(y,z),z)),u),i(i(n(z),y),u))
(thesis_15) i(i(n(x),y),i(i(y,x),x))
(thesis_16) i(x,x)
(thesis_17) i(x,i(i(y,x),x))
(thesis_18) i(x,i(y,x))
(thesis_19) i(i(i(x,y),z),i(y,z))
(thesis_20) i(x,i(i(x,y),y))
(thesis_21) i(i(x,i(y,z)),i(y,i(x,z)))
(thesis_22) i(i(x,y),i(i(z,x),i(z,y)))
(thesis_23) i(i(i(x,i(y,z)),u),i(i(y,i(x,z)),u))
(thesis_24) i(i(i(x,y),x),x)
(thesis_25) i(i(i(x,y),z),i(i(x,u),i(i(u,y),z)))
(thesis_26) i(i(i(x,y),z),i(i(z,x),x))
(thesis_27) i(i(i(x,y),y),i(i(y,x),x))
(thesis_28) i(i(i(i(x,y),y),z),i(i(i(y,u),x),z))
(thesis_29) i(i(i(x,y),z),i(i(x,z),z))
(thesis_30) i(i(x,i(x,y)),i(x,y))
(thesis_31) i(i(x,y),i(i(i(x,z),u),i(i(y,u),u)))
(thesis_32) i(i(i(x,y),z),i(i(x,u),i(i(u,z),z)))
(thesis_33) i(i(x,y),i(i(y,i(z,i(x,u))),i(z,i(x,u))))
(thesis_34) i(i(x,i(y,i(z,u))),i(i(z,x),i(y,i(z,u))))
(thesis_35) i(i(x,i(y,z)),i(i(x,y),i(x,z)))
(thesis_36) i(n(x),i(x,y))
(thesis_37) i(i(i(x,y),z),i(n(x),z))
(thesis_38) i(i(x,n(x)),n(x))
(thesis_39) i(n(n(x)),x)
(thesis_40) i(x,n(n(x)))
(thesis_41) i(i(x,y),i(n(n(x)),y))
(thesis_42) i(i(i(n(n(x)),y),z),i(i(x,y),z))
(thesis_43) i(i(x,y),i(i(y,n(x)),n(x)))
(thesis_44) i(i(x,i(y,n(z))),i(i(z,y),i(x,n(z))))
(thesis_45) i(i(x,i(y,z)),i(i(n(z),y),i(x,z)))
(thesis_46) i(i(x,y),i(n(y),n(x)))
(thesis_47) i(i(x,n(y)),i(y,n(x)))
(thesis_48) i(i(n(x),y),i(n(y),x))
(thesis_49) i(i(n(x),n(y)),i(y,x))
(thesis_50) i(i(i(n(x),y),z),i(i(n(y),x),z))
(thesis_51) i(i(x,i(y,z)),i(x,i(n(z),n(y))))
(thesis_52) i(i(x,i(y,n(z))),i(x,i(z,n(y))))
(thesis_53) i(i(n(x),y),i(i(x,y),y))
(thesis_54) i(i(x,y),i(i(n(x),y),y))
(thesis_55) i(i(x,y),i(i(x,n(y)),n(x)))
(thesis_56) i(i(i(i(x,y),y),z),i(i(n(x),y),z))
(thesis_57) i(i(n(x),y),i(i(x,z),i(i(z,y),y)))
(thesis_58) i(i(i(i(x,y),i(i(y,z),z)),u),i(i(n(x),z),u))
(thesis_59) i(i(n(x),y),i(i(z,y),i(i(x,z),y)))
(thesis_60) i(i(x,i(n(y),z)),i(x,i(i(u,z),i(i(y,u),z))))
(thesis_61) i(i(x,y),i(i(z,y),i(i(n(x),z),y)))
(thesis_62) i(i(n(n(x)),y),i(x,y))
(thesis_63) i(x,i(y,y))
(thesis_64) i(n(i(x,x)),y)
(thesis_65) i(i(n(x),n(i(y,y))),x)

(thesis_66) i(n(i(x,y)),x)
(thesis_67) i(n(i(x,y)),n(y))
(thesis_68) i(n(i(x,n(y))),y)
(thesis_69) i(x,i(n(y),n(i(x,y))))
(thesis_70) i(x,i(y,n(i(x,n(y)))))
(thesis_71) n(i(i(x,x),n(i(y,y))))

I conducted experiments *with* and *without* the hot list strategy to see which and how many of the 68 theses OTTER would prove, starting with the three axioms of Lukasiewicz. Of my experiments, for reasons of economy, I cite but one. Among the options, I assigned max_weight the value 20, the pick_given_ratio the value 5, max_mem the value 60 megabytes, and max_distinct_vars the value 5. Because of the last cited assignment, OTTER discarded any deduced formula that contains six or more distinct variables. The program also discarded any deduced formula that contains 21 or more symbols (counting the predicate symbol), because of the assignment to max_weight and because no weight templates were included. The assignment to max_weight and to max_distinct_vars was motivated, respectively, by the fact that none of the 68 theses has weight strictly greater than 20 and by the fact that none of them contains strictly more than five distinct variables. Clearly, in a more general context, such characteristics do *not* guarantee that the desired proofs are obtainable within corresponding bounds; however, in this case, prior experimentation shows that proofs do exist within the specified bounds. (Shortly, when I discuss the use of the dynamic hot list strategy, one will see how significant the inclusion of weight templates can be. This variant of the hot list strategy was formulated by McCune as an extension of the hot list strategy, perhaps motivated in part by my interest in self-analytical programs.) The members of the hot list consisted of the nonunit clause for condensed detachment and the clauses corresponding to the three Lukasiewicz axioms.

The experiment under discussion consisted of two parts, one in which the hot list strategy was *not* used, and one in which it *was* used with the heat parameter assigned the value 3. When the hot list was not used, the program deduced proofs of the following 41 theses (not counting proofs of the same thesis):

th_04 th_05 th_06 th_07 th_08 th_09 th_10 th_11 th_12 th_13 th_15
th_16 th_17 th_18 th_19 th_20 th_21 th_22 th_23 th_24 th_25 th_30
th_36 th_37 th_38 th_39 th_40 th_41 th_42 th_45 th_48 th_49 th_50
th_62 th_63 th_64 th_65 th_66 th_67 th_68 th_71

When the hot list strategy was used, the following 44 were proved (including all but two, thesis 48 and thesis 50, of the 41 proved without the use of the strategy).

th_04 th_05 th_06 th_07 th_08 th_09 th_10 th_11 th_12 th_13 th_14
th_15 th_16 th_17 th_18 th_19 th_20 th_21 th_22 th_23 th_24 th_25
th_26 th_27 th_28 th_30 th_36 th_37 th_38 th_39 th_40 th_41 th_42
th_43 th_45 th_49 th_62 th_63 th_64 th_65 th_66 th_67 th_68 th_71

The following five were the ones proved in the second half of the experiment but not in the first half.

th_14 th_26 th_27 th_28 th_43

Of the 39 theses proved (in common) without the hot list strategy, that approach was more effective (in CPU time) twenty-eight times when compared with using the strategy. On a SPARCstation-10, in approximately 124 CPU-seconds, 41 theses were proved without the hot list strategy; in approximately 410 CPU-seconds, 44 theses were proved with its use. Each of the two halves of the experiment was terminated by the max_mem parameter.

In the experiment under discussion, I was also after bigger game (than seeing how many of the 68 theses would be proved). Specifically, to show that the three given Lukasiewicz formulas axiomatize two-valued sentential calculus, one can deduce all of the members of some (other) known axiom system. With and without the use of the hot list strategy, none of the five known axiom systems was deduced, Church's, Frege's, Hilbert's, Lukasiewicz's (an alternate), and Wos's.

At this point in my experimentation, in part prompted by the objective of deducing each of theses 4 through 71, I had an inspiration: Perhaps I could reach the objective by using the dynamic hot list

strategy and otherwise making no other changes from the experiments cited earlier in this subsection—except, as will be seen, the possible use of yet one more strategy, the *resonance strategy*. For those who are curious about how research sometimes proceeds, I note that I asked myself how I might ''choose'' the clauses to be adjoined to the hot list *during* the run. An appealing set of such clauses consists of the members of the known axiom systems (other than that consisting of *L*1, *L*2, and *L*3), namely, theses 18, 19, 21, 22, 30, 35, 37, 39, 40, 46, 49, 54, 59, and 60, each being a member of a known axiom system for two-valued sentential calculus. Intuitively, their appeal rests with the fact that they offer enough power, when taken in the appropriate combinations, to axiomatize this area of logic. Clearly, in the spirit of mathematics and logic, I could *not* include *any* of the cited fourteen theses until deduced from the Lukasiewicz axiom system. But, with the use of the resonance strategy, I saw how to include such a clause *if* it was deduced. To see how that can be done, I briefly review the nature of the resonance strategy.

The resonance strategy is a strategy for directing a program's reasoning. The means for doing so rests with the inclusion of *resonators*, formulas or equations that the researcher conjectures to have a symbol pattern (ignoring specific variables by treating them as indistinguishable from each other) that should be emphasized. To each included resonator, the researcher is asked to assign a value to reflect its conjectured significance, the smaller the value, the greater the significance. With OTTER, one places chosen resonators in the form of weight templates in the appropriate weight_list, specifically, in almost all cases, weight_list(pick_and_purge) or weight_list(pick_given). (As McCune observes, if one includes as resonators the 68 theses, each with low weight, the actions of OTTER are somewhat reminiscent of proof checking a set of theorems.)

To cause OTTER during the run to adjoin to the hot list a formula that corresponds to one of the cited fourteen members of known axiom systems, I assigned the (input) heat parameter the value 2 and included the following weight templates in weight_list(pick_and_purge). (A clause technically has two weights, its pick_given weight which is used in the context of choosing clauses as the focus of attention to drive the program's reasoning, and its purge_gen weight which is used in the context of clause discarding; often the two weights are the same.)

```
weight(P(i(x,i(y,x))),2).
weight(P(i(i(i(x,y),z),i(y,z))),2).
weight(P(i(i(x,i(y,z)),i(y,i(x,z)))),2).
weight(P(i(i(x,y),i(i(z,x),i(z,y)))),2).
weight(P(i(i(x,i(x,y)),i(x,y))),2).
weight(P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))),2).
weight(P(i(i(i(x,y),z),i(n(x),z))),2).
weight(P(i(n(n(x)),x)),2).
weight(P(i(x,n(n(x)))),2).
weight(P(i(i(x,y),i(n(y),n(x)))),2).
weight(P(i(i(n(x),n(y)),i(y,x))),2).
weight(P(i(i(x,y),i(i(n(x),y),y))),2).
weight(P(i(i(n(x),y),i(i(z,y),i(i(x,z),y)))),2).
weight(P(i(i(x,i(n(y),z)),i(x,i(i(u,z),i(i(y,u),z))))),2).
```

(The given fourteen templates correspond in order to theses 18, 19, 21, 22, 30, 35, 37, 39, 40, 46, 49, 54, 59, and 60.) Because of my actions, when and if the program deduces one of the fourteen members of the known axiom systems, the corresponding clause will be adjoined to the hot list; [Wos96c] focuses heavily on the combination of the resonance strategy and the dynamic hot list strategy. However, because weighting treats variables as indistinguishable from each other, I was aware that clauses similar to, but not identical to, one of the fourteen target clauses might also be adjoined to the hot list during the run.

I was ready to study the effects of using the described combination of strategies, namely, the coupling of the dynamic hot list strategy with the resonance strategy. Four experiments were in order. In the first, I would omit the use of the dynamic hot list strategy, but I would use the fourteen resonators. In the second through the fourth, I would, respectively, assign the heat parameter the value 3, 2, and 1 and, in each of the three experiments, use the fourteen resonators and the dynamic hot list strategy with the assignment of the value 2 to each of the resonators and to the dynamic_heat_weight parameter. I

chose not to study the use of the dynamic hot list strategy in isolation, for, other than the discussed approach for suggesting which clauses to adjoin during the run, I had no other appealing idea at the time.

When the dynamic hot list strategy was absent and the resonance strategy was used (with the cited fourteen resonators), OTTER deduced the Hilbert axiom system (consisting of theses 3, 18, 21, 22, 30, and 54, of which thesis 30 is dependent). On a SPARCstation-10, the proof was completed in approximately 184 CPU-seconds with retention of clause (40303). The result shows that, in fact, the three cited Lukasiewicz formulas do axiomatize two-valued sentential calculus. By comparing the success to the results of the preceding subsection, one also sees that the resonance strategy is indeed promising. The length of the proof is 46, and the level is 23. Of the clauses that were chosen as the focus of attention, twenty-five have weight 2 (because of the inclusion of the resonators, each of which has the assigned value of 2). I expected evidence of this phenomenon, which is why I commented on the fact that weighting does not distinguish between particular variables. Because none of theses 35, 46, 59, and 60 was deduced, the other known axiom systems were out of reach, namely, the Church system (consisting of theses 18, 35, and 49), the Frege system (consisting of theses 18, 21, 35, 39, 40, and 46, of which thesis 21 is dependent on the other five), the alternate Lukasiewicz system (consisting of theses 19, 37, and 59), and the Wos system (consisting of theses 19, 37, and 60).

Contrary to one's preference, if one is after evidence of the value of the dynamic hot list strategy when coupled with the resonance strategy, OTTER was less successful in the second of the four experiments. Indeed, for none of the five target axiom systems did the program deduce all of its members. In fact, excluding duplicates, only the following 38 theses were proved.

th_04 th_05 th_06 th_07 th_08 th_09 th_10 th_11 th_12 th_13 th_14
th_15 th_16 th_17 th_18 th_19 th_20 th_24 th_30 th_36 th_37 th_38
th_39 th_40 th_41 th_42 th_48 th_50 th_53 th_56 th_62 th_63 th_64
th_65 th_66 th_67 th_68 th_71

The third experiment (in which the assignment to the input heat parameter was changed from 3 to 2) proved more satisfying. In addition to proving 52 (the following) of the 68 theses, OTTER completed a proof of the Hilbert axiom system in approximately 1344 CPU-seconds on a SPARCstation-10.

th_04 th_05 th_06 th_07 th_08 th_09 th_10 th_11 th_12 th_13 th_14
th_15 th_16 th_17 th_18 th_19 th_20 th_21 th_22 th_23 th_24 th_25
th_29 th_30 th_36 th_37 th_38 th_39 th_40 th_41 th_42 th_45 th_46
th_47 th_48 th_49 th_50 th_51 th_52 th_53 th_54 th_56 th_62 th_63
th_64 th_65 th_66 th_67 th_68 th_69 th_70 th_71

The proof of the Hilbert axiom system completed with retention of clause (136916), and it has length 80 and level 38; the announced length is 87 because of the presence of duplicate deduced steps resulting from the use of the dynamic hot list strategy. Of the 87 deduced steps, 56 are present because of the use of the dynamic hot list strategy, 32 showing heat=1, and 24 showing heat=2. (The value of *n* in heat=n, in the history of a deduced clause, shows how much recursion occurred regarding the use of the hot list.) Of course, its length is unimpressive, to say the least, for the Lukasiewicz proof has length 34, and the proof of the Hilbert axiom system produced in the first of the four experiments has length 46 (and level 23) and was obtained in approximately 184 CPU-seconds. On the other side, the deduction of thesis 46 in the third experiment is a plus, for that thesis is a member of the Frege axiom system, and it was not deduced in the first experiment or in the second.

Regarding the fourth experiment, its results differed little from those of the third experiment. Of the 68 target theses, again 52 were deduced. Rather than thesis 29 (proved in the third experiment), thesis 43 was deduced, which is of little interest in this study. Again, the Hilbert axiom system was proved, in approximately 1039 CPU-seconds, with retention of clause (126949). The proof has length 50 and level 26; the announced length is 54 because of containing duplicate deduced steps resulting from the use of the dynamic hot list strategy. Of the 54 deduced steps, 23 result from the use of the dynamic hot list strategy as shown by heat=1 in the history. Other than the Hilbert axiom system, each of the second, third, and fourth experiments failed to deduce at least one member of the other target axiom systems. As for the number of clauses chosen as the focus of attention that match a resonator, in

contrast to the 25 in the first of the four experiments, 19 were chosen in the second, 48 in the third, and 39 in the fourth. It seems no coincidence that the lowest of the four numbers occurred in the second experiment, the only experiment that failed to complete a proof of any of the five target axiom systems.

Especially for researchers intent on using OTTER to aid in the discovery of new results, a subtle point merits discussion here, a point that focuses on completing a proof of an *entire* axiom system in contrast to completing proofs of *all* of the individual members of that axiom system. For example, the third experiment proved all individual members of the Hilbert axiom system in approximately 1099 CPU-seconds when the proof of thesis 54 was completed. The proof of thesis 54 was completed by finding a unit conflict involving clause (111208), a deduced clause that corresponds to thesis 54. Nevertheless, the proof of the entire axiom system required approximately 1344 CPU-seconds, completing only after the retention of clause (136916), the empty clause. The explanation for the delay (in CPU time) rests with the need for OTTER to choose as the focus of attention all clauses that are needed to be considered by hyperresolution, where the nucleus (in this case) corresponds to the negation of the **and** of the members of the Hilbert axiom system. When the assumed falseness of the theorem to be proved results in the inclusion of a nonunit clause to be used as a nucleus, such delays are frequently encountered for the given reason. In the example being used for illustration, clause (111208) is the 518th clause chosen to drive the reasoning, in contrast to the 500th clause which is used to deduce clause (111208). Before choosing clause (111208) as the focus of attention, the program first chose other earlier-retained clauses with a weight of 2 (which is the weight of the clause in question) and, because of the ratio strategy, also first chose some clauses with weight greater than 2. After all, the program had no way of knowing that the choice of clause (111208) would complete the proof of one of the target axiom systems. If, contrary to the experiment, no resonator had been included that corresponded to thesis 54, then almost certainly even more CPU time would have been required before clause (111208) was chosen.

Of course, in an obvious sense, when all members of an axiom system have been deduced, then the desired proof (of the entire axiom system) has been completed. But, were one then to take the ensemble of proofs, the result would, in the main, be quite unsatisfying. Indeed, duplicate steps would abound, typically. The avoidance of such duplicate steps is one of the motivations for including a nonunit clause that corresponds to the denial of the **and** of a set of targets, in this case representing a target axiom system. Hardly acceptable in mathematics or in logic is a proof with duplicate steps, a proof produced by simply taking the ensemble of a set of proofs. (Note that, under certain conditions—for example, with the use of ancestor subsumption coupled with back subsumption, and with the use of the dynamic hot list strategy—OTTER can return a proof containing duplicate deduced steps. With the use of ancestor subsumption, when a conclusion is drawn more than once, the lengths of the corresponding derivation path lengths are compared; the program retains the copy reached by the shorter path.)

For the curious, when I modified the fourth experiment by replacing the dynamic hot list strategy with the hot list strategy, OTTER completed a proof of the Hilbert axiom system in far less CPU time, namely, in approximately 273 CPU-seconds on a SPARCstation-10. The proof has length 46, level 26, and was completed with retention of clause (55374). Of significance, the modification prevents the program from adjoining *during* the run new clauses to the hot list. Regarding possible losses with the replacement, the clause correspondent of thesis 46 was not deduced; that thesis is important because of being a member of the Frege axiom system. As for the other four target axiom systems, in each case, at least one of their members was not deduced.

I conducted three families of additional experiments (experiments 5a, 5b, and 5c, experiments 6a, 6b, and 6c, and experiments 7a, 7b and 7c), prompted by the intention of demonstrating the power offered by the fourteen resonators corresponding to members of known axiom systems for two-valued sentential calculus. In each family of experiments, to have as little contact with known axiom systems as possible, I deliberately chose a set of resonators containing no members of such an axiom system. All other options were unchanged from the last three of the four experiments just discussed: For example, experiments 5a through 7a each relied on an assignment of the value 3 to the heat parameter, 5b through 7b an assignment of the value 2, and 5c through 7c an assignment of the value 1. The dynamic hot list strategy in combination with the resonance strategy was the focus, with the intent of adjoining to the hot list *during* the run any clause that matched a resonator.

As noted, each of the three families consists of three experiments, where the difference between two families rests with the choice of resonators. In an attempt to remove any bias on my part—in the first of the three families of experiments, Experiments 5a, 5b, and 5c—I chose the first available set of fourteen theses that excludes a member of a known axiom system, namely, theses 04 through 17. How convenient that there were just enough theses from which to choose before encountering the first member of an axiom system, thesis 18! For those researchers wishing some insight into my approach to research, I made a guess (with little foundation) about the outcome; I often make a guess or a conjecture, even if others view the guess as a wild one. In this case, I thought that the use of theses 04 through 17 as resonators would not produce a great deal in the context of deducing an entire axiom system.

A cursory glance at Experiment 5a suggested I had guessed right, for none of the other known axiom systems was proved in its entirety. Indeed, the only occurrence of the empty clause was that regarding the Lukasiewicz axiom system consisting of $L1$, $L2$, and $L3$. Closer inspection, however, revealed that, except for thesis 60, each of the fourteen desired theses had in fact been deduced. Then, other than the Wos axiom system which depends on thesis 60, why did OTTER fail to deduce any of the corresponding empty clauses, for Church, Hilbert, (the alternate of) Lukasiewicz, or Frege? The following observations, which extend some of my earlier remarks regarding the proof of an *entire* axiom system versus proving all of the individual members, capture what is neeeded to answer this natural question.

None of the included weight templates corresponds to one of the fourteen members of the other known axiom systems. Therefore, each of the corresponding fourteen clauses when (and if) retained is given a weight equal to its symbol count. Even if a clause has a small weight (in terms of its symbol count), that clause may not be chosen as the focus of attention, for, among other reasons, many clauses with smaller weight may be retained. Because of the fourteen resonators, theses 04 through 17, 209 of the clauses chosen as the focus of attention have weight equal to 2. From each of the five target axiom systems, at least one clause correspondent was *not* chosen as the focus of attention or, in the case of thesis 60, was not even deduced. Where weight is measured in symbol count, from Church, thesis 35 with a weight of 14 was not chosen; from Frege, thesis 46 with a weight of 10 was not chosen; from Hilbert, thesis 30 with a weight of 10 was not chosen; from (the alternate of) Lukasiewicz, thesis 59 with a weight of 15 was not chosen; from Wos, thesis 60 with a weight of 19 was not deduced.

Nevertheless, except for thesis 60, OTTER did deduce all of the members of each of the five known axiom systems and, therefore, (in effect) completed a proof of all but the Wos axiom system. I thus have come to that point where the earlier citing of the alternate figures for proof completion is relevant, the figure concerning the proof of the last member of an axiom system in contrast to the figure for proving the entire axiom system. Their significance and usefulness become apparent, namely, for the purpose of comparison. Specifically, in Experiment 5a, the last member deduced of the Hilbert axiom system (thesis 54) was deduced in approximately 1224 CPU-seconds (using a SPARCstation-10) with the retention of clause (62727), with length 44 and level 22; OTTER reported a length of 51, due to the duplicate clauses that can be present when the dynamic hot list strategy is used. The last member deduced of the (alternate) Lukasiewicz axiom system (thesis 59) was deduced in approximately 1550 CPU-seconds with the retention of clause (76107), with length 61 and level 24; OTTER reported a length of 70. The last member of the Church axiom system (thesis 35) was deduced in approximately 1837 CPU-seconds with the retention of clause (84229), with length 49 and level 24; OTTER reported a length of 57. The last member of the Frege axiom system (thesis 46) was deduced in approximately 1992 CPU-seconds with the retention of clause (88108), with length 45 and level 24; OTTER reported a length of 53. In other words, although the empty clause was *not* generated in *any* of the five cases focusing on the respective five known axiom systems, proofs of four were completed. Although the CPU times and the proof lengths in Experiment 5a are hardly impressive, still I find it interesting that the arbitrary selection of the first fourteen theses as resonators did so well, especially when compared with its closest counterpart, namely, the second of the four experiments cited earlier. Perhaps the nature of this experiment correctly suggests that a powerful approach to proving theorems is that of using a combination of the dynamic hot list strategy and the resonance strategy.

The explanation for not citing proof lengths for the entire axiom systems, given the lengths of individual proofs, rests with the high probability that the proofs of the individual members share steps

in common. For but one example, the 34-step proof of thesis 30 (found in Experiment 5a) and the 44-step proof of thesis 54 share 28 steps in common; both theses are members of the Hilbert axiom system. One of the advantages of deducing the empty clause (by keying on the disjunction of the negations of the members of each axiom system) is that the resulting proofs, in almost all cases but the somewhat bizarre, contain no duplicate steps. When I offered this ''trick'' of keying on the respective disjunctions to D. Scott for solving the problem of producing a proof with no duplicate steps, he seemed pleased. With this trick, one (in almost all cases) obtains the desired proof length by merely consulting the appropriate output file for the corresponding figure.

Immediately, one might wonder whether the cited use of the resonance strategy—and, for that matter, other such uses—causes the program OTTER simply to proof check rather than proof find. My view is that in this case, and in most others in my experimentation, some, but not much, proof checking is occurring. For a hint of why I believe this to be so, one merely looks at the proofs that are obtained, say, of thesis 35. I choose this thesis, for experimentation over many months shows that thesis 35 is difficult to deduce if one does not employ strategy. (A glance at this subsection illustrates the difficulty.) Of the 49 unduplicated deduced steps of its proof, although 12 are from among the 14 resonators, there still remain 37 that must be found that are outside a proof-checking mode. If the object had been more in the spirit of proof checking, typically I would have included all of the steps of some proof, each with a weight of 2, and assigned max_weight the value 2. I have rather often made such cursory proof-checking runs; rigorous proof checking, for me, requires strict adherence to the precise history, which can be done through the use of demodulation. (For sample input files for cursory proof checking and for rigorous proof checking, see, respectively, Appendix B and Appendix C of [Wos96a].)

Perhaps a tougher test of whether OTTER is proof checking or proof finding rests with the percentage of deduced proof steps (of a completed proof) that match one of the included resonators. A more intense glance at the proof of thesis 35 produces more persuasive evidence that proof checking is *not* occurring. Specifically, 28 of its 49 steps do *not* match *any* of the resonators, where all variables are treated as indistinguishable from each other. In my view, among the significant properties possessed by resonators, their use enables a program to escape cul-de-sacs and to cross wide plateaus of sets of conclusions.

Next in order was Experiment 5b. The only change that was made was to replace the assignment of the value 3 to the heat parameter with an assignment of the value 2. Other than thesis 46, all members of the five target axiom systems were deduced. However, as in Experiment 5a, no axiom system was proved in its entirety (in the sense just discussed). Among the interesting results, where thesis 35 was proved in approximately 1837 CPU-seconds in Experiment 5a, in Experiment 5b its proof was completed in approximately 196 CPU-seconds. In contrast to the 49-step proof obtained with Experiment 5a, Experiment 5b produced a 55-step proof. Of the 55 deduced steps, 23 are not present among the 49-step proof. Regarding the CPU time to deduce the individual members, of the 13 deduced by both experiments, all but thesis 18 were deduced in less time with Experiment 5b.

As for the third element of the first family, Experiment 5c in which the heat parameter was assigned the value 1, a small but significant success occurred: The Hilbert axiom system was proved. The proof has length 42 and level 28, and it was completed in approximately 1335 CPU-seconds with retention of clause (118987). The announced proof length is 47 (because of duplicate steps resulting from the use of the dynamic hot list strategy), and 26 of the 47 steps are present through the use of the hot list. A glance at the output file again treats one to the vagaries of proof finding, for, of the fourteen members of the target axiom systems, theses 35 and 59 are the ones that were not deduced. Regarding interesting comparisons, thesis 30 was deduced in approximately 1180 CPU-seconds in Experiment 5a, in approximately 133 CPU-seconds in Experiment 5b, and in approximately 9 CPU-seconds in Experiment 5c. Thesis 54 was deduced in approximately 1224 CPU-seconds in Experiment 5a, in approximately 294 CPU-seconds in Experiment 5b, and in approximately 959 CPU-seconds in Experiment 5c.

Immediately, one might wish some insight into the disparate behavior in the three experiments caused by the single change, that in the assignment to the heat parameter. Especially researchers not familiar with OTTER or, for that matter, with automated reasoning might be justly puzzled by the data just presented for Experiments 5a through 5c. The key rests with the perturbing of the search space explored by the program. Just the change in the assignment from the value 3 to the value 2 to the heat parameter causes less recursion through the hot list, in turn causing a potentially sharp change in the

order in which conclusions are drawn. When conclusions are drawn in a sharply different order, then, clearly, they are typically retained in a sharply different order, generally resulting in their being examined in a sharply different order.

Conceivably, one might argue that such apparently small option changes should not cause such possibly large changes in the results, and thus a flaw is encountered. One might go further and say that the introduction of strategy should not have such dramatic effects. Predictably, I take the opposite stand, asserting that the purpose of strategy is in fact to heavily perturb the search space of conclusions. Through such perturbations—due to the use of strategy—new and interesting proofs are found, more elegant proofs are completed, and, occasionally, an open question is answered.

Regarding the second family of three experiments (Experiments 6a through 6c), because I noted that, at least in some respects, the first family of experiments (5a through 5c) seemed to be progressing slowly (which I attributed to the presence of a thesis with five distinct variables), I decided to replace that thesis (thesis 07) with thesis 20 (also not a member of a known axiom system). As in Experiment 5a, of the five target axiom systems, none were proved in Experiment 6a in the sense of deducing the empty clause. However, similar to Experiment 5a, all members of the five systems, except theses 46 and 60, were deduced. Therefore, except for the Frege axiom system and the Wos axiom system, the other three were proved in the obvious technical sense. Among the highlights, proofs for the axiom systems were completed in the following order: Hilbert in approximately 645 CPU-seconds, (the alternate of) Lukasiewicz in approximately 650 CPU-seconds, and Church in approximately 728 CPU-seconds. Respectively, the last member proved was thesis 54, thesis 59, and thesis 35. Their respective proof lengths and levels are 46 and 25; 45 and 25; and 57 and 27.

Experiment 6b also proved none of the target axiom systems in its entirety (by deducing the empty clause). However, of the fourteen members of the five systems, all but thesis 60 was deduced. If viewed from proving all members, the axiom systems were proved in the following order: Hilbert, (the alternate of) Lukasiewicz, Frege, and Church. Respectively, the last member proved in order was thesis 54 (in approximately 250 CPU-seconds), thesis 59 (in approximately 400 CPU-seconds), thesis 46 (in approximately 402 CPU-seconds), and thesis 35 (in approximately 406 CPU-seconds). For thesis 54, the proof length is 66 and level is 30; for thesis 59, the length is 68 and level is 31; for thesis 46, the length is 65 and level is 31; and for thesis 35, the length is 79 and level is 34.

The last of the second family of experiments is Experiment 6c. Puzzling indeed, the experiment deduced all of the members of the Hilbert axiom system, but failed to deduce theses 35, 59, and 60. Therefore, only one target axiom system was proved, that of Hilbert. The last member of the system that was deduced was thesis 54, in approximately 820 CPU-seconds, with a length of 40 and a level of 25. The entire proof of the Hilbert axiom system was found in approximately 991 CPU-seconds. The proof has length 43 and level 25, completing with retention of clause (107685). The experiment provides some evidence for the value of using some recursion through the hot list, for example, by assigning the heat parameter the value 2 (as in the preceding experiment) rather than the value 1.

For the third family of experiments (Experiments 7a through 7c), to see how important is the choice of the set of fourteen theses to be used as resonators—requiring that none of the chosen theses be a member of a known axiom system—I decided to use as resonators theses 56 through 71, excluding theses 59 and 60 (which are members of known axiom systems). Experiment 7a was indeed unsuccessful. In particular, of the fourteen members of interest, only theses 18, 19, 39, 40, 49, and 54 were deduced. When the heat parameter was changed from 3 to 2 in Experiment 7b, little progress occurred, namely, theses 21 and 22 were also deduced. As for the last member of the family of three experiments, Experiment 7c in which the heat parameter was assigned the value 1, it fared no better than Experiment 7b.

Among the tentative conclusions one might draw from the results of Experiments 2 through 4 and the three families of experiments just discussed, evidence suggests that early steps in proofs serve more effectively than do later steps. For example, the early theses proved to produce more than did the later theses 56 through 71 (excluding theses 59 and 60). Also, as supported by the experiments, resonators that correspond to powerful lemmas (such as members of known axiom systems) can offer at least reasonable effectiveness. However, being a powerful lemma can be correlated with difficulty of proof and may, therefore, delay or even prevent an automated reasoning program from deducing the corresponding

clause, thus delaying or preventing its use (as a resonator) to direct the reasoning. In contrast, early steps of known proofs are generally easier to deduce and hence more accessible to the program to direct its search, though perhaps they lack power.

Various possible additional conclusions are suggested, all or most of all of which must await further experimentation to decide which are valid.

1. If the goal is to prove that the three Lukasiewicz formulas axiomatize two-valued sentential calculus by deducing *any* of the other five known axiom systems, then the selection (from theses 04 through 71) of the resonators can be crucial. After all, selecting the last available fourteen that exclude a member of an axiom system leads to the deduction of none of the five target axiom systems, whether the goal is the empty clause or the goal is merely the deduction of all appropriate theses (members of the system). One might have expected that far more power is offered by choosing for the fourteen resonators the members of the known axiom systems or choosing the initial set of theses, the former because of their significance (in the sense that powerful lemmas are significant) and the latter because they represent the beginning of the Lukasiewicz study. The results of Experiments 3 and 4 support the value of using as resonators members of the five target axiom systems, and the results of Experiment 5c and 6c support the value of using proof steps occurring early in a proof. Regarding the choice of the class of resonators, the following provides some insight. Resonators that correspond to early steps of some known proof are in part appealing because of the usual ease of deducing a clause that matches one of them. A resonator that corresponds to a known lemma, though having less appeal in the context of ease of deducing a matching clause, tends to offer more power. When Experiment 6c was modified by removing the use of all resonators, among the theses that were *not* deduced are 35, 46, 54, 59, and 60, each of which is a member of a known axiom system for two-valued sentential calculus.

2. The use of the later theses as resonators is not very reminiscent of proof checking, for that set forms the final segment of the Lukasiewicz study. Further, (in my view) when the resonators consist of a small fraction of the steps of a proof completed by OTTER, the notion that proof checking is essentially what is occurring seems in error.

3. The combination of the dynamic hot list strategy and the resonance strategy, in which the choice of the newly adjoined members of the hot list is guided by the choice of resonators that are included in the input, merits substantial study; see [Wos96c]. I offer additional evidence in that regard in the next subsection, where I briefly study Robbins algebra.

3.3. Robbins Algebra

In this subsection—counting my earlier published papers—I revisit Robbins algebra for the *k*th time, where *k* is beginning to be a large integer. My fascination with this area of mathematics mainly rests with three factors. First, just three axioms suffice to study this algebra, the following expressed in yet one more notation acceptable to OTTER, where one can interpret the function *n* as complement and the function + as union.

```
EQ(+(x,y),+(y,x)).    %  commutativity
EQ(+(+(x,y),z),+(x,+(y,z))).    %  associativity
EQ(n(+(n(+(x,y)),n(+(x,n(y))))),x).    %  Robbins axiom
```

Second, at least on the surface, Robbins algebra is a natural target for automated reasoning; indeed, one can easily study this field by using paramodulation and choosing various options to control this inference rule or by using paramodulation within a Knuth-Bendix approach. Third, and so intriguing, the question of whether every Robbins algebra is a Boolean algebra was open until McCune with his program EQP answered it in the affirmative [McCune97]; in fact, Tarski and his students failed to answer the question. The question is posed in [Henkin71]. One can rather quickly prove that every *finite* Robbins algebra is Boolean. Far more tantalizing, as one sees in this subsection and in earlier papers (for example, by my colleague S. Winker), the adjunction to the cited three axioms of any one of a number of properties that hold in Boolean algebra suffices to yield a Boolean algebra. Here, I focus on some of

the corresponding theorems, many of which were proved in part at my suggestion to Winker [Winker90,Winker92].

I begin with some easy-to-prove theorems. As noted earlier, the use of an automated reasoning program would be even more attractive than it is if ''easy'' theorems would almost always be proved quickly. Noting that no panacea exists, I provide some evidence that suggests that the quickness is increased when the hot list strategy is used.

The first theorem I studied (*RAT*1, Robbins Algebra Theorem 1) asserts that the adjunction (to the three axioms for a Robbins algebra) of the familiar property ''*x* equals the complement of the complement of *x*'' yields Boolean. My plan was to try this theorem and others to be discussed *without* and then *with* the hot list strategy. If all went well, I would accrue yet more evidence of the value of using this strategy, even for simple theorems. Of the myriad of choices offered by OTTER, I chose the following options.

```
set(knuth_bendix).
clear(eq_units_both_ways).
set(index_for_back_demod).
set(dynamic_demod_lex_dep).
set(process_input).
assign(max_weight,20).
assign(max_mem,24000).
% assign(heat,1).
lex([A, B, c, d, e, f, g(x), n(x), +(x,x)]).
```

As it turns out, OTTER easily produces a proof by contradiction from the Robbins axioms together with $n(n(x)) = x$ and the denial of Huntington's axiom, the following.

```
-EQ(+(n(+(a,n(B))),n(+(n(a),n(B)))),B).  % denial of Huntington's axiom
```

Huntington's axiom together with commutativity and associativity of union suffice to axiomatize Boolean algebra. In approximately .2 CPU-seconds, OTTER produced a proof of length 4 and level 4, with the retention of clause (71). Interesting to me, associativity of union was not used.

Because I was studying the potential of the hot list strategy—even though I could hardly expect any significant improvement in efficiency—I nevertheless repeated the experiment, but this time using a hot list. In part to test the usefulness of my recommendation concerning what to place in the (input) hot list, I chose as its only member the special hypothesis, the clause equivalent of $n(n(x)) = x$. For the heat parameter, I assigned the value 1. In approximately .1 CPU-seconds, OTTER produced a proof of length 5 and level 5, with the retention of clause (58). Again associativity was unused. As shown by the designation (heat=1), two of the five proof steps were present because of the hot list strategy.

With the goal of gathering more evidence, positive or negative, concerning the potential of the hot list strategy, I next focused on the theorem (*RAT*2) asserting that the adjunction (to the Robbins axioms) of the equation $0+x = x$ implies Boolean. (For the historical minded, when Winker had shown his original interest in Robbins algebra, I suggested the study of adjoining known properties of Boolean algebra, one at a time; the adjunction of the existence of a 0 was one of my earliest recommendations.) In approximately 5 CPU-seconds, OTTER produced a proof—still keying on the denial of Huntington's axiom—of length 12 and level 8, with the retention of clause (561). Associativity still was not used.

Regarding the companion experiment, again I chose for the (input) hot list only the special hypothesis, namely, the clause equivalent of $0+x = x$. I assigned the value 1 to the heat parameter. In approximately 6 CPU-seconds, OTTER produced a proof of length 12 and level 8, with the retention of clause (572). The experiment exhibited more of the charm (at least for me) of mathematics and logic: In addition to the lack of the use of associativity, one finds that the proof produced *with* the hot list strategy is the same as that produced *without* it. Ordinarily, such an occurrence is unlikely, for the path through the possible conclusions to be drawn is quite different in the two cases.

The next theorem I studied (*RAT*3), without and with the hot list strategy, focuses on the first suggestion I made to Winker: the adjunction to the Robbins axioms of the property that the union of x and $x = x$ implies Boolean. Again, the target was Huntington's axiom. In approximately 90 CPU-

seconds, OTTER produced a proof of length 23 and level 12, with retention of clause (809). Associativity *is* used, perhaps giving one the feeling that this theorem offers a bit more depth than the two preceding theorems that I discussed. When I used a hot list with the sole member the clause equivalent of $x+x = x$ and with the assignment of the value 1 to the heat parameter, in approximately 2 CPU-seconds, OTTER produced a proof of length 23 and level 11, with retention of clause (332). Obviously, I think of this two-part experiment as offering a plum because of the impressive reduction in CPU time to produce a proof when the hot list strategy was used.

I now come to a delightful and, to me, remarkable theorem (*RAT*4), one of many results obtained by Winker's excellent research. Indeed, rather than adjoining $x+x = x$ for all x, one need only posit the existence of one such element c with $c+c = c$ to obtain a set of axioms that implies Boolean. In approximately 19 CPU-seconds, OTTER produced a proof of length 41 and level 17, with retention of clause (908). Then, with the hot list strategy, in approximately 3 CPU-seconds, OTTER produced a proof of length 36 and level 18, with retention of clause (423). For each half of this experiment, I used the denial of the Huntington axiom; for the second half, as I have been reporting, I used a hot list consisting only of the special hypothesis and assigned the value 1 to the heat parameter. Both proofs relied on the use of associativity.

In preparation for one of the deeper theorems (*RAT*5) proved in this subsection, I studied two additional theorems, *RAT*6 and *RAT*7, each focusing on the adjunction (to the Robbins axioms) of a single equation. For background, I first note that Winker in his original study of Robbins algebra asked me for a notion of ordering the elements of such an algebra. No doubt, he had a notion that an ordering might eventually lead to answering the open question: Is every Robbins algebra a Boolean algebra? I suggested the following: The element d is less than or equal to the element c if and only if the union of d and c is c. Winker eventually proved the theorem (*RAT*5) that the adjunction of the equation $c+d = c$ to the Robbins axioms implies Boolean, where c and d are two elements (constants) with no other assumed properties; I return to this theorem in Section 4. To me, this property seems indeed weak to offer what is needed to get to a Boolean algebra.

I was told by Winker that the two promised theorems offer a difficulty somewhere between that just discussed and that focusing on the adjunction of $c+c = c$. The first of the two theorems, *RAT*6, focuses on adjoining the following clause.

```
EQ(+(c,+(c,d)),+(n(c),d)).  % hypothesis
```

As with the preceding theorems focusing on Robbins algebra, I conducted experiments whose objective was to gain more data for evaluating the power of the hot list strategy. I also thought that the use of the dynamic hot list strategy might prove interesting. I report the experiments in historical order in part to illustrate how I approach research.

Because of Winker's estimate of the difficulty of the theorem under discussion, I began with a wild guess at an effective attack. Among other options, I used the hot list strategy, assigned the heat parameter the value 2, included thirty resonators (taken from a proof of the theorem focusing on $c+c = c$), assigned the max_weight the value 30, assigned the pick_given_ratio the value 3, assigned the max_distinct_vars parameter the value 3, and placed in the hot list the special hypothesis and the three Robbins axioms. Each resonator was modified by replacing all occurrences of c with $(1) to permit the corresponding term to be any constant or any variable. In deference to the expected difficulty, with the following weight template, I also employed McCune's tail strategy, which gives preferences (in this case) to equations whose second argument is short.

```
weight(EQ($(1),$(2)),1).
```

I was therefore virtually forced to replace weight_list(pick_and_purge) with weight_list(pick_given), which *purges* deduced clauses by symbol count and *picks* them (as the focus of attention) based on the weight templates that are included and, if none apply, based on symbol count. Finally—because I conjectured that more conditions were needed to detect a completed proof—in addition to the inclusion of the denial of Huntington's axiom, I also included, in list(passive), the clause equivalent of $x+x \; != \; x$. In other words, I was taking advantage of the proved theorem that the existence of a constant that, when added to itself, yields itself is enough to imply Boolean in the presence of the three axioms for a Robbins algebra. (Such wild guesses for an initial attack are not uncommon in my experimentation.)

OTTER produced *no* proofs. However, based somewhat on intuition and somewhat on prior experiments that suggested that removal from the hot list of the third axiom for Robbins algebra, I conjectured that I was close to finding an effective approach to seeking a proof.

To obtain a benchmark for comparing the usefulness (in this context) of the hot list strategy, for the first of the seven experiments, I dropped the use of a hot list. For some of the seven, I also used the resonance strategy. OTTER succeeded in producing a proof, one in which the added denial (rather than the denial of Huntington's axiom) played a role. Therefore, I had a proof of the theorem under attack, for I had already proved (*RAT*4) that the adjunction to the Robbins axioms of $c+c = c$ for any c implies Boolean. The proof was obtained in approximately 686 CPU-seconds, with length 12 and level 7, with retention of clause (6563).

Immediately, in the second of the seven experiments, I turned to the dynamic hot list strategy—rather than, as in the experiments reported earlier, the hot list strategy—instructing OTTER to adjoin during the run any clause that matched one of the thirty resonators cited earlier, those that were modified as discussed. I assigned the (input) heat parameter the value 2. I leaned slightly toward the possibility that the CPU time would be reduced, which in part motivated my experiment. In approximately 6613 CPU-seconds, OTTER produced a proof of length 18 and level 7, with retention of clause (17171). Five of the deduced proof steps have heat=1 as part of their history, and five have heat=2.

Disappointment or annoyance caused me to conjecture that a reduction in the pick_given_ratio might increase the effectiveness of the attack. Perhaps heavier (more complex) clauses retained earlier would be useful in reducing the CPU time. Therefore, I assigned the pick_given_ratio the value 1 and made no other changes in the third experiment. In approximately 21,065 CPU-seconds, OTTER produced a proof of length 19 and level 7, with retention of clause (21133); OTTER reported the length as 20 due to the duplicated step resulting from the use of the dynamic hot list strategy. Six of the deduced proof steps have heat=1 as part of their history, and seven have heat=2.

This unsatisfying result led me to the fourth experiment, where I relied on the resonance strategy (keying on the already-discussed thirty resonators) and omitted use of either variant of the hot list strategy. I also returned to my typical assignment to the pick_given_ratio, namely, 3. In approximately 24,930 CPU-seconds, OTTER produced a proof of length 17 and level 6, with retention of clause (13063). However, unexpectedly, this proof, when compared with the preceding, contains nine different steps.

A review of the preceding success followed by the disappointments led me to the fifth experiment. I returned to using the hot list strategy, abandoned the resonance strategy (essentially), assigned the value 2 to the heat parameter, and removed from the (input) hot list the third axiom of Robbins algebra. Thus the hot list consisted of the clause equivalents of the special hypothesis (of the theorem under attack) and commutativity and associativity of union. In approximately 241 CPU-seconds, OTTER produced a proof of length 12 and level 7, with the retention of clause (5365). Five of its steps are not among those of the earlier-cited 12-step proof, two of its steps have heat=1, and one has heat=2.

Witnessing this advance in efficiency, I turned to the sixth experiment. I replaced the assignment of the value 2 to the heat parameter with the value 1, and made no other changes. In approximately 523 CPU-seconds, OTTER produced a proof of length 53 and level 16, with retention of clause (11846). Of the deduced steps, twenty-five have heat=1. A comparison with the preceding experiment shows that, in terms of percentages, the hot list played a bigger role, but it also shows that the assignment of the value 2 to the heat parameter (in this case) was most beneficial if measured in CPU time.

Although I was not optimistic, just for curiosity I conducted a seventh experiment. I modified the sixth by removing two more elements from the hot list, namely, commutativity and associativity of union. In other words, the only member present in the hot list for the experiment was the special hypothesis; the heat parameter was still assigned the value 1; no resonance to speak of was used, and no new clauses were adjoined to the hot list during the run. In approximately 25 CPU-seconds, OTTER produced a proof of length 17 and level 7, with retention of clause (1846). Just two of the deduced steps are present because of the actions of the hot list strategy.

Immediately, one might naturally ask for some conjecture concerning the marked improvement in efficiency, when compared with any of the other six experiments, especially since just two of the

deduced proof steps result from the use of the hot list strategy. Perhaps the hot list strategy enabled the program to avoid traversing a wide valley of low-weight clauses that would otherwise have consumed much CPU time in their consideration, and perhaps, by immediately though briefly focusing on complex clauses, the strategy enabled the program to cross a wide plateau of high-weight clauses. Perhaps also the program benefited from a small hot list, and perhaps one should frequently use this strategy with the only member(s) of the hot list being those corresponding to the special hypothesis of the theorem under attack. I am partial to the conjectured conclusions, and I give the following pertinent evidence.

The second promised theorem, *RAT*7, (suggested by Winker) concerns proving Boolean from adjoining to the Robbins axioms the following clause.

EQ(+(c,d),+(n(c),+(n(c),d))). % hypothesis

Of my experiments, I cite two, the first *without* the hot list strategy and the second *with* it under the conditions of the seventh experiment just discussed. Without the hot list strategy, in approximately 74 CPU-seconds, OTTER produced a proof of length 18 and level 8, with retention of clause (2785). With the hot list strategy, in approximately 87 CPU-seconds, OTTER produced a proof of length 18 and level 8, with retention of clause (3262); none of the deduced steps are present because of the hot list strategy.

Although not directly germane to an analysis of the power of the hot list strategy, I note that one additional experiment merits mention. In that experiment, to test the usefulness of using the tail strategy, I revisited the benchmark experiment for the first of the two theorems just discussed. Except for omitting the tail strategy and replacing weight_list(pick_given) with weight_list(pick_and_purge), I chose the same options. Therefore, in my revisit, clauses were in general chosen (as the focus of attention to drive the program's reasoning) based on symbol count, rather than on their weight computed mainly in terms of their respective second arguments. After approximately 4 CPU-hours, the run was terminated with no proof.

I close this subsection by briefly focusing on *RAT*5, a theorem that provides a splendid challenge for automated reasoning programs, especially those that do *not* offer AC-unification or induction. (Winker first proved *RAT*5 with intuition, induction, and assistance from one of Argonne's reasoning programs; later, using one of McCune's unannounced programs, McCune obtained a proof with AC-unification.) This intriguing result of Winker's, which asserts that Boolean is provable from the adjunction (to the axioms of Robbins algebra) of the equation $c+d = c$ for constants c and d, also provides a good test for evaluating new ideas. In terms of the ''ordering relation'' on the elements of Robbins algebra (suggested by me to Winker), the theorem says that the existence of two elements c and d with d less than or equal to c together with the Robbins axioms is all that is needed to imply Boolean. (That the adjunction of any number of properties—among which is the cited extremely weak property—one at a time, is enough to imply Boolean led me from the early 1980s to the conjecture that *all Robbins algebras are Boolean algebras*, a conjecture that McCune proved in late 1996 to be true [McCune97]. Obviously, Robbins himself thought the theorem to be true, or he would not have offered the axiomatization for Robbins algebra; that offering occurred in the mid-1930s.) Of the theorems discussed in this subsection, this last one has for years presented me with a goal: to prove it without induction and without AC-unification. Because various colleagues have expressed puzzlement at my seeking this goal (in view of the existing proofs of the theorem), I remark that, for the following reason, this goal is consistent with my global approach to research designed to increase the power of reasoning programs. Indeed, I find that important advances sometimes occur by imposing some seemingly arbitrary constraint (such as blocking the use of AC-unification) on the program's attack and then attempting to formulate a method that nevertheless finds a proof.

In the case of *RAT*5, numerous attempts on my part to find a sufficiently powerful combination of strategies and options had failed. These failures coupled with the successes reported earlier in this subsection suggested this theorem as an excellent test for study with McCune's tail strategy and the hot list strategy. Directly as a consequence of my earlier experiments, I assigned max_weight the value 30, the pick_given_ratio the value 3, and max_distinct_vars the value 3 and included clear(eq_units_both_ways). Regarding weight templates, in weight_list(pick_given), I included two to respectively purge associative variants in four and five variables, one to purge expressions in which $n(n(n(t)))$ terms occur, and the template for the tail strategy. The inclusion of the template for the tail strategy causes OTTER to prefer clauses in the equality predicate whose right-hand argument is short.

Success: In approximately 9770 CPU-seconds, OTTER produced a proof (given in the Appendix) of length 78 and level 16, with retention of clause (48308). Almost required, I repeated the experiment, but with the addition of the hot list strategy. Of the choices, I thought the most instructive would be to assign heat the value 1 and place in the (input) hot list only the clause corresponding to the special hypothesis, $c+d = c$. In approximately 44,926 CPU-seconds, OTTER produced a proof of length 80 and level 18, with retention of clause (66147). When the heat parameter was assigned the value 2 rather than 1, the results were essentially unchanged. Without the hot list strategy, 986 clauses were chosen as the focus of attention; with it 1820 and 1822 were chosen for the respective assignments of 1 and 2 to the heat parameter.

Although I clearly would have preferred the hot list strategy to have shown itself well for the theorem, finding a proof in which neither induction nor AC-unification played a role was indeed satisfying. As far as I know, such had not occurred before.

3.4. Elegant Proofs and the Hot List Strategy

One measure of the elegance of a proof is its brevity. In this subsection, I show how the hot list strategy has proved useful in the search for elegant proofs, in the context of proof length. I note that no practical algorithm appears to exist for searching for short proofs, and I note that numerous obstacles, some of which are indeed subtle, are encountered in such a search; see my new book [Wos96a], which takes the form of an experimental notebook. Not only am I sometimes preoccupied with the search for short proofs, but I have verified (through direct contact) that various mathematicians and logicians share my interest, including some of the more famous.

I begin the discussion by focusing on two-valued sentential (or propositional) calculus, using as axioms $L1$, $L2$, and $L3$ (also known, respectively, as theses 1, 2, and 3); see Section 3.2 for theses by number, including those I study here. As part of the discussion, I include comments to further illustrate how I approach research and experimentation.

For the historians and for background, my first success was obtained as a byproduct of my agreeing to assist McCune in verifying that the hot list was in fact working as intended. For one of my first tests, I chose to have OTTER use the hot list strategy in the context of deducing the Church axiom system (consisting of theses 18, 35, and 49) from the Lukasiewicz axiom system. To save CPU time and also to quickly accrue evidence of the correctness of the corresponding implementation, I used weight templates corresponding to a 22-step proof I had found after substantial experimentation; I call that proof Proof 4 and present it in my new book on experimentation [Wos96a]. To each weight template (corresponding to one of the 22 steps of the chosen proof), I assigned a weight of 2; in other words, I was also using the resonance strategy. My expectation was that OTTER would quickly reproduce the 22-step proof, Proof 4.

Such was not the case. Instead, OTTER produced a 21-step proof—called Proof 5 in the cited new book—which was remarkable (to me), for I had for months sought with no success a proof with strictly fewer than 22 applications of condensed detachment. In addition to finally reaching one of the objectives (in the context of shorter proofs), perhaps more important, I had my first evidence that the hot list strategy was useful in finding proofs more elegant than previously known. The study also produced proofs numbered 6, 7, and 8 (presented in the cited book); Proof 6 (given in the Appendix) was obtained with the aid of ancestor subsumption, and Proof 7 and Proof 8 were obtained with the hot list strategy focusing on a 30-step proof of the Frege axiom system. The Frege axiom system consists of theses 18, 21, 35, 39, 40, and 46, of which thesis 21 is dependent.

In part because OTTER had undergone further development and in part to verify my results for this report, I revisited my experiment of using as resonators the steps of Proof 4, again emphasizing the role of the hot list strategy. Obviously, I expected the program to reproduce Proof 5—but, instead, Proof 6 was produced, and no other proof of the Church axiom system. Did the explanation rest with a faulty recording (on my part) of history, or did it rest with some change to OTTER? Fortunately, I was able to retrieve the input file used in the original experiment (that produced Proof 5) and compare it with what I had just done. Indeed, no change had been made to the heat parameter (assigned the value 1), but the max_weight was assigned the value 20 to produce Proof 5 and assigned the value 2 to produce Proof 6. The extra latitude permitted with the higher max_weight clearly allowed the program to

traverse a different search path. (The assignment of the value 2 to the max_weight coupled with assignments of 2 to the resonators is a technique I often use to proof check, in a cursory fashion, a result.) I thus have a pleasing example of the use of the hot list strategy deflecting a program's attempt to proof check in a cursory manner and, instead, producing a more elegant proof than that being proof checked.

Naturally, three additional experiments were demanded, obtained by replacing the 22 resonators corresponding to Proof 4 with, respectively, the 22 from Proofs 1, 2, and 3. The object of each of the three experiments was to see whether the use of the hot list strategy would deflect the cursory proof checking that would ordinarily occur with an assignment of the value 2 to the max_weight and assignments of 2 to each of the proof steps used as resonators. In the experiment in which the steps from Proof 1 are used as resonators with the hot list, Proof 1 is all that results. The Proof 2 resonators, however, produce Proof 8. The Proof 3 resonators merely produce Proof 3. So, in the obvious sense, what happened to Proof 5 and to Proof 7?

Not with the goal of again finding Proofs 5 and 7, I decided to see what might occur if I replaced the 22 resonators from one of my proofs of the Church axiom system with those of the shortest proof I had ever seen of the Frege axiom system, a proof of length 30. To be candid, I was only going through the motions—behaving as a scientist should—knowing that much effort (without success) had been expended by me in seeking a proof with strictly fewer than 30 applications of condensed detachment. nevertheless, unexpectedly and to my delight, the hot list strategy prevailed over cursory proof checking: OTTER produced a 29-step proof of the Frege axiom system, one of level 16. Although no subproof relation holds, all formulas in the 29-step proof appear in the 30-step proof; the following clause appears in the 30-step proof and not in the 29-step proof.

P(i(i(i(x,y),z),i(y,z))).

In addition to finally finding an even shorter proof of the Frege axiom system, in the same run, OTTER produced Proof 7 (of length 21) of the Church axiom system.

I next turned to the Hilbert axiom system as target, consisting of theses 3, 18, 21, 22, 30, and 54, of which thesis 30 is dependent. As resonators, I used weight templates corresponding to the 24 steps of a proof I had found (during the writing of the cited new book) when seeking shorter proofs. I knew of a 23-step proof (also produced with my experimentation with OTTER) of the Hilbert axiom system, and wondered whether it would be found again. With an assignment of the value 1 to the heat parameter (as I had been doing), the known 23-step proof (given in the Appendix) was found and no others of the Hilbert axiom system. I decided to turn up the heat, assign the value 2 rather than 1 to the heat parameter. OTTER returned a 23-step proof, but a proof I had never seen before, differing from the known 23-step proof by one step, the new step being the following.

P(i(i(x,i(y,z)),i(z,i(x,z)))).

From the viewpoint of the order of the proof steps, the two 23-step proofs are almost identical. (Turning up the heat in the context of the experiments focusing on the Church axiom system produced no interesting results, nor did turning it up further for Hilbert.)

Then I applied the cited approach to the alternate Lukasiewicz axiom system consisting of theses 19, 37, and 59. I used as resonators weight templates corresponding to the steps of a 25-step proof I had found with OTTER. The objective was to find a proof with strictly fewer than 24 applications of condensed detachment—I already had a proof of that length in hand—or a different 24-step proof. With the heat parameter assigned the value 1, OTTER reproduced the known 24-step proof. However, turning the heat up by assigning the value 2 to the heat parameter resulted in the program finding a different 24-step proof, only one of whose steps (the following) is not present in the earlier-discovered 24-step proof.

P(i(i(x,i(y,z)),i(z,i(x,z)))).

To complete the application of the given approach to seeking more elegant proofs by using the hot list strategy (in the study of two-valued sentential calculus), I turned to an axiom system I discovered, consisting of theses 19, 37, and 60. For resonators, I used correspondents to the 26 steps of a known proof (I had found) of this axiom system, keeping in mind that the shortest proof of which I

know has length 25. When the heat parameter was assigned the value 1, OTTER reproduced the cited 25-step proof. However, when I turned up the heat by assigning the value 2 to the heat parameter, a new 25-step proof was found, only one of whose steps (the following) is absent from the first 25-step proof.

P(i(i(x,i(y,z)),i(z,i(x,z)))).

I next turn to equivalential calculus [Lukasiewicz70]. I focus on my two nemeses, the formulas known as *XHK* and *XHN*, found among the following complete list of the shortest single axioms (each expressed in clause notation) for this area of logic.

```
% Following are all of the shortest single axioms for equivalential calculus.
P(e(e(x,y),e(e(z,y),e(x,z)))).  % P1_YQL
P(e(e(x,y),e(e(x,z),e(z,y)))).  % P2_YQF
P(e(e(x,y),e(e(z,x),e(y,z)))).  % P3_YQJ
P(e(e(e(x,y),z),e(y,e(z,x)))).  % P4_UM
P(e(x,e(e(y,e(x,z)),e(z,y)))).  % P5_XGF
P(e(e(x,e(y,z)),e(z,e(x,y)))).  % P7_WN
P(e(e(x,y),e(z,e(e(y,z),x)))).  % P8_YRM
P(e(e(x,y),e(z,e(e(z,y),x)))).  % P9_YRO
P(e(e(e(x,e(y,z)),z),e(y,x))).  % PYO
P(e(e(e(x,e(y,z)),y),e(z,x))).  % PYM
P(e(x,e(e(y,e(z,x)),e(z,y)))).  % XGK
P(e(x,e(e(y,z),e(e(x,z),y)))).  % XHK
P(e(x,e(e(y,z),e(e(z,x),y)))).  % XHN
```

Actually, the two formulas *XHK* and *XHN* are my ‘‘friends’’; their study throughout the years has brought me much satisfaction, for which I am forever indebted to John Kalman.

Each of *XHK* and *XHN* alone is strong enough to provide a complete axiomatization for equivalential calculus. To prove either of the corresponding theorems (by deducing one of the other known single axioms) provides an excellent test for ideas and for programs. (There exist thirteen shortest single axioms, which were listed earlier, for this area of logic, each of length 11, excluding commas and parentheses.) Indeed, whether either is a single axiom was an open question until Winker obtained proofs with excellent insight, many computer runs, much time, and considerable assistance from one of Argonne’s automated reasoning programs [Wos83,Wos84a]. As an indication of the difficulty offered by the two benchmark theorems, (not counting the predicate *P*) Winker’s 84-step proof for *XHK* relies on the use of a formula of length 71, and his 159-step proof for *XHN* relies on the use of a formula of length 103. From *XHN*, his proof completes by deducing the shortest single axiom *UM*. From *XHK*, his proof completes by deducing the shortest single axiom *YRO*.

From various experiments (whose complete history is, unfortunately, currently lost), I had managed to find a 27-step proof showing that *XHK* is a single axiom and a 24-step proof showing that *XHN* is also a single axiom. I am almost certain that the methodology relied on, among other mechanisms, ancestor subsumption, level saturation, and the resonance strategy; less certain, I believe that the hot list strategy was not used. Although not algorithmic, the methodology is one of iteration, covered in copious detail in my new book. Motivated by the material in this report, I decided to use the hot list strategy to attempt to find proofs that *XHK* and that *XHN* are each a single axiom, proofs with, respectively, strictly fewer than 27 and 24 applications of condensed detachment.

I began with *XHN*, in part because I knew a proof existed at level 11, and in part because I knew I would most likely turn to level saturation at some point. I assigned max_weight the value 2 and assigned the value 2 to each of 24 resonators corresponding to the steps of the known 24-step proof that *XHN* implies *UM*, a known single axiom for equivalential calculus; see the earlier list of such formulas. I included in list(passive) the negations of each of the other twelve shortest single axioms, expecting a deduction of *UM* only. I used the hot list strategy, assigning the value 1 to the heat parameter, and placed in the hot list the clauses corresponding to *XHN* and the condensed detachment nucleus. Clearly, I was mirroring my approach to seeking shorter proofs of axiom systems in two-valued sentential calculus (discussed earlier). Such is my practice: If an approach succeeds often in one area, then

in a related area—even if distantly related—the approach merits revisiting. In this case, OTTER merely reproduced the known 24-step proof.

I immediately turned to level saturation and increased the max_weight to 36, knowing from earlier experiments that 36 suffices for obtaining a proof. Note that, among the 24 resonators, four have length 40 (measured in symbol count). In approximately 38 CPU-seconds (on the equivalent of a SPARCstation-2), OTTER deduced *UM* with a proof of length 22 and level 11, with retention of clause (864). The obvious breakthrough caused me to press on, first seeking information regarding what role the hot list strategy played in the success. In the same run, OTTER deduced *XGF* in approximately 993 CPU-seconds; the proof has length 23 and level 12 and completes with retention of clause (15376). (For the connoisseur, I note the following. The proof that completes with *UM* is, as one might suspect, a subproof of the proof that completes with *XGF*, the latter merely having the one additional step, the condensed detachment of *UM* with itself.)

The simplest experiment that occurred to me was to comment out the heat parameter and the hot list and otherwise repeat the experiment that produced the 22-step proof. In approximately 23 CPU-seconds, OTTER deduced the formula *UM* with a proof of length 24 and level 11, with retention of clause (471). In the same run, in approximately 211 CPU-seconds, OTTER deduced *XGF* with a proof of length 25 and level 12, with retention of clause (3332). Tentatively, I concluded that the hot list strategy had in fact played an important role in finding the shorter proof. After all, eight steps of the 22-step proof are present directly because of the use of the hot list strategy, each showing in its history (heat=1).

To check further that indeed the hot list strategy was valuable in the breakthrough, I repeated the preceding experiment with one change: I deleted the use of the resonance strategy. In particular, by commenting out the weight templates that correspond to the steps of the cited 24-step proof and still assigning the value 36 to max_weight, I prevented OTTER from retaining clauses it might otherwise retain. For example, in the preceding experiment, 25 clauses were retained and chosen as the focus of attention that match (not necessarily identically) one of the 24 resonators. Since four of the resonators have weight (measured in symbol count) in excess of 36, one sees that the program would almost certainly traverse a different search path.

Indeed, such was the case. In approximately 91 CPU-seconds, OTTER deduced *UM* with a proof of length 21 and level 14, with retention of clause (1229). The result was both satisfying and disappointing, satisfying because an even shorter proof had been found, disappointing because the hot list strategy played no role in the success. Some might say that disappointment is not in order, for one is not supposed to root for any particular strategy; but I do, as many researchers no doubt know.

Therefore, I made one additional experiment, like the preceding except for adding the use of the hot list strategy. In approximately 770 CPU-seconds, OTTER deduced *UM* with a proof of length 20 and level 14, with retention of clause (9777). Three formulas, the following, were present in the 21-step proof and not in the 20-step proof.

P(e(e(x,e(y,e(e(z,u),e(e(u,y),z)))),e(e(v,w),e(e(w,e(x,e(v6,e(e(v7,v8),e(e(v8,v6),v7))))),v)))).
P(e(e(e(e(e(x,y),e(e(y,z),x)),u),z),e(u,e(v,e(e(w,v6),e(e(v6,v),w)))))).
P(e(x,e(e(y,z),e(e(z,e(e(u,e(e(v,w),e(e(w,u),v))),x)),y)))).

Immediately, one might wish a hint concerning the success with the hot list strategy, a notion of how it helped find an even shorter proof. A key rests with the effect the strategy has when level saturation is being used. For an illustration, assume the heat parameter is assigned the value 1, that condensed detachment is in use, and that the hot list contains the needed clauses (such as that for condensed detachment and, say, the shortest single axiom candidate under study). When a level-1 clause *A* is deduced and retained and the hot list strategy is in use, *A* will immediately be used to initiate applications of condensed detachment with the clauses needed to complete the applications chosen from the hot list. If condensed detachment succeeds yielding a clause *B*, *B* will have level 2 (and heat level 1). If *B* is retained, then it will be placed among the level-1 clauses, even though it has level 2. Keep in mind that, very likely, the program is still generating level-1 clauses and simply paused, because of the use of the hot list strategy. Then, when the program is using the level-1 clauses to deduce those of level 2, *B* will be used; but its use will generate level-3 clauses, which, if retained will be placed

among the level-2 clauses. In addition, because of the use of the hot list strategy, such a level-3 clause *C* will be used immediately. If clauses are deduced and retained, they will be of level 4, but be placed among those of level 2, just as *B* was deduced and placed among the level-1 clauses.

Regarding the successful completion of the cited 20-step proof, a glance at the output file shows that the program was deducing and retaining clauses of level 11 when it found and used a level-14 clause to complete the proof. In other words, the program, because of using the hot list strategy in conjunction with level saturation, was able to look ahead into higher levels. One thus sees how the program found a different proof, one of length 20, by traversing a sharply different search path.

To determine the effect on CPU time, on a computer that is perhaps 1.3 times as fast as a SPARCstation-10, I conducted two experiments. Approximately 38 CPU-seconds suffices with level saturation and without the hot list strategy, in contrast to approximately 306 CPU-seconds with the combination of level saturation and the hot list strategy. Again, for part of the explanation, one need only glance at the corresponding output files, finding that level 10 completes with clause (87) when the hot list strategy is not in use, and level 10 completes with clause (1488) when it is in use. These two figures further illustrate how the hot list strategy, when level saturation is in use, causes the program to look ahead into higher levels. The figures also illustrate a disadvantage of using this combination of strategies, for the size of the levels can grow far more rapidly.

- Summarizing, the shortest proof (at this point in my experimentation) that establishes *XHN* to be a single axiom for equivalential calculus was obtained (by OTTER) with the hot list strategy, level saturation, no weight templates, and a max_weight of 36; it completed by deducing the formula *UM*. The only members of the (input) hot list were the clause for condensed detachment and that corresponding to *XHN*. Six of the 20 proof steps have length 36 (measured in symbol count including the predicate)—and none are longer—five of the steps have heat=1, and ten of the steps are not among those of the 24-step proof that prompted this study.
- When a more elegant proof than one has in hand is sought, a useful although perhaps surprising approach focuses on the use of level saturation, especially if coupled with the use of the hot list strategy; this combination enables a program to look ahead into higher levels than being currently examined.

As one more example of how I approach experimentation and research—and of especial interest in view of the result—I cite the following. Because cursory proof checking occasionally leads to a even shorter proof, I took the 20-step proof that *XHN* implies *UM* and used its steps as resonators, assigning the value 2 to each; I also assigned max_weight the value 2. I used the hot list strategy, for its use in the context of cursory proof checking sometimes yields a shorter proof than that whose steps are used as resonators. OTTER merely returned the same 20-step proof. On a whim, I repeated the experiment except for commenting out the heat parameter and the hot list. I was more than startled to find that OTTER completed a 19-step proof of level 14 (given in the Appendix), not a subproof of the 20-step proof. However, the 20-step proof contains but one formula, the following, not in the 19-step proof.

P(e(e(x,y),e(e(y,e(e(e(e(z,e(e(u,v),e(e(v,z),u))),w),w),e(v6,e(e(v7,v8),e(e(v8,v6),v7))))),x))).

It seemed immediately pressing to explain how such could occur, particularly because I had never thought that removal of the hot list strategy could also lead to proof shortening; in other words, I had not previously encountered what might be termed the converse (of adding the hot list strategy).

The situation with which I was familiar concerns how cursory proof checking a proof produced *without* the use of the hot list strategy can lead to a shorter proof when the strategy is added. For an illustration of how such can occur, assume the following. Assume the proof contains in order, not necessarily consecutively, deduced steps *A*, *C*, *D*, *E*, *F*, *B*, and *G*, where *G* is the last step of the proof. Assume that the choice of *A* to initiate applications of condensed detachment leads to the deduction of both *C* and *B*. Assume that the choice of *C* leads to the deduction of *D*, the choice of *D* to *E*, the choice of *E* to *F*, and the choice of *B* to *G* with the other parent being *F*. Assume that the condensed detachment of *B* and an (input) axiom *J* yields a second copy of *F*, which is then forward subsumed. As for the proof shortening resulting from the use of the hot list strategy, assume that the (input) hot

list contains *J* and the nucleus for condensed detachment and that the heat parameter is assigned the value 1. When the hot list strategy is added to the resonance strategy (with the proof steps from the proof produced without the hot list strategy used as resonators), the retention of *B* immediately leads to the deduction of *F*, which is not forward subsumed in this case. Then, the deduction of *F* from *E* results in this second copy (of *F*) being forward subsomed. Therefore, with the finer details omitted, the sequence *A*, *C*, *D*, *E*, *F*, *B G* is replaced with the sequence *A*, *B*, *F*, *G*. In other words, adding the use of the hot list strategy leads to a shorter proof.

The just-discussed illustration can be quickly modified to show how the omission of the hot list strategy can lead to a shorter proof than was completed with its use. Assume that the choice of *A* immediately leads to the deduction of both *H* and *B* with the complexity of *H* strictly greater than that of *B*. Assume that the use of the hot list strategy and the retention of *H* yields *C*, closely followed by *D*, *E*, *F*, *B*, and *G*. Then, when the hot list strategy is omitted and the result imitates the preceding illustration, *B* is chosen over *H*, and the shorter proof is completed by relying on the sequence *A*, *B*, *F*, *G*.

In addition to the two examples just given, a third example merits brief discussion, another example of how omitting the use of the hot list strategy can lead to a shorter proof. Such examples can lead to insight into the intertwining of the various mechanisms on which an automated reasoning program of substance is based. The example assumes that the first step *A* in the proof is yielded by applying condensed detachment to the hypothesis *S* (the shortest single axiom) with itself. Then, from the choice of *A*, two proof steps are deduced, *B* and *C*, and from *C D* is deduced immediately because of the use of the hot list strategy. Next, assume that *C* serves no other purpose in the proof other than the deduction of *D*. Then assume that from *B* the program deduces a second copy of *D*, which is forward subsumed. When the hot list strategy is omitted, the program can seek a proof in which the occurrence of *D* that is retained has *B* as one of its parents. The clause *C* will not be needed, and the resulting proof will contain one less step. This example also serves nicely to illustrate that even the use of ancestor subsumption would not have yielded the shorter proof, for both paths to *D* have length 3.

The finding of a 19-step proof by keying on the 20-step proof can be put to further use, that of showing how an idea or an approach is born, at least showing how my research sometimes works. Not directly because of noting that exactly one of the formulas in the 20-step proof fails to occur in the 19-step proof, but no doubt on some level triggered by that knowledge, it occurred to me that the following merited investigation. The idea is to cursory proof check a given proof with the intent of finding a shorter one, but, instead of using *all* of the proof steps of the given proof as resonators, one attempts to identify at least one that is conjectured to be unnecessary. By unnecessary, I mean that a proof exists—not a subproof of the given proof—consisting of all of the remaining steps of the given proof, all but the so-called dispensable one. Of course, rearranging and changing parenthood is more than permitted. The burden thus shifts to finding a means for identifying a possibly dispensable step. My first guess, and one that I think is obvious once the question is posed, suggests that one seek a step in the given proof that is used exactly once as a parent in a later step.

Rather than (as one might expect and predict) examining the 20-step proof to see if the step not used in the 19-step proof has the stated property, I instead read through a proof focusing on the shortest single axiom *XHK*. An explanation is in order. I was most intent on finding a means to improve on what I had in hand regarding a proof that *XHK* is indeed a shortest single axiom; that goal was for me more pressing than seeing what could be done with the 20-step proof focusing on *XHN*. When I did in fact examine the 20-step proof, I found that the so-called unneeded step is used only once as a parent. When that step was removed from the resonators and the input file was otherwise unmodified—how pleasing!—the same 19-step proof was produced, in particular, with the hot list strategy active. Just to cover the obvious missing point, most such potentially dispensable steps (with which I experimented) fail to yield a proof when removed from the set of resonators. Nevertheless, I feel certain I shall add this approach to the arsenal of weapons for seeking shorter proofs.

- A charming example of how piquant can be some area of mathematics or logic when studied with an automated reasoning program is provided by the experiment using level saturation to prove that *XHN* is a single axiom, for one can have simultaneously almost the best of all worlds. The experiment yields a 21-step proof, and the shortest known to me has length 19; the most complex

formula in the proof has length 36 including the predicate symbol, and no proof exists if formulas are limited to length 32; and the proof was completed in approximately 38 CPU-seconds on a computer roughly 1.3 times faster than a SPARCstation-10.

Before leaving *XHN*, the following historical oddity merits visiting. Winker, with more than one stroke of brilliance, was able to answer the open question concerning *XHN* being a single axiom. His success marked a significant achievement for the field of automated reasoning. However, to take nothing away from his excellent result, as it turns out, the proof was actually within reach (in the early 1980s) of one of our programs, at least theoretically within reach. The only move that was required was to use level saturation, which was offered by the program in use, choose a max_weight of 36, and wait perhaps xxx CPU-seconds. (I believe we were using an IBM 370/195, which is perhaps 70 to 100 times slower than a SPARCstation-10.) Just over 1 megabyte of memory would have sufficed. I would prefer to say that level saturation was not used because, at that time, it would have been too expensive; we paid for computer use out of a not large budget. However, the key point—the truth—is that none of us thought much of level saturation, especially for attacking deep questions.

Perhaps more significant than the fact that the proof could have been found with far less effort is what one finds in a comparison of then and now. Specifically, if one glances at but four experiments, one finds evidence of the marked progress that has been made in the area of strategy formulation. In all four experiments, OTTER was instructed to seek as many as 10 proofs, restricted by using no more than 120 megabytes; the computer used is 1.3 times faster than a SPARCstation-10. In the first experiment, level saturation was the choice, and three shortest single axioms were deduced, namely and in order, *UM*, *XGF*, and *YRM*. The respective times in CPU-seconds are 38, 492, and 25450; the respective proofs lengths are 21, 22, and 40. The second experiment utilized the hot list strategy in addition to level saturation, deducing one shortest single axiom, *UM*, with a proof of length 20.

The third experiment was similar to the first except that the ratio strategy replaced level saturation; the pick_given_ratio was assigned the value 1. One other feature was used, that of reducing max_weight to 24 after 50 clauses had been chosen as the focus of attention. This assignment causes the program to choose clauses to initiate applications of condensed detachment alternately, one by complexity, one by first come first serve, and the like. Eight of the other twelve shortest single axioms were deduced, in order, *UM*, *XGF*, *WN*, *PYM*, *YRM*, *XGK*, *YQJ*, and *PYO*. The first of the eight proofs was completed in approximately 871 CPU-seconds and the last in approximately 5074 CPU-seconds. The length of the first of the eight proofs is 51 and of the last is 81; of the other six, the length ranges from 52 to 74.

The fourth experiment was similar to the third except that the use of the hot list strategy was added with an assignment of the value 1 to the heat parameter. Again, eight shortest single axioms were deduced and in almost the same order, except that *WN* and *PYM* were switched and *YQL* was deduced rather than *PYO*. The last of the eight proofs was completed in approximately 3201 CPU-seconds, and—most significant—the first was completed in approximately 19 CPU-seconds. Regarding proof lengths, the first has length 31, the last has length 56, and the others have lengths between the two values. In other words, due to the use of the ratio strategy combined with the hot list strategy—in addition to deducing seven shortest single axioms—OTTER found a 31-step proof of *UM* in roughly 19 CPU-seconds, which suggests that progress in the area of strategy formulation has indeed occurred. My assertion concerning progress is based on the comparisons among the experiments and on the fact that the question that was answered had resisted the experts for years.

Next I turned to a study of *XHK*, applying a similar approach to that which yielded the 20-step proof that *XHN* is indeed a shortest single axiom for equivalential calculus. I am content to present the results of one of the late experiments. In that experiment, I assigned max_weight the value 48 (because I knew from earlier work that was sufficient), used ancestor subsumption (and, therefore, used back subsumption), assigned the pick_given_ratio the value 3, reassigned the max_weight to the value 20 after 30 clauses were chosen as the focus of attention, and used the hot list strategy with the heat parameter assigned the value 1. I placed in the hot list the clauses corresponding to *XHK* and the condensed detachment nucleus. I used the pick_and_purge weight_list and included, for the resonance strategy, weight templates corresponding to the steps of the earlier-mentioned 27-step proof, which completed with a deduction of *YRO*. Various remarkable (to me) results were obtained, including a 26-step proof that completes with the deduction of *YRO* in approximately 18,756 CPU-seconds. The proof has level

20, and it completes with retention of clause (38781). Three of the proof steps are not among the 27 steps that correspond to the resonators. Four of the steps reflect the use of the hot list strategy, each showing (heat=1). Only one of its steps has length (in symbol count) 48, and none are longer. In addition to producing a 26-step proof, the following items are of particular interest. In approximately 61,514 CPU-seconds, OTTER deduced YQF with a proof of length 32 and level 22, with retention of clause (86649). As far as memory serves, (before the experiment) I cannot recall ever seeing a deduction from *XHK* of YQF.

- Of especial significance (which will be explained shortly), the proof does *not* include the deduction of *any* other shortest single axiom for equivalential calculus.

In approximately 78,273 CPU-seconds, the program deduced *WN* with a proof of length 39 and level 25, with retention of clause (98393).

- Of especial significance (which will be explained shortly), the proof does *not* include the deduction of *any* other shortest single axiom for equivalential calculus.

The ''significance'' rests with the desire to produce, as far as possible, what might be termed a *circle proof*; see [Wos95b,Wos96b]. Such a proof is actually a set of proofs in which one of the thirteen shortest single axioms is used to deduce another *without* using a third shortest single axiom. In other words, with a key proviso, the object is to rearrange the thirteen formulas (in a circle) such that the first can be used to deduce the second, the second the third, and so on, until the thirteenth is used to deduce the first. The proviso is, of course, that no formula k in the rearrangement appear in the proof of i+1 from i, where k is neither i+1 nor i. Although the problem has been solved [Wos95b], one might enjoy an independent attack on it with the aid of some automated reasoning program. A comparable problem (open at the time) focusing on Moufang identities was brought to my attention in the 1960s, a problem now solved with invaluable aid from OTTER, which provides yet more bit of evidence of the progress that has occurred; see [Wos96b}. Whereas the circle problem for equivalential calculus relies in no way on equality, that for Moufang loops does.

Before terminating (for now) my study of *XHK*, one additional experiment merited conducting. My reasoning was simple: If I was able to start with a 24-step proof and eventually find a 19-step proof showing that *XHN* is a single axiom for equivalential calculus, then (not scientifically) I could perhaps do better than a reduction of one in length (starting with a 27-step proof and producing a 26-step proof for *XHK*). A few mildly radical changes in the input file were in order. First, I replaced the 27 resonators (corresponding to the steps of the 27-step proof) with resonators corresponding to the 26-step proof. Next, with the notion of encouraging the use of complex formulas retained early in the run, I decreased the pick_given_ratio from 3 to 2. The assignment of the value 2 instructs OTTER to choose (as the focus of attention to drive the reasoning) two clauses by weight (complexity influenced by the included weight templates), one by first come first serve (level saturation or breadth first), two by weight, one by first come first serve, and the like. Finally, to increase the availability of complex clauses, with the following two commands I made a two-option change: Rather than assigning the new max_weight the value 20 after 30 clauses had been chosen as the focus of attention, I instructed the program to assign the new max_weight the value 24 after 50 clauses had been chosen as the focus of attention.

```
assign(change_limit_after, 50).
assign(new_max_weight, 24).
```

Of the four proofs that were completed, two interesting proofs were produced. The third proof completed with the deduction of *YRO* in approximately 945 CPU-seconds; the proof has length 25 and level 21 and was completed with retention of clause (7139). So the extra experiment proved a wise move. However, to me, far more significant was the second completed proof. In approximately 380 CPU-seconds, OTTER deduced *YQL* from *XHK* with a proof (given in the Appendix) of length 23 and level 19, with retention of clause (4788). To my knowledge, never before this point in my research had

I seen a proof (establishing that *XHK* is a single axiom) where *YQL* was deduced.

That ancestor subsumption (whose use, from a practical standpoint, requires the use of back subsumption) played an important role is demonstrated by the fact that the first proof found is a 26-step proof that deduces *YRO*, a proof matching the 26 resonators that were used. As for the importance of reliance on the hot list strategy, I note that four steps of the cited 25-step proof have heat=1 and four of the 23-step proof (including the last) have heat=1.

Regarding the need to rely on a formula of length 48 (including predicate symbol), OTTER was used to find a proof in which the most complex formula has length 36. The proof completes with a deduction of *YRO*. If one reduces the max_weight to 32, however, where the weight is computed strictly in terms of symbol count, one finds that no proofs are completed; indeed, the set of support goes empty. Therefore, in the context of another aspect of proof elegance (that concerned with formula complexity), 36 is as small as is possible.

For a quite different example of proof shortening in which the hot list strategy is used, one in which equality plays a key role, I turn to group theory. The two-part problem, Theorem 4 in [McCune96b], was brought to me by McCune, most likely motivated in part by my supplying him with far shorter proofs than he had in hand, proofs he wished to include in his monograph [McCune96a]. Those proofs were obtained with an iterative methodology, not an algorithm, in which the hot list strategy played a key role; see [Wos97a]. The objective of Theorem 4 is to show that a given implication is a single axiom by proving the well-known Higman-Neumann single equational axiom for groups that are defined in terms of division. In the first half, one is asked to show that x/x is independent of x for all x, thus establishing the existence of an identity e. In the second half, one is asked to prove the Higman-Neumann equation, using $x/x = e$ established in the first half. McCune had used OTTER to prove Theorem 4, but wished simpler proofs. For the first half, he had found a 16-step proof of level 11, and for the second half he found a 20-step proof of level 11. With OTTER and (in part) with the use of the hot list strategy, I returned to him (for the first half of Theorem 4) a proof of length 12 and level 8 and for the second half a proof of length 11 and level 8. For the researcher who might wish to improve on the cited results, the following clauses suffice for the first half.

```
x = x.
x / y != z / u | y / (((z / z) / z) / ((x / x) / x)) = u.
A / A != B / B.
```

For the second half, the following suffice.

```
x = x.
x / y != z / u | y / (((z / z) / z) / ((x / x) / x)) = u.
x / x = e.
(A / ((((A / A) / B) / C) / (((A / A) / A) / C))) != B.
```

4. Challenges and Research Topics

Continuing in the tradition begun at Argonne National Laboratory approximately three decades ago (in the early 1960s), I offer a set of problems for research and for testing new ideas, approaches, and programs. The problems are taken from group theory, Robbins algebra, and various logic calculi. Rather than asking for a proof as is my usual custom—although, in many cases, the problems present a formidable challenge in that regard—in the majority of cases, each problem challenges one to find a shorter proof than I know of, if one exists, or find a proof whose length equals that which I cite. In the more general context of science as a whole, the successful search for shorter proofs can result in the discovery of basic truths that are fundamental to the field under study.

Of a more specific nature, the interest in seeking shorter proofs rests in part with their correlation to elegance, in part with the possible discovery of useful lemmas, and in part with practical considerations. Regarding the last point, one can often extract from a proof the design of a circuit or the synthesis of an algorithm. In each case, typically, the shorter the proof, the more appealing is the extracted object—for example, for circuit design, fewer gates, and for algorithm synthesis, fewer instructions. (Other practical considerations are of course pertinent—for example, the cost of executing a given instruction.)

For each of the various challenge problems I offer, I give the smallest length of the proofs with which I am familiar; for some, I give the level of the proof. Length is defined to be the number of deduced steps in a proof; the level of an input statement is defined to be to be 0, and the level of a deduced conclusion to be one greater than the maximum of the levels of its immediate parents or ancestors. I have made no serious attempt to find proofs of smaller level, but I have in many cases seriously attempted to find the shortest proof possible. The researcher might enjoy seeking proofs of smaller level and might enjoy, even more, seeking proofs of strictly shorter length.

4.1. Logic Calculi Challenges

I begin with logic calculi in which equality is not present, for the absence of equality simplifies various issues such as whether to count demodulation applications in computing proof length. For the problems of this subsection, I require the use of condensed detachment as the inference rule, captured with hyperresolution and the following clause in which ‘‘|’’ denotes **or** and ‘‘-’’ denotes **not**.

-P(i(x,y)) | -P(x) | P(y).

The first area from which problems are taken is two-valued sentential (or propositional) calculus, axiomatizable with various subsets of the following formulas (or theses) expressed in OTTER notation [McCune90,Wos92].

(thesis_1 L1) P(i(i(x,y),i(i(y,z),i(x,z)))).
(thesis_2 L2) P(i(i(n(x),x),x)).
(thesis_3 L3) P(i(x,i(n(x),y))).
(thesis_18) P(i(x,i(y,x))).
(thesis_19) P(i(i(i(x,y),z),i(y,z))).
(thesis_21) P(i(i(x,i(y,z)),i(y,i(x,z)))).
(thesis_22) P(i(i(x,y),i(i(z,x),i(z,y)))).
(thesis_30) P(i(i(x,i(x,y)),i(x,y))).
(thesis_35) P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).
(thesis_37) P(i(i(i(x,y),z),i(n(x),z))).
(thesis_39) P(i(n(n(x)),x)).
(thesis_40) P(i(x,n(n(x)))).
(thesis_46) P(i(i(x,y),i(n(y),n(x)))).
(thesis_49) P(i(i(n(x),n(y)),i(y,x))).
(thesis_54) P(i(i(x,y),i(i(n(x),y),y))).
(thesis_59) P(i(i(n(x),y),i(i(z,y),i(i(x,z),y)))).
(thesis_60) P(i(i(x,i(n(y),z)),i(x,i(i(u,z),i(i(y,u),z))))).

Each of the problems I suggest for an attempt to find a shorter proof focuses on the Lukasiewicz axiom system consisting of theses 1, 2, and 3. For similar problems, one can focus instead on the alternative Lukasiewicz axiom system consisting of theses 19, 37, and 59; on the Hilbert system consisting of theses 3, 18, 21, 22, 30, and 54 (of which thesis 30 is dependent); on the Church system consisting of theses 18, 35, and 49; on the Frege system consisting of theses 18, 21, 35, 39, 40, and 46 (of which thesis 21 is dependent); and on the system of Wos consisting of theses 19, 37, and 60.

Problem TV1 asks for a proof (if one exists) with strictly fewer than 21 applications of condensed detachment establishing that the Lukasiewicz axiom system (consisting of theses 1, 2, and 3) implies the Church axiom system. I know of four *short* proofs (Proof 5 through Proof 8 in my new book) of this theorem, each of length 21 and, respectively, of levels 15, 15, 16, and 16. I give as an example the following level-15 proof (proof 5), which is the first I found in my experiments. (If order_history is set as an instruction for OTTER, then [hyper,*i*,*j*,*k*] denotes that *i* is the nucleus, *j* is the major premiss unified with the first negative literal, and *k* is the minor premiss unified with the second negative literal. Also, the number in the first field gives the corresponding position among the retained clauses. The presence of duplicate input clauses in a proof denotes the use of the hot list strategy, the second copy being a member of the input hot list.)

Length 21 Proof of the Church Axiom System for Two-Valued Sentential Calculus

```
-----> EMPTY CLAUSE at   4.60 sec ----> 782 [hyper,4,260,628,199]
$ANSWER(step_allBEH_Church_FL_18_35_49).

Length of proof is 21.  Level of proof is 15.

---------- PROOF ----------

1 [] -P(i(x,y)) | -P(x) | P(y).
4 [] -P(i(p,i(q,p))) | -P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) | -P(i(i(n(p),n(q)),i(q,p))) |
      $ANSWER(step_allBEH_Church_FL_18_35_49).
10 [] P(i(i(x,y),i(i(y,z),i(x,z)))).
11 [] P(i(i(n(x),x),x)).
12 [] P(i(x,i(n(x),y))).
32 [] -P(i(x,y)) | -P(x) | P(y).
33 [] P(i(i(x,y),i(i(y,z),i(x,z)))).
34 [] P(i(i(n(x),x),x)).
35 [] P(i(x,i(n(x),y))).
-----------------------------------
36 [hyper,1,10,10] P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))).
49 [hyper,1,10,12] P(i(i(i(n(x),y),z),i(x,z))).
50 [hyper,1,12,11] P(i(n(i(i(n(x),x),x)),y)).
60 (heat=1) [hyper,32,33,50] P(i(i(x,y),i(n(i(i(n(z),z),z)),y))).
63 [hyper,1,36,36] P(i(i(x,i(y,z)),i(i(u,y),i(x,i(u,z))))).
67 (heat=1) [hyper,32,63,34] P(i(i(x,y),i(i(n(i(y,z)),i(y,z)),i(x,z)))).
88 [hyper,1,49,60] P(i(x,i(n(i(i(n(y),y),y)),z))).
124 [hyper,1,63,67] P(i(i(x,i(n(i(y,z)),i(y,z))),i(i(u,y),i(x,i(u,z))))).
133 [hyper,1,124,88] P(i(i(x,i(n(y),y)),i(z,i(x,y)))).
144 [hyper,1,36,133] P(i(i(n(x),y),i(z,i(i(y,x),x)))).
188 [hyper,1,124,144] P(i(i(x,i(y,z)),i(i(n(z),y),i(x,z)))).
199 (heat=1) [hyper,32,188,35] P(i(i(n(x),n(y)),i(y,x))).
233 [hyper,1,63,188] P(i(i(x,i(n(y),z)),i(i(u,i(z,y)),i(x,i(u,y))))).
260 [hyper,1,49,199] P(i(x,i(y,x))).
287 [hyper,1,36,233] P(i(i(n(x),y),i(i(z,i(u,x)),i(i(y,u),i(z,x))))).
312 [hyper,1,233,260] P(i(i(x,i(y,z)),i(y,i(x,z)))).
331 (heat=1) [hyper,32,312,35] P(i(n(x),i(x,y))).
445 [hyper,1,188,331] P(i(i(n(x),y),i(n(y),x))).
517 [hyper,1,445,331] P(i(n(i(x,y)),x)).
613 [hyper,1,287,517] P(i(i(x,i(y,i(z,u))),i(i(z,y),i(x,i(z,u))))).
628 (heat=1) [hyper,32,613,33] P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).
782 [hyper,4,260,628,199] $ANSWER(step_allBEH_Church_FL_18_35_49).
```

The given proof has the added satisfying feature that it is *compact*, meaning that it is a proof of exactly one of the members of the target axiom set, namely, thesis 35. (The notion of a compact proof was introduced in [Wos96a]; see especially Section 13.1.) Ignoring the concern for proof length, if one wishes a difficult theorem to prove (without guidance from the researcher), where the object is the deduction of a single thesis, I suggest attempting to prove thesis 35 from theses 1, 2, and 3.

Problem TV2 asks for a proof (if one exists) of the alternative Lukasiewicz axiom system with strictly fewer than 24 steps; the 24-step proof has level 16, and it is given in the Appendix.

Problem TV3 asks for a proof (if one exists) of the Hilbert axiom system (including the dependent member, thesis 30) with strictly fewer than 23 steps; the 23-step proof has level 15.

Problem TV4 asks for a proof (if one exists) of the Frege axiom system (including the dependent member, thesis 21) with strictly fewer than 28 steps; the 28-step proof has level 17, and it is given in

the Appendix.

Problem TV5 asks for a proof (if one exists) of the Wos axiom system with strictly fewer than 25 steps; the 25-step proof has level 17, and it is given in the Appendix.

Problem TV6 asks one to obtain a proof with an automated reasoning program, unguided by the researcher, that Meredith's axiom (the following) provides an axiomatization of two-valued sentential calculus.

```
% Following is Meredith's single axiom.
P(i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x)))).  % CN-CAM
```

The next area from which problems are taken is many-valued sentential calculus, a field that is weaker than is two-valued sentential calculus in that the axioms of the former (the following) are implied by those of the latter.

(1) P(i(x,i(y,x))).
(2) P(i(i(x,y),i(i(y,z),i(x,z)))).
(3) P(i(i(i(x,y),y),i(i(y,x),x))).
(4) P(i(i(n(x),n(y)),i(y,x))).

The following four lemmas are provable in many-valued sentential calculus as well as is Theorem_5.

(Lemma_24) P(i(n(n(x)),x)).
(Lemma_29) P(i(x,n(n(x)))).
(Lemma_25) P(i(i(x,y),i(i(z,x),i(z,y)))).
(Lemma_36) P(i(i(x,y),i(n(y),n(x)))).
(Theorem_5) P(i(i(i(x,y),i(y,x)),i(y,x))).

Problem MV1 asks for a proof (if one exists) of Lemma_24 with strictly fewer than 7 steps; the 7-step proof has level 5.

Problem MV2 asks for a proof (if one exists) of Lemma_29 with strictly fewer than 8 steps; the 8-step proof has level 6.

Problem MV3 asks for a proof (if one exists) of Lemma_25 with strictly fewer than 5 steps; the 5-step proof has level 3.

Problem MV4 asks for a proof (if one exists) of Lemma_36 with strictly fewer than 19 steps; the 19-step proof has level 9.

Problem MV5 asks for a proof (if one exists) of Theorem_5 with strictly fewer than 34 steps; the 34-step proof has level 17. Ignoring the concern for proof length, without guidance from the researcher, Lemma_36 is the most difficult to prove when compared with the other three lemmas. Immensely more difficult to prove than Lemma_36 is Theorem_5; indeed, its proof eluded McCune and me for two and one-half years, until he applied his tail strategy and I formulated the recursive tail strategy.

For the final field of logic, I turn to equivalential calculus, still studied with condensed detachment, but with the function *i* replaced with the function *e*. Of the thirteen shortest single axioms, I focus on the two known as *XHK* and *XHN*. To have one's reasoning program prove (without substantial guidance) either formula to be a single axiom for this calculus can offer an impressive challenge indeed. The formulas that are needed for the two problems I offer are the following.

```
P(e(x,e(e(y,z),e(e(x,z),y)))).  % XHK
P(e(x,e(e(y,z),e(e(z,x),y)))).  % XHN
P(e(e(x,y),e(e(z,y),e(x,z)))).  % P1_YQL
P(e(e(e(x,y),z),e(y,e(z,x)))).  % P4_UM
```

Problem EC1 asks for a proof (if one exists) that the formula *XHK* implies the formula *YQL* with strictly fewer than 23 steps; the 23-step proof has level 19.

Problem EC2 asks for a proof (if one exists) that the formula *XHN* implies the formula *UM* with strictly fewer than 19 steps; the 20-step proof has level 14.

4.2. Group Theory Challenges

In this and the next subsection, equality plays the key role, with both paramodulation and demodulation emphasized. Not counted in the length of a proof are the (usually) large number of intermediate steps resulting from demodulation. The results in this and the next subsection were obtained with OTTER applying a Knuth-Bendix approach. Each problem I offer in this subsection focuses on some *variety* of groups, the set of all groups that satisfy the usual axioms for a group and also satisfy one or more additional equations. Specifically, I focus on groups of exponent 7—those in which the seventh power of *x* is the identity *e* for all elements *x*—9, 11, 13, 15, and 17. For each problem, I give a single equation that is in fact a single axiom for the variety of exponent *n* under study, using *f* to denote product and *e* to denote the identity. In each case, the cited lengths are those of the proofs OTTER completed. I give no level for problems in this and the next subsection because, when these problems were run with an earlier version of OTTER, the CPU cost of computing that value was too high when Knuth-Bendix was used.

Problem GT1 asks for proofs (if they exist) that the equation

(f(x,f(x,f(x,f(f(x,f(x,f(x,f(f(x,y),z)))),f(e,f(z,f(z,f(z,f(z,f(z,f(z,z))))))))) = y)

implies left and right identity, exponent 7, and associativity with, respectively, strictly fewer than 73, 38, 41, and 74 steps.

Problem GT2 asks for proofs (if they exist) that the equation

(f(x,f(x,f(x,f(x,f(f(x,f(x,f(x,f(x,f(f(x,y),z))))),f(e,f(z,f(z,f(z,f(z,f(z,f(z,f(z,z))))))))))) = y)

implies left and right identity, exponent 9, and associativity with, respectively, strictly fewer than 74, 36, 37, and 75 steps.

Problem GT3 asks for proofs (if they exist) that the equation

(f(x,f(x,f(x,f(x,f(x,f(f(x,f(x,f(x,f(x,f(x,f(f(x,y),z))))))),
f(e,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,z))))))))))) = y)

implies left and right identity, exponent 11, and associativity with, respectively, strictly fewer than 90, 78, 78, and 91 steps.

Problem GT4 asks for proofs (if they exist) that the equation

(f(x,f(x,f(x,f(x,f(x,f(x,f(f(x,f(x,f(x,f(x,f(x,f(x,f(f(x,y),z)))))))),
f(e,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,z))))))))))))))) = y)

implies left and right identity, exponent 13, and associativity with, respectively, strictly fewer than 176, 54, 56, and 178 steps.

Problem GT5 asks for proofs (if they exist) that the equation

(f(x,f(x,f(x,f(x,f(x,f(x,f(x,f(f(x,f(x,f(x,f(x,f(x,f(x,f(x,f(f(x,y),z)))))))),
f(e,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,z))))))))))))))))) = y)

implies left and right identity, exponent 15, and associativity with, respectively, strictly fewer than 157, 80, 82, and 158 steps.

Problem GT6 asks for proofs (if they exist) that the equation

(f(x,f(x,f(x,f(x,f(x,f(x,f(x,f(x,f(f(x,f(x,f(x,f(x,f(x,f(x,f(x,f(x,f(f(x,y),z)))))))))),
f(e,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,f(z,z))))))))))))))))))) = y)

implies left and right identity, exponent 17, and associativity with, respectively, strictly fewer than 90, 54, 57, and 92 steps.

In each of my studies of groups of odd exponent, OTTER proves in order right identity, the appropriate exponent property, left identity, and associativity. As a measure of the difficulty of getting *any* proof (ignoring the search for a shorter proof), the respective times on a SPARCstation-2 or its equivalent to prove associativity for exponent 7, 9, 11, 13, 15, and 17 were approximately 134, 602, 1711, 7740, 14310, and 62410 CPU-seconds.

4.3. Robbins Algebra Challenges

Whether the axioms (given in yet another notation acceptable to OTTER) of a Robbins algebra imply those of a Boolean algebra was open until McCune answered the question in the affirmative by using his program EQP, which relies on AC-unification [McCune97]. Therefore, for one of the more significant challenges, one might attack the cited question with the objective of finding a proof (with an auutomated reasoning program) *without* guidance from the researcher and *without* the use of AC-unification. (When McCune produced his startling result with EQP, he did not provide guidance, other than choosing the strategies that were used.) Huntington's axiom together with commutativity and associativity of + suffice to characterize Boolean algebra. One can interpret the function ''+'' as addition (or, more familiarly in the context of Boolean algebra, as union) and the function *n* as negation or complement.

```
EQ(+(x,y),+(y,x)).
EQ(+(+(x,y),z),+(x,+(y,z))).
EQ(n(+(n(+(x,y)),n(+(x,n(y))))),x).    % Robbins axiom
-EQ(+(n(+(A,n(B))),n(+(n(A),n(B)))),B).  % denial of Huntington axiom
```

As discussed in Section 3.3, if one adds any one of a number of well-known properties of a Boolean algebra to those of a Robbins algebra, with the aid of a reasoning program, one can prove that the amended algebra is Boolean. For example, adding the property that there exists an element *c* with $c+c = c$ suffices, and adding the property that there exist elements *c* and *d* with $c+d = d$ suffices.

Problem RA1 asks one to find a proof (if one exists) with strictly fewer than 29 steps of the theorem asserting that the addition of $c+c = c$ (for a constant *c*) to the axioms of a Robbins algebra yields Boolean, using Knuth-Bendix but *not* using AC-unification.

Problem RA2 asks one to find a ''short'' proof (if one exists) of the theorem asserting that the addition of $c+d = d$ (for constants *c* and *d*) to the axioms of a Robbins algebra yields Boolean, using Knuth-Bendix but *not* using AC-unification. As noted in Section 3.3, the first proof I found (without AC-unification) has length 78.

Problem RA3 asks one to use an automated reasoning program, not relying on AC-unification, to show that the adjunction (to the three axioms for a Robbins algebra) of the equation $n(n(c)) = c$ for a constant *c* implies Boolean. Indeed, the theorem itself is of course true, as established by McCune [McCune97].

4.4. Global Challenges

Here I briefly suggest general or global topics for study.

Because of McCune's use of discrimination trees and other implementation decisions, the program's drawing of conclusions per CPU-second remains a constant (relative speaking), even for long runs. I call this property *local linearity*. In contrast, the number of clauses chosen as the focus of attention (to initiate applications of inference rules) can deteriorate sharply as the run progresses. I say that the program lacks *global linearity*. The topic for research asks for a means for giving an automated reasoning program, such as OTTER, global linearity.

Of a totally different nature, I suggest for research a study of using the members of the hot list with *generated* clauses, rather than with *retained* clauses. This incarnation or modification of the hot list strategy would permit the program to draw conclusions that are children of clauses that might be discarded because of being too complex as measured in weight.

Also of a strategic nature, I suggest for research the strategy that, once a clause is to be retained, immediately moves it from the set of support to list(usable) based on user-supplied criteria. Intuitively, the criteria should reflect the user's conjecture that the clause(s) to be immediately moved play the role of inference completion rather than of inference initiation. Often, a clause has such heavy weight (substantial complexity defined either by the included weight templates or by its symbol count) that the clause remains in the set of support for far too long or, perhaps, forever. Many such clauses are valuable because the desired proof depends on their use to complete the application of, say, hyperresolution and not to initiate its application. Related is the class of clauses that are chosen as the focus of

attention and, unfortunately, spawn a large number of uninteresting clauses and are better used to complete applications of inference rules. The power of an automated reasoning program might be materially enhanced by immediately moving members of this class to list(usable) and *never* allowing them to be the focus of attention to drive the program's reasoning, but still permitting them to participate in the reasoning in the role of inference completion.

For an involved and multifaceted impressive and significant challenge, I suggest searching for metarules for choosing wisely from among the options offered by OTTER. In my new book, I offer in Chapter 8 the barest beginning. One could minutely reduce the scope of the suggested research by confining the study to finding good metarules for wisely choosing the combination of strategies to employ. Possibly an analysis of the experiments reported in this report would lead to an identification of which properties of an input set of clauses suggest which strategy or strategies clearly merit using. In that regard, the autonomous mode offered by OTTER chooses options based on syntactic properties of the input. I recommend that mode especially for the new user of OTTER.

5. Hints for the Use of OTTER

I have often been encouraged to provide some guidance for the use of the various Argonne automated reasoning programs and—because of its proven power—especially for notions about the use of McCune's OTTER. This section is devoted to such guidance and notions. My suggestions are *not* based on careful scientific analysis but, instead, on my recent experiments and even on my experiments that were conducted during the 1960s. The order of the following hints implies nothing about their relative significance. I provide merely a taste of my opinions and biases.

- Regarding the hot list strategy, especially in the beginning of one's experiments, I recommend that the heat parameter be assigned the value 1, and I recommend that the (input) hot list consist of the clause or clauses that correspond to the special hypothesis of the theorem under attack. For example, if the theorem concerns groups in which the cube of x is the identity e, then the special hypothesis is the equation $xxx = e$. For a second recommendation for the elements of the hot list, I suggest the elements of the (input) set of support that take the form of positive unit clauses. When studying some logic calculus, for a third recommendation, I suggest the axioms of the theory (if fewer than eight in number) and the nonunit clause (if such is used) corresponding to condensed detachment.

- By placing in the (input) hot list clauses for associativity and commutativity, one can use the hot list in place of a limited form of AC-unification. The greater the value assigned to the heat parameter, the more AC-unification that occurs. However, effectiveness can be severely impaired with associativity in the hot list if the heat parameter is assigned the value 3 or greater.

- Regarding the dynamic hot list strategy, a small value is recommended for the dynamic_heat_weight, enough to permit new clauses to be adjoined to the hot list during the run, but not so big as to cause the hot list to become large. Even a small value can drown the hot list, if the weight_list contains templates that both have smaller values assigned to them and are frequently matched during the run. In particular, one must exercise care when combining the dynamic hot list strategy with the resonance strategy. For example, if one includes resonators corresponding to formulas from equivalential calculus, because many formulas can match a single resonator, havoc may be the result. A clue is provided when one sees that OTTER is spending substantial CPU time on a single clause that is chosen as the focus of attention.

- The inclusion of an input hot list is suggested when one conjectures that certain input clauses have been identified as meriting repeated consideration as hypotheses for drawing conclusions by completing applications of an inference rule. A value of 2 or greater for the heat parameter is suggested when one wishes a recursively heavier emphasis on the members of the hot list. The dynamic hot list strategy is suggested when one suspects that some of the clauses adjoined during the run merit repeated visiting as hypotheses for completing applications of an inference rule.

- When attacking a problem of the form *P* if and only if *Q*, such as occurs when attempting to prove the equivalence of two conditions, I recommend *always* attacking it as two separate problems, one for each implication. Otherwise, in far too many cases (as my experiments have shown), the effectiveness of the automated reasoning program in use will be severely reduced, for the two implications will become intermingled. This observation probably comes as no surprise, especially for the researcher familiar with mathematics and logic, for in each of those fields the argument usually is separated into its two components. (For the curious, I am in no way suggesting that imitation of a person's reasoning merits consideration for the basis of an automated reasoning program; quite the contrary.)

- For a first cut at attacking a problem in which one is given the entire underlying theory, when the problem appears to be concerned with equality, I recommend commenting out all nonunit clauses and all unit clauses about concepts other than those that are obviously germane. In other words, one should attempt to solve the problem by including unit clauses only such that the equality predicate is present and such that they focus on concepts directly germane to the objective.

- A good choice for the members of the input set of support consists of the clauses corresponding to the special hypothesis and those corresponding to the negation (or denial) of the conclusion of the theorem under study. When paramodulation is in use, sometimes it is necessary to include in the input set of support clauses corresponding to some interesting law of the underlying theory, for example, the clauses for distributivity in ring theory.

- When attempting to prove a theorem, especially when one is encountering the underlying theory for the first time, the inclusion of resonators that correspond to the proof steps of a related theorem can spell the difference between failure and quick success.

- An effective way to monitor OTTER's progress is to include in list(passive) unit clauses that correspond to the negation (or denial) of steps of the expected proof and unit clauses that correspond to negations (or denials) of lemmas that one suspects are not too difficult to prove.

- Combining a level saturation search with the hot list strategy often produces impressive results. For a taste, when the program is adjoining clauses at level 4, with the hot list strategy in use and the heat parameter assigned the value 2, the program also is deducing (for possible retention) clauses at level 6. In the obvious sense, the cited combination permits the program to look ahead, and the distance is greater than or equal to the value assigned to the heat parameter. For example, although the heat parameter was assigned the value 1 in my study of *XHN*, a proof of level 14 was completed as the program was deducing clauses of level 11; see Section 3.4.

- For the new user of OTTER, I recommend the use of the autonomous mode. That mode enables one to get started *without* making choices from among the various options.

- Regarding the clauses chosen as the focus of attention to drive OTTER's reasoning, I note that the printed weight of each in the output file is the weight of the clause used for clause choosing, and not for clause purging. In technical terms, the printed weight is the pick_given weight of the clause, not the purge_gen weight. Of course, the weight is first assigned (if possible) based on the included weight templates; if none apply, then it is assigned based on symbol count. OTTER can spend a huge amount of CPU time on a single clause chosen as the focus of attention; see the research topic on global linearity in Section 4. The explanation almost always rests with the corresponding large size of list(usable), those clauses placed in that list from the input and those moved to that list and then chosen as the focus of attention. However, the phenomenon can also occur when list(hot) becomes rather large—as can occur, for example, when combining the resonance strategy with the dynamic hot list strategy—especially if the heat parameter is assigned the value 3 or greater. Other related unexpected phenomena can occur when the max_weight is

raised or lowered during a run.

- Level saturation with equality-oriented approaches, such as a Knuth-Bendix approach, in so many cases drowns the program and fails. The explanation rests at least in part with the fact that a rule such as paramodulation is term oriented, not literal oriented, resulting in far more fecundity.

- When using the tail strategy, so that one can maintain a reasonable max_weight, I suggest replacing weight_list(pick_and_purge) with weight_list(purge_gen).

6. Summary and Conclusions

In this report, I have featured the *hot list strategy* and the *dynamic hot list strategy*, presenting numerous pertinent experiments. The hot list strategy asks the researcher to provide an input list (called the hot list) of statements (clauses) that the automated reasoning program uses to complete, in contrast to initiate, applications of the inference rules in use. Ordinarily, one chooses for the members of the hot list clauses that are conjectured to merit revisiting repeatedly, clauses on which to key the program's attack. The dynamic hot list strategy (formulated by McCune) extends the hot list strategy to permit the program to adjoin members to the hot list during the run.

The impetus for my formulation of the hot list strategy was the intention of addressing the following frequently encountered obstacle. A retained clause that is needed for assignment completion remains inaccessible for far too long, measured in CPU time. The source of the inaccessibility is the complexity of the clause, especially encountered when the mechanism for choosing clauses on which to focus rests with their complexity. Using clause complexity as the basis for where next to focus the program's attention (*weighting*) is one of the more effective strategies for directing an attack.

The experiments are taken from lattice ordered groups, Robbins algebra, and various logic calculi. The experiments are accompanied by commentary intended to illustrate my approach to research and to provide some guidance for the use of McCune's program OTTER. By presenting evidence pro and con, I intend to stimulate further research, part of which might focus on analyzing the results reported here. I am certain that such an analysis offers a monumental challenge, at least regarding some aspects. To offer additional specific challenges, I have included Section 4. To aid the researcher in the use of OTTER, I offer some hints in Section 5. The Appendix provides appropriate input files and proofs.

Note that the experimental results vary widely when one fixes a combination of strategies. Rather than discouraging, such is to be expected in view of the nature of mathematics and logic. Indeed, the fault does *not lie* with the automation of reasoning. Precisely because of the complexity of mathematics and logic, access to diverse strategies is required, strategies to be used singly and, far more often, in combination. Regarding strategy at the global level, I note that in OTTER, from the set of support list, a light or less complex clause is often quickly chosen as the focus of attention and used to drive the program's reasoning by initiating applications of inference rules. In contrast, each clause in the hot list, whether input or adjoined during the run, is always and immediately considered with each newly retained clause to complete (rather than initiate) inference rule applications; in particular, members of the hot list do not wait for a newly retained clause to be chosen as the focus of attention before being considered with it. With the hot list, the distinction is that such a clause, if it is in the input hot list or is adjoined to that list through the use of the dynamic hot list strategy, is immediately considered with newly retained clauses that might otherwise wait for a long time or forever to be combined with such a light clause.

I have offered persuasive evidence that the hot list strategy does indeed address a significant obstacle encountered in automated reasoning. More significant, I have presented persuasive evidence that OTTER provides the researcher with a powerful assistant. I conjecture that a most impressive colleague would be provided by OTTER connected to 100 workstations such that one could issue a *grand* command that would cause the program to execute 100 different attacks on the question or problem under study, the difference resting with the choice of strategies being used.

Appendix

As promised, here I present input files and proofs. The input files are intended to facilitate the further study by researchers of the areas touched on in this report. They also are intended to serve as templates for research in other areas of mathematics and logic. In some cases, I include lines preceded with ‘‘%’’, which McCune’s program OTTER treats as a comment. The input files, as well as the proofs given here, provide the merest taste of what one can do with OTTER; more is found in my new book.

One of my main reasons for including specific proofs is my strong conviction that likelihood of experimentation producing valuable results is sharply increased. Indeed, when the objective is the formulation of, say, a new strategy or new inference rule or the testing of a reasoning program, I have always been more than puzzled at the nonchalance of some regarding the value of having in hand a proof of the theorem under attack. Few (if any) means are better for measuring progress than seeing how many proof steps of a given proof have been produced with the new approach or the program under evaluation.

I begin with an input file for proving *LOGT*1, that corresponding to Experiment 7. Through appropriate modification, one can use the file to study lattice ordered groups. The 32-step proof I give immediately after the file is obtained by assigning the heat parameter the value 1 (rather than 2) and by commenting out in the hot list the clauses for commutativity of union and of intersection and those for inverse. I then give McCune’s 33-step proof of *LOGT*1. Then, I give the shortest proof of which I know for *LOGT*2.

Input File for Studying Lattice Ordered Groups

```
set(knuth_bendix).
lex([1,a,u(_,_),n(_,_),*(_,_),i(_),pp(_),np(_)]).
assign(max_weight, 15).
assign(max_proofs, 36).
assign(max_mem, 40000).
assign(pick_given_ratio, 6).
assign(heat, 2).
assign(report, 900).
% set(really_delete_clauses).
clear(print_kept).
clear(print_new_demod).
clear(print_back_demod).

list(usable).
x = x.
(x*y)*z = x* (y*z).
1*x = x.
x*1 = x.
i(x)*x = 1.
x*i(x) = 1.
i(1) = 1.
i(i(x)) = x.
i(x*y) = i(y)*i(x).
n(x,x) = x.
u(x,x) = x.
n(x,y) = n(y,x).
u(x,y) = u(y,x).
n(x,n(y,z)) = n(n(x,y),z).
u(x,u(y,z)) = u(u(x,y),z).
end_of_list.
```

```
list(sos).
u(n(x,y),y) = y.
n(u(x,y),y) = y.
x*u(y,z) = u(x*y,x*z).
x*n(y,z) = n(x*y,x*z).
u(y,z)*x = u(y*x,z*x).
n(y,z)*x = n(y*x,z*x).
pp(x) = u(x,1).
np(x) = n(x,1).
pp(a)*np(a) != a.
end_of_list.

list(passive).
pp(a)*np(a) != a | $ANSWER(step_d33).
i(q)*q*r != r | $ANSWER(step_d01).
u(n(q,r),q) != q | $ANSWER(step_d02).
u(q,n(r,q)) != q | $ANSWER(step_d03).
n(u(q,r),q) != q | $ANSWER(step_d04).
u(q,u(r,q)) != u(r,q) | $ANSWER(step_d05).
u(q,n(q,r)) != q | $ANSWER(step_d06).
n(q,n(r,q)) != n(r,q) | $ANSWER(step_d07).
u(q,u(q,r)) != u(q,r) | $ANSWER(step_d08).
u(n(q,r),n(q,n(r,s))) != n(q,r) | $ANSWER(step_d09).
u(i(u(q,r))*q,i(u(q,r))*r) != 1 | $ANSWER(step_d10).
n(i(n(q,r))*q,i(n(q,r))*r) != 1 | $ANSWER(step_d11).
n(u(x,x*x),u(x,1)) = x | $ANSWER(step_d12).
n(u(x,x*x),u(1,x)) = x | $ANSWER(step_d13).
u(i(u(q,r))*q,1) != 1 | $ANSWER(step_d14).
u(i(u(1,q)),1) != 1 | $ANSWER(step_d15).
u(i(u(q,r))*q*s,s) != s | $ANSWER(step_d16).
n(1,i(u(1,q))) != i(u(1,q)) | $ANSWER(step_d17).
n(i(n(q,r))*r,1) != 1 | $ANSWER(step_d18).
n(i(n(q,1)),1) != 1 | $ANSWER(step_d19).
u(1,i(n(q,1))) != i(n(q,1)) | $ANSWER(step_d20).
n(u(1,x),u(x,x*x)) = x | $ANSWER(step_d21).
u(n(q,1),n(q,i(u(1,r)))) != n(q,1) | $ANSWER(step_d22).
u(i(u(q,r)),i(q)) != i(q) | $ANSWER(step_d23).
u(i(u(i(q),r)),q) != q | $ANSWER(step_d24).
u(i(u(q,i(r))),r) != r | $ANSWER(step_d25).
u(q,i(u(r,i(q)))) != q | $ANSWER(step_d26).
n(q,i(u(r,i(q)))) != i(u(r,i(q))) | $ANSWER(step_d27).
u(i(q),i(n(q,r))) != i(n(q,r)) | $ANSWER(step_d28).
u(n(q,1),i(u(1,i(q)))) != n(q,1) | $ANSWER(step_d29).
i(n(q,1)) != u(1,i(q)) | $ANSWER(step_d30).
n(u(q*r,r),u(r,i(q)*r)) != r | $ANSWER(step_d31).
n(u(1,q),u(q,q*q)) != q | $ANSWER(step_d32).
end_of_list.

list(hot).
% (x*y)*z = x* (y*z).
n(x,y) = n(y,x).
u(x,y) = u(y,x).
% n(n(x,y),z) = n(x,n(y,z)).
% u(u(x,y),z) = u(x,u(y,z)).
pp(x) = u(x,1).
```

```
np(x) = n(x,1).
i(x)*x = 1.
x*i(x) = 1.
u(n(x,y),y) = y.
n(u(x,y),y) = y.
x*u(y,z) = u(x*y,x*z).
x*n(y,z) = n(x*y,x*z).
u(y,z)*x = u(y*x,z*x).
n(y,z)*x = n(y*x,z*x).
end_of_list.
```

A 32-Step Proof of *LOGT*1

```
----> UNIT CONFLICT at 3148.67 sec ----> 23567 [binary,23565.1,85.1] $F.
Length of proof is 32.  Level of proof is 13.
---------- PROOF ----------

36 [] u(n(x,y),y)=y.
37 [] n(u(x,y),y)=y.
40 [] u(y,z)*x=u(y*x,z*x).
41 [] n(y,z)*x=n(y*x,z*x).
44,43 [] (x*y)*z=x*y*z.
46,45 [] 1*x=x.
48,47 [] x*1=x.
50,49 [] i(x)*x=1.
51 [] x*i(x)=1.
54,53 [] i(1)=1.
56,55 [] i(i(x))=x.
63 [] n(x,y)=n(y,x).
64 [] u(x,y)=u(y,x).
66,65 [] n(n(x,y),z)=n(x,n(y,z)).
68,67 [] u(u(x,y),z)=u(x,u(y,z)).
70,69 [] u(n(x,y),y)=y.
71 [] n(u(x,y),y)=y.
74,73 [] x*u(y,z)=u(x*y,x*z).
76,75 [] x*n(y,z)=n(x*y,x*z).
78,77 [] u(x,y)*z=u(x*z,y*z).
82,81 [] pp(x)=u(x,1).
84,83 [] np(x)=n(x,1).
------------------------------------
85 [demod,82,84,76,78,46,48] n(u(a*a,a),u(a,1))!=a.
------------------------------------
88 [para_into,69.1.1.1,63.1.1] u(n(x,y),x)=x.
94 (heat=1) [para_into,88.1.1.1,37.1.1] u(x,u(y,x))=u(y,x).
100 [para_from,69.1.1,67.1.1.1] u(n(x,y),u(y,z))=u(y,z).
103,102 (heat=1) [para_into,100.1.1.1,37.1.1] u(x,u(x,y))=u(x,y).
107 [para_into,71.1.1.1,64.1.1] n(u(x,y),x)=x.
111,110 [para_into,71.1.1,63.1.1] n(x,u(y,x))=x.
123 [para_from,88.1.1,67.1.1.1] u(n(x,y),u(x,z))=u(x,z).
141 [para_into,107.1.1,63.1.1] n(x,u(x,y))=x.
152,151 [para_into,73.1.1,49.1.1] u(i(u(x,y))*x,i(u(x,y))*y)=1.
154 (heat=1) [para_from,151.1.1,37.1.1.1] n(1,i(u(x,y))*y)=i(u(x,y))*y.
307,306 [para_from,123.1.1,141.1.1.2,demod,66] n(x,n(y,u(x,z)))=n(x,y).
818 [para_into,151.1.1.1.1.1,102.1.1,demod,103,74,152] u(i(u(x,y))*x,1)=1.
833 (heat=1) [para_from,818.1.1,40.1.1.1,demod,46,44,46] u(i(u(x,y))*x*z,z)=z.
```

866 [para_into,818.1.1.1,47.1.1] u(i(u(1,x)),1)=1.
931 [para_from,866.1.1,306.1.1.2.2] n(i(u(1,x)),n(y,1))=n(i(u(1,x)),y).
957 (heat=1) [para_from,931.1.1,36.1.1.1] u(n(i(u(1,x)),y),n(y,1))=n(y,1).
1413 [para_into,833.1.1.1.2,51.1.1,demod,48] u(i(u(x,y)),i(x))=i(x).
1416 [para_into,833.1.1.1.2,49.1.1,demod,48] u(i(u(i(x),y)),x)=x.
1443 [para_into,1416.1.1.1.1,94.1.1] u(i(u(x,i(y))),y)=y.
1460 [para_into,1416.1.1,64.1.1] u(x,i(u(i(x),y)))=x.
1567 [para_from,1443.1.1,141.1.1.2] n(i(u(x,i(y))),y)=i(u(x,i(y))).
1662,1661 [para_into,1413.1.1.1.1,88.1.1] u(i(x),i(n(x,y)))=i(n(x,y)).
1684 [para_from,1413.1.1,141.1.1.2] n(i(u(x,y)),i(x))=i(u(x,y)).
1705 (heat=1) [para_into,1684.1.1.1.1,36.1.1,demod,70] n(i(x),i(n(y,x)))=i(x).
1910 [para_into,1705.1.1.1,53.1.1,demod,54] n(1,i(n(x,1)))=1.
1933,1932 (heat=1) [para_from,1910.1.1,36.1.1.1] u(1,i(n(x,1)))=i(n(x,1)).
7237,7236 [para_into,957.1.1.1,1567.1.1] u(i(u(1,i(x))),n(x,1))=n(x,1).
7252,7251 [para_from,7236.1.1,1460.1.1.2.1,demod,68,1662,1933] i(n(x,1))=u(1,i(x)).
7302 [para_from,7236.1.1,154.1.1.2.1.1,demod,7252,76,78,46,50,48,307,111,7237,7252,
76,78,46,50,48] n(u(x,1),u(1,i(x)))=1.
7338 (heat=1) [para_from,7302.1.1,41.1.1.1,demod,46,78,46,78,46] n(u(x*y,y),u(y,i(x)*y))=y.
23565 [para_into,7338.1.1.2.2,51.1.1,demod,56,56,56,56] n(u(x*x,x),u(x,1))=x.
23567 [binary,23565.1,85.1] $F.

A 33-Step Proof of *LOGT*1

----> UNIT CONFLICT at 19280.59 sec ----> 34171 [binary,34169.1,1674.1] $F.
Length of proof is 33. Level of proof is 10.
---------- PROOF ----------

3,2 [] (x*y)*z=x*y*z.
5,4 [] 1*x=x.
7,6 [] x*1=x.
8 [] i(x)*x=1.
10 [] x*i(x)=1.
15,14 [] i(i(x))=x.
20 [] u(x,x)=x.
22 [] n(x,y)=n(y,x).
23 [] u(x,y)=u(y,x).
24 [] n(n(x,y),z)=n(x,n(y,z)).
27,26 [] u(u(x,y),z)=u(x,u(y,z)).
28 [] u(n(x,y),y)=y.
30 [] n(u(x,y),y)=y.
33,32 [] x*u(y,z)=u(x*y,x*z).
35,34 [] x*n(y,z)=n(x*y,x*z).
37,36 [] u(x,y)*z=u(x*z,y*z).
39,38 [] n(x,y)*z=n(x*z,y*z).
41,40 [] pp(x)=u(x,1).
43,42 [] np(x)=n(x,1).
---
44 [demod,41,43,35,37,5,7] n(u(a*a,a),u(a,1))!=a.

46,45 [para_from,8.1.1,2.1.1.1,demod,5] i(x)*x*y=y.
60 [para_into,28.1.1.1,22.1.1] u(n(x,y),x)=x.
62 [para_into,28.1.1,23.1.1] u(x,n(y,x))=x.
64 [para_into,30.1.1.1,23.1.1] n(u(x,y),x)=x.
70 [para_into,60.1.1.1,30.1.1] u(x,u(y,x))=u(y,x).
74 [para_into,60.1.1,23.1.1] u(x,n(x,y))=x.

79,78 [para_from,62.1.1,30.1.1.1] n(x,n(y,x))=n(y,x).
88,87 [para_into,26.1.1.1,20.1.1] u(x,u(x,y))=u(x,y).
107 [para_into,74.1.1.2,24.1.1] u(n(x,y),n(x,n(y,z)))=n(x,y).
120,119 [para_into,32.1.1,8.1.1] u(i(u(x,y))*x,i(u(x,y))*y)=1.
134,133 [para_into,34.1.1,8.1.1] n(i(n(x,y))*x,i(n(x,y))*y)=1.
211 [para_into,44.1.1.1,23.1.1] n(u(a,a*a),u(a,1))!=a.
555 [para_into,211.1.1.2,23.1.1] n(u(a,a*a),u(1,a))!=a.
637 [para_into,119.1.1.1.1.1,87.1.1,demod,88,33,120] u(i(u(x,y))*x,1)=1.
671 [para_into,637.1.1.1,6.1.1] u(i(u(1,x)),1)=1.
683 [para_from,637.1.1,36.1.1.1,demod,5,3,5] u(i(u(x,y))*x*z,z)=z.
725 [para_from,671.1.1,64.1.1.1] n(1,i(u(1,x)))=i(u(1,x)).
934 [para_into,133.1.1.1.1.1,78.1.1,demod,79,35,134] n(i(n(x,y))*y,1)=1.
1117 [para_into,934.1.1.1,6.1.1] n(i(n(x,1)),1)=1.
1169,1168 [para_from,1117.1.1,60.1.1.1] u(1,i(n(x,1)))=i(n(x,1)).
1674 [para_into,555.1.1,22.1.1] n(u(1,a),u(a,a*a))!=a.
1783 [para_from,725.1.1,107.1.1.2.2] u(n(x,1),n(x,i(u(1,y))))=n(x,1).
2464 [para_into,683.1.1.1.2,10.1.1,demod,7] u(i(u(x,y)),i(x))=i(x).
2466 [para_into,683.1.1.1.2,8.1.1,demod,7] u(i(u(i(x),y)),x)=x.
2490 [para_into,2466.1.1.1.1,70.1.1] u(i(u(x,i(y))),y)=y.
2567 [para_into,2490.1.1,23.1.1] u(x,i(u(y,i(x))))=x.
2599 [para_from,2490.1.1,64.1.1.1] n(x,i(u(y,i(x))))=i(u(y,i(x))).
2649,2648 [para_into,2464.1.1.1.1,60.1.1] u(i(x),i(n(x,y)))=i(n(x,y)).
19584 [para_into,1783.1.1.2,2599.1.1] u(n(x,1),i(u(1,i(x))))=n(x,1).
19808 [para_from,19584.1.1,2567.1.1.2.1,demod,27,2649,1169] i(n(x,1))=u(1,i(x)).
20017 [para_from,19808.1.1,45.1.1.1,demod,39,5,35,37,5,46,37,5] n(u(x*y,y),u(y,i(x)*y))=y.
34169 [para_into,20017.1.1.1.1,8.1.1,demod,15] n(u(1,x),u(x,x*x))=x.
34171 [binary,34169.1,1674.1] $F.

A Short Proof of *LOGT*2

----> UNIT CONFLICT at 3566.71 sec ----> 19786 [binary,19784.1,1.1] $ANSWER(step_thm).
Length of proof is 22. Level of proof is 10.
---------- PROOF ----------

1 [] i(n(a,b))!=u(i(a),i(b)) | $ANSWER(step_thm).
4 [] i(x)*x=1.
5 [] x*i(x)=1.
10 [] u(n(x,y),y)=y.
11 [] n(u(x,y),y)=y.
13 [] x*y*z= (x*y)*z.
15,14 [copy,13,flip.1] (x*y)*z=x*y*z.
17,16 [] 1*x=x.
19,18 [] x*1=x.
20 [] i(x)*x=1.
34 [] n(x,y)=n(y,x).
35 [] u(x,y)=u(y,x).
39 [] u(x,u(y,z))=u(u(x,y),z).
40 [copy,39,flip.1] u(u(x,y),z)=u(x,u(y,z)).
46 [] u(n(x,y),y)=y.
51,50 [] x*u(y,z)=u(x*y,x*z).
54 [] u(x,y)*z=u(x*z,y*z).
61 [] n(x,u(y,z))=u(n(x,y),n(x,z)).
----
65 [para_into,46.1.1.1,34.1.1] u(n(x,y),x)=x.
70,69 (heat=1) [para_into,65.1.1.1,11.1.1] u(x,u(y,x))=u(y,x).

```
73 [para_from,46.1.1,40.1.1.1,flip.1] u(n(x,y),u(y,z))=u(y,z).
76,75 (heat=1) [para_into,73.1.1.1,11.1.1] u(x,u(x,y))=u(x,y).
151,150 [para_into,50.1.1,20.1.1,flip.1] u(i(u(x,y))*x,i(u(x,y))*y)=1.
164 [para_into,150.1.1.1.1.1,75.1.1,demod,76,51,151] u(i(u(x,y))*x,1)=1.
166 [para_into,150.1.1.1.1.1,69.1.1,demod,70,51,151] u(i(u(x,y))*y,1)=1.
342 [para_into,54.1.1.1,166.1.1,demod,17,15,17,flip.1] u(i(u(x,y))*y*z,z)=z.
344 [para_into,54.1.1.1,164.1.1,demod,17,15,17,flip.1] u(i(u(x,y))*x*z,z)=z.
354 (heat=1) [para_into,342.1.1.1.2,4.1.1,demod,19] u(i(u(x,i(y))),y)=y.
356 (heat=1) [para_into,344.1.1.1.2,5.1.1,demod,19] u(i(u(x,y)),i(x))=i(x).
371,370 (heat=2) [para_into,356.1.1.1.1,10.1.1] u(i(x),i(n(y,x)))=i(n(y,x)).
514 [para_into,344.1.1,35.1.1] u(x,i(u(y,z))*y*x)=x.
518 (heat=1) [para_into,514.1.1.2.2,4.1.1,demod,19] u(x,i(u(i(x),y)))=x.
526 (heat=2) [para_from,518.1.1,11.1.1.1] n(x,i(u(i(x),y)))=i(u(i(x),y)).
599,598 [para_into,356.1.1.1.1,65.1.1] u(i(x),i(n(x,y)))=i(n(x,y)).
1009 [para_from,354.1.1,61.1.1.2,flip.1] u(n(x,i(u(y,i(z)))),n(x,z))=n(x,z).
1033 [para_into,518.1.1,40.1.1] u(x,u(y,i(u(i(u(x,y)),z))))=u(x,y).
13284 [para_into,1009.1.1.1,526.1.1] u(i(u(i(x),i(y))),n(x,y))=n(x,y).
19784 [para_from,13284.1.1,1033.1.1.2.2.1,demod,371,599] i(n(x,y))=u(i(x),i(y)).
19786 [binary,19784.1,1.1] $ANSWER(step_thm).
```

The following input file can be used, with suitable modifications, to study Robbins algebra. It was used to prove *RAT*5. The commented out weight templates illustrate, if comments are removed, how one can used the resonance strategy by keying on the positive proof steps from a related theorem.

Input File for Studying Robbins Algebra

```
set(knuth_bendix).
clear(eq_units_both_ways).
set(index_for_back_demod).
set(dynamic_demod_lex_dep).
% set(lex_rpo).
set(process_input).
% set(display_terms).
set(input_sos_first).
clear(print_kept).
clear(print_new_demod).
clear(print_back_demod).
assign(max_proofs, 2).
assign(report, 1800).
assign(max_weight, 30).
assign(pick_given_ratio, 3).
assign(max_mem, 80000).
% assign(heat,1).
% assign(dynamic_heat_weight, 2).
assign(max_distinct_vars, 3).
lex([a, b, c, d, e, f, g(x), n(x), +(x,x)]).
% lrpo_lr_status([+(x,x)]).

weight_list(pick_given).
%  Following is hypothesis.
% weight(EQ(+(c,d),c),2).
%  Following are positive steps from a 31-step proof of +(c,c)=c, the union of c and c = c,
%  modified to not mention constants.
% weight(EQ(n(+(n(+(x,y)),n(+(y,n(x))))),y),2).
% weight(EQ(n(+(n(+(x,y)),n(+(n(y),x))))),x),2).
```

```
% weight(EQ(n(+(n(+(x,+(y,z))),n(+(z,n(+(x,y))))),z),2).
% weight(EQ(n(+(n(+(x,y)),n(+(n(x),y)))),y),2).
% weight(EQ(n(+(n(+(x,n(y))),n(+(y,x)))),x),2).
% weight(EQ(n(+(n(+(n(x),y)),n(+(y,x)))),y),2).
% weight(EQ(n(+(n(+(x,+(y,z))),n(+(y,n(+(x,z))))),y),2).
% weight(EQ(n(+(n(+(x,+(y,z))),n(+(x,n(+(z,y))))),x),2).
% weight(EQ(n(+(n(+(x,+(y,z))),n(+(n(+(x,z)),y)))),y),2).
% weight(EQ(n(+(n($(1)),n(+($(1),n($(1))))),$(1)),-2).
% weight(EQ(+($(1),+($(1),x)),+($(1),x)),-1).
% weight(EQ(n(+($(1),n(+(n($(1)),+($(1),n($(1)))))),n($(1))),-3).
% weight(EQ(n(+(n(+($(1),x)),n(+(n($(1)),+(x,n(+($(1),n($(1))))))))),x),-2).
% weight(EQ(+($(1),+(x,$(1))),+($(1),x)),-1).
% weight(EQ(n(+(n($(!)),n(+(n($(1)),+($(1),n($(1)))))),$(1)),-3).
% weight(EQ(n(+(n($(1)),+($(1),n($(1))))),n(+($(1),n($(1))))),-3).
% weight(EQ(n(+($(1),n(+($(1),n($(1))))),n($(1))),-2).
% weight(EQ(n(+(n($(1)),n(+(n($(1)),+($(1),n(+($(1),n($(1))))))))),$(1)),-4).
% weight(EQ(n(+(n(+($(1),x)),n(+(n($(1)),+(x,$(1))))),+(x,$(1))),-2).
% weight(EQ(+($(1),+(x,+($(1),y))),+($(1),+(x,y))),-1).
% weight(EQ(+($(1),n(+($(1),n($(1))))),$(1)),-2).
% weight(EQ(+($(1),+(x,n(+($(1),n($(1))))),+($(1),x)),-2).
% weight(EQ(+(x,n(+($(1),n($(1))))),x),0).
% weight(EQ(+(n(+($(1),n($(1)))),x),x),0).
% weight(EQ(n(+(n(x),n(+($(1),+(x,n($(1)))))),x),0).
% weight(EQ(n(+(n(x),n(n(x)))),n(+($(1),n($(1))))),0).
% weight(EQ(n(+(x,n(x))),n(+($(1),n($(1))))),0).
% weight(EQ(n(n(+(x,n(n(x))))),x),2).
% weight(EQ(n(n(x)),x),2).
% weight(EQ(+(n(+(y,x)),n(+(n(y),x))),n(x)),2).
%  Following are to pitch associative variants.
weight(EQ(+(x,+(x,+(x,x))),+(x,+(x,+(x,x)))), 500).
weight(EQ(+(x,+(x,+(x,+(x,x)))),+(x,+(x,+(x,+(x,x))))), 500).
%  Following is for tail strategy.
weight(EQ($(1),$(2)), 1).
%  Following is for discarding triple n.
weight(n(n(n(1))), 500).
end_of_list.

list(usable).
EQ(x,x).
EQ(+(x,y),+(y,x)).
EQ(+(+(x,y),z),+(x,+(y,z))).
end_of_list.

list(sos).
EQ(n(+(n(+(x,y)),n(+(x,n(y))))),x).     % Robbins axiom
EQ(+(c,d),c).   % hypothesis
-EQ(+(n(+(a,n(b))),n(+(n(a),n(b)))),b).  % denial of Huntington axiom
end_of_list.

list(passive).
-EQ(+(x,x),x) | $ANS(step_thm).
-EQ(n(n(n(n(a)+b)+n(a+b)+e)+n(e+b)),e) | $ANS(step_winker01).
-EQ(n(n(n(a+b)+n(a)+b)+b),n(a+b)) | $ANS(step_winker02).
-EQ(n(n(c+c+n(d+n(a))+n(d+a))+n(d+n(c))),c+c) | $ANS(step_winker03).
-EQ(n(n(n(a+b)+n(a)+b+b)+n(a+b)),b) | $ANS(step_winker04).
```

-EQ((n(n(c)+n(d+n(c)))),d) | $ANS(step_m01).
-EQ((n(n(c+a+b)+n(d+n(c+a)+b))),d+b) | $ANS(step_m02).
-EQ((n(d+n(c+n(d+n(c))))),n(d+n(c))) | $ANS(step_m03).
-EQ((n(n(n(n(a)+b)+n(a+b)+e)+n(b+e))),e) | $ANS(step_m04).
-EQ((n(n(n(n(a)+b)+a+b)+b)),n(n(a)+b)) | $ANS(step_m05).
-EQ((n(n(c)+n(d+n(c+n(a))+n(c+a)))),d) | $ANS(step_m06).
-EQ((n(n(d+n(c+n(d+n(c)))+a)+n(n(d+n(c))+a))),a) | $ANS(step_m07).
-EQ((n(n(c+n(d+n(c)))+n(d+n(c)))),d) | $ANS(step_m08).
-EQ((n(n(c+n(c+n(d+n(c))))+n(c+n(d+n(c))))),c) | $ANS(step_m09).
-EQ((n(d+n(d+n(c)+n(c+n(d+n(c))))),n(c+n(d+n(c)))) | $ANS(step_m10).
-EQ((n(d+n(n(c)+n(n(d+n(c))+n(a))+n(n(d+n(c))+a)))),n(c)) | $ANS(step_m11).
-EQ((n(n(n(n(n(a)+b)+a+b)+b+e)+n(n(n(a)+b)+e))),e) | $ANS(step_m12).
-EQ((n(c+n(d+n(c)))),n(c)) | $ANS(step_m13).
-EQ((n(n(c)+n(c+n(c)))),c) | $ANS(step_m14).
-EQ((n(n(c+a)+n(n(c)+n(c+n(c))+a))),a) | $ANS(step_m15).
-EQ((n(n(c+c+n(c+n(c)))+n(c+n(c)))),c) | $ANS(step_m16).
-EQ((n(c+n(c+n(c)+n(c+c+n(c+n(c))))),n(c+c+n(c+n(c)))) | $ANS(step_m17).
-EQ((n(c+c+n(c+n(c)))),n(c)) | $ANS(step_m18).
-EQ((d+n(c+n(c))),d) | $ANS(step_m19).
-EQ((c+n(c+n(c))),c) | $ANS(step_m20).
-EQ((n(c+n(c))+a),a) | $ANS(step_m21).
end_of_list.

% list(demodulators).
% EQ(+(x,y),+(y,x)).
% EQ(+(+(x,y),z),+(x,+(y,z))).
% EQ(n(+(n(+(x,y)),n(+(x,n(y))))),x). % Robbins axiom
% end_of_list.

% list(hot).
% EQ(+(c,d),c). % hypothesis
% % EQ(n(+(n(+(x,y)),n(+(x,n(y))))),x). % Robbins axiom
% % EQ(+(x,y),+(y,x)).
% % EQ(+(+(x,y),z),+(x,+(y,z))).
% end_of_list.

In view of the historical significance of finding a proof of *RAT*5 *without* induction and *without* AC-unification, I include the following proof.

A Historically Significant Proof of *RAT*5

----> UNIT CONFLICT at 9770.64 sec ----> 48310 [binary,48308.1,1.1] $ANS(step_thm).
Length of proof is 78. Level of proof is 16.
---------- PROOF ----------

1 [] -EQ(x+x,x) | $ANS(step_thm).
29,28 [] EQ(x+y,y+x).
31,30 [] EQ((x+y)+z,x+y+z).
32 [] EQ(n(n(x+y)+n(x+n(y))),x).
35,34 [] EQ(c+d,c).
----
37 [para_into,32.1.1.1.1.1,30.1.1,demod,31] EQ(n(n(x+y+z)+n(x+y+n(z))),x+y).
39 [para_into,32.1.1.1.1.1,28.1.1] EQ(n(n(x+y)+n(y+n(x))),y).
41 [para_into,32.1.1.1.1,32.1.1] EQ(n(x+n(n(x+y)+n(n(x+n(y))))),n(x+y)).
43 [para_into,32.1.1.1.2.1.2,32.1.1,demod,29] EQ(n(n(x+y)+n(x+n(y+z)+n(y+n(z)))),x).

45 [para_into,32.1.1.1.2.1,28.1.1] EQ(n(n(x+y)+n(n(y)+x)),x).
52,51 [para_from,34.1.1,30.1.1.1] EQ(c+d+x,c+x).
61 [para_into,37.1.1.1.1.1.2,28.1.1] EQ(n(n(x+y+z)+n(x+z+n(y))),x+z).
63 [para_into,37.1.1.1.1.1,28.1.1,demod,31] EQ(n(n(x+y+z)+n(z+x+n(y))),z+x).
71 [para_from,37.1.1,32.1.1.1.2,demod,29,31] EQ(n(x+y+n(n(x+y+z)+x+y+n(z))),n(x+y+z)).
75 [para_into,51.1.1.2,28.1.1] EQ(c+x+d,c+x).
79 [para_from,51.1.1,32.1.1.1.1.1] EQ(n(n(c+x)+n(c+n(d+x))),c).
81 [para_into,75.1.1.2,30.1.1] EQ(c+x+y+d,c+x+y).
90,89 [para_from,75.1.1,30.1.1.1,demod,31,31] EQ(c+x+d+y,c+x+y).
95 [para_into,39.1.1.1.1.1,34.1.1] EQ(n(n(c)+n(d+n(c))),d).
97 [para_into,39.1.1.1.1.1,30.1.1] EQ(n(n(x+y+z)+n(z+n(x+y))),z).
101 [para_into,39.1.1.1.1,39.1.1] EQ(n(x+n(n(x+n(y))+n(n(y+x)))),n(x+n(y))).
115 [para_into,39.1.1.1.2.1,28.1.1] EQ(n(n(x+y)+n(n(x)+y)),y).
119 [para_into,39.1.1.1,28.1.1] EQ(n(n(x+n(y))+n(y+x)),x).
126,125 [para_from,39.1.1,32.1.1.1.1] EQ(n(x+n(n(y+x)+n(n(x+n(y))))),n(y+x)).
131 [para_into,41.1.1.1.2.1.1.1,28.1.1,demod,126] EQ(n(x+y),n(y+x)).
166,165 [para_into,131.1.1.1,30.1.1] EQ(n(x+y+z),n(z+x+y)).
173 [para_from,131.1.1,39.1.1.1.2.1.2,demod,31] EQ(n(n(x+y+z)+n(z+n(y+x))),z).
195 [para_into,45.1.1.1.2.1.1,39.1.1] EQ(n(n(x+n(y+z)+n(z+n(y)))+n(z+x)),x).
199 [para_into,45.1.1.1.2,95.1.1,demod,29,29] EQ(n(d+n(c+n(d+n(c)))),n(d+n(c))).
203 [para_into,45.1.1.1.2,45.1.1,demod,166,29,29] EQ(n(x+n(y+x+n(n(y)+x))),n(n(y)+x)).
209 [para_into,45.1.1.1,28.1.1] EQ(n(n(n(x)+y)+n(y+x)),y).
229 [para_into,43.1.1.1.1,95.1.1] EQ(n(d+n(n(c)+n(n(d+n(c))+x)+n(n(d+n(c))+n(x)))),n(c)).
267 [para_into,115.1.1.1.1.1,30.1.1] EQ(n(n(x+y+z)+n(n(x+y)+z)),z).
325 [para_into,119.1.1.1.1.1,30.1.1] EQ(n(n(x+y+n(z))+n(z+x+y)),x+y).
329 [para_from,119.1.1,43.1.1.1.2.1.2.2,demod,166,29,29] EQ(n(n(x+n(y+n(z)))+
n(x+y+n(y+z+n(y+n(z))))),x).
361 [para_into,209.1.1.1.2.1,30.1.1] EQ(n(n(n(x)+y+z)+n(y+z+x)),y+z).
411 [para_into,79.1.1.1.1.1,28.1.1] EQ(n(n(x+c)+n(c+n(d+x))),c).
551 [para_into,411.1.1.1,28.1.1] EQ(n(n(c+n(d+x))+n(x+c)),c).
739 [para_into,81.1.1.2,28.1.1,demod,31,90] EQ(c+x+y,c+y+x).
755 [para_into,739.1.1.2,739.1.1,demod,29] EQ(c+c+x+y,c+c+y+x).
862,861 [para_from,61.1.1,32.1.1.1.2,demod,29,31] EQ(n(x+y+n(n(x+z+y)+x+y+n(z))),n(x+z+y)).
889 [para_into,97.1.1.1.1.1.2,75.1.1,demod,31] EQ(n(n(x+c+y)+n(y+d+n(x+c))),y+d).
903 [para_into,97.1.1.1.1.1,28.1.1,demod,31] EQ(n(n(x+y+z)+n(y+n(z+x))),y).
1000,999 [para_into,63.1.1.1.1.1.2,739.1.1,demod,31,31] EQ(n(n(x+c+y+z)+n(z+y+x+n(c))),z+y+x).
1047 [para_into,63.1.1.1,28.1.1] EQ(n(n(x+y+n(z))+n(y+z+x)),x+y).
1829 [para_into,71.1.1.1.2.2.1.1.1.2,28.1.1,demod,862] EQ(n(x+y+z),n(x+z+y)).
1997 [para_into,173.1.1.1.2.1.2.1,75.1.1,demod,31] EQ(n(n(x+d+c+y)+n(y+n(c+x))),y).
2087,2086 [para_from,173.1.1,119.1.1.1.2,demod,29] EQ(n(x+n(n(x+n(y+z))+n(n(z+y+x)))),n(x+n(y+z))).
2410 [para_into,267.1.1.1.1.1.2,28.1.1] EQ(n(n(x+y+z)+n(n(x+z)+y)),y).
2416 [para_into,267.1.1.1.1.1,28.1.1,demod,31] EQ(n(n(x+y+z)+n(n(z+x)+y)),y).
3178 [para_into,903.1.1.1.2.1.2.1,51.1.1,demod,31] EQ(n(n(d+x+y+c)+n(y+n(c+x))),y).
3294 [para_into,101.1.1.1.2.1.1.1.2,131.1.1,demod,31,2087] EQ(n(x+n(y+z)),n(x+n(z+y))).
4032 [para_into,2410.1.1.1.2.1.1,32.1.1,demod,29] EQ(n(n(x+y)+n(n(x+z)+y+n(x+n(z)))),y).
4064 [para_into,2416.1.1.1.2.1.1.1,75.1.1,demod,31] EQ(n(n(x+d+y+c)+n(n(c+x)+y)),y).
4887,4886 [para_into,165.1.1.1.2,28.1.1] EQ(n(x+y+z),n(y+x+z)).
5674 [para_into,1829.1.1.1,28.1.1,demod,31] EQ(n(x+y+z),n(z+y+x)).
7392 [para_into,1997.1.1.1.2.1.2.1,34.1.1] EQ(n(n(d+d+c+x)+n(x+n(c))),x).
7562 [para_into,7392.1.1.1.1.1.2.2,28.1.1] EQ(n(n(d+d+x+c)+n(x+n(c))),x).
7710 [para_into,7562.1.1.1,28.1.1] EQ(n(n(x+n(c))+n(d+d+x+c)),x).
9589,9588 [para_into,195.1.1.1.2.1,34.1.1,demod,29] EQ(n(n(c)+n(d+n(x+c)+n(c+n(x)))),d).
9834 [para_into,4064.1.1.1.1.1,28.1.1,demod,31,31] EQ(n(n(d+x+c+y)+n(n(c+y)+x)),x).
9998 [para_from,199.1.1,3178.1.1.1.2,demod,29,35,29,4887,52,29] EQ(n(n(d+n(c))+n(c+n(d+n(c)))),d).
10020 [para_from,199.1.1,551.1.1.1.1.1.2,demod,29] EQ(n(n(c+n(d+n(c)))+n(c+n(c+n(d+n(c))))),c).

10049,10048 [para_from,9998.1.1,2416.1.1.1.2,demod,29,4887,29] EQ(n(d+n(d+n(c)+n(c+n(d+n(c))))),
n(c+n(d+n(c)))).
12883,12882 [para_into,229.1.1.1.2.1.2.2,7710.1.1,demod,29,35,29,35,29,35,29,29,4887,10049]
EQ(n(c+n(d+n(c))),n(c)).
12889,12888 [back_demod,10020,demod,12883,12883] EQ(n(n(c)+n(c+n(c))),c).
12954 [para_from,12888.1.1,203.1.1.1.2.1.2.2,demod,29,12889] EQ(n(n(c+n(c))+n(c+c+n(c+n(c)))),c).
16947,16946 [para_from,12954.1.1,9834.1.1.1.2,demod,29,31,4887,52,29] EQ(n(c+n(c+n(c)+
n(c+c+n(c+n(c))))),n(c+c+n(c+n(c)))).
19738 [para_into,325.1.1.1.1,5674.1.1] EQ(n(n(n(x)+y+z)+n(x+z+y)),z+y).
19912 [para_into,329.1.1.1.1,12888.1.1,demod,4887,16947] EQ(n(c+c+n(c+n(c))),n(c)).
22688 [para_into,361.1.1.1.2.1,28.1.1,demod,31] EQ(n(n(n(x)+y+z)+n(z+x+y)),y+z).
23276 [para_from,755.1.1,63.1.1.1.1.1,demod,31,1000,29] EQ(x+y+c,c+x+y).
24330 [para_from,23276.1.1,30.1.1.1,demod,31,31,31] EQ(c+x+y+z,x+y+c+z).
24334 [para_from,23276.1.1,28.1.1,demod,31] EQ(c+x+y,y+c+x).
24970 [para_into,24334.1.1.2,28.1.1] EQ(c+x+y,x+c+y).
30025,30024 [para_from,3294.1.1,131.1.1] EQ(n(x+n(y+z)),n(n(z+y)+x)).
41187,41186 [para_into,22688.1.1.1.2.1.2.24970.1.1,demod,31,31]
EQ(n(n(n(c)+x+y+z)+n(z+x+c+y)),x+y+z).
43403,43402 [para_from,24330.1.1,19738.1.1.1.2.1,demod,31,41187] EQ(x+y+z,z+x+y).
43621,43620 [para_into,43402.1.1.2,28.1.1] EQ(x+y+z,y+x+z).
43961,43960 [para_into,43620.1.1,43402.1.1] EQ(x+y+z,z+y+x).
48174 [para_into,889.1.1.1.1,19912.1.1,demod,43961,43621,9589,29] EQ(d+n(c+n(c)),d).
48245,48244 [para_from,48174.1.1,4032.1.1.1.2.1.2.2.1,demod,43621,52,43403] EQ(n(n(d+x)+
n(n(d)+n(c+n(c))+x)),x).
48308 [para_from,48174.1.1,1047.1.1.1.2.1.2,demod,43403,30025,48245] EQ(n(c+n(c))+x,x).
48310 [binary,48308.1,1.1] $ANS(step_thm).

To enable researchers to conveniently begin a study of two-valued sentential (or propositional) calculus, I include the input file that I cited earlier, that which produces Proof 6 of the Church axiom system (consisting of theses 18, 35, and 49), using as hypotheses the Lukasiewicz axiom system (*L*1, *L*2, and *L*3). This file nicely illustrates one of the uses of the resonance strategy, combined with the use of the hot list strategy for seeking shorter proofs. When using the Lukasiewicz axiom system (consisting of *L*1, *L*2, and *L*3) as the hypotheses and condensed detachment as the inference rule, I know of no shorter proof of the Church axiom system than Proof 6 and Proofs 5, 7, and 8, all four proofs, each of length 21, are presented in my new book.

Input File for Studying Two-Valued Sentential Calculus

```
set(hyper_res).
assign(max_weight,2).
assign(max_proofs, -1).
clear(print_kept).
% clear(for_sub).
clear(back_sub).
% assign(max_seconds, 1200).
assign(max_mem, 24000).
assign(report, 900).
% assign(max_distinct_vars, 4).
% assign(pick_given_ratio, 3).
assign(heat,1).
set(order_history).
set(input_sos_first).
% set(sos_queue).

weight_list(pick_and_purge).
```

```
%  Following are steps from a late 22-step proof, Proof 4, of Church.
weight(P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))),2).
weight(P(i(i(x,y),i(i(n(x),x),y))),2).
weight(P(i(i(i(n(x),y),z),i(x,z))),2).
weight(P(i(n(i(i(n(x),x),x)),y)),2).
weight(P(i(i(x,i(y,z)),i(i(u,y),i(x,i(u,z))))),2).
weight(P(i(i(x,y),i(i(n(i(y,z)),i(y,z)),i(x,z)))),2).
weight(P(i(i(x,y),i(n(i(i(n(z),z),z)),y))),2).
weight(P(i(i(x,i(n(i(y,z)),i(y,z))),i(i(u,y),i(x,i(u,z))))),2).
weight(P(i(x,i(n(i(i(n(y),y),y)),z))),2).
weight(P(i(i(x,i(n(y),y)),i(z,i(x,y)))),2).
weight(P(i(i(n(x),y),i(z,i(i(y,x),x)))),2).
weight(P(i(i(x,i(y,z)),i(i(n(z),y),i(x,z)))),2).
weight(P(i(i(x,i(n(y),z)),i(i(u,i(z,y)),i(x,i(u,y))))),2).
weight(P(i(i(n(x),n(y)),i(y,x))),2).
weight(P(i(i(n(x),y),i(i(z,i(u,x)),i(i(y,u),i(z,x))))),2).
weight(P(i(x,i(y,x))),2).
weight(P(i(i(i(x,y),z),i(y,z))),2).
weight(P(i(n(x),i(x,y))),2).
weight(P(i(i(n(x),y),i(n(y),x))),2).
weight(P(i(n(i(x,y)),x)),2).
weight(P(i(i(x,i(y,i(z,u))),i(i(z,y),i(x,i(z,u))))),2).
weight(P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))),2).
end_of_list.

list(usable).
%  The following clause is used with hyperresolution for condensed detachment.
-P(i(x,y)) | -P(x) | P(y).
%  The following disjunctions, except those mentioning Scott,
%  are the negation of known axiom systems.

-P(i(p,i(q,p))) | -P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) | -P(i(n(n(p)),p)) |
-P(i(p,n(n(p)))) | -P(i(i(p,q),i(n(q),n(p)))) | -P(i(i(p,i(q,r)),i(q,i(p,r)))) |
$ANSWER(step_allFrege_18_35_39_40_46_21).  % 21 is dependent.

-P(i(p,i(q,p))) | -P(i(i(p,i(q,r)),i(q,i(p,r)))) |
-P(i(i(q,r),i(i(p,q),i(p,r)))) | -P(i(p,i(n(p),q))) |
-P(i(i(p,q),i(i(n(p),q),q))) | -P(i(i(p,i(p,q)),i(p,q))) |
$ANSWER(step_allHilbert_18_21_22_3_54_30).  % 30 is dependent.

-P(i(p,i(q,p))) | -P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) |
-P(i(i(n(p),n(q)),i(q,p))) | $ANSWER(step_allBEH_Church_FL_18_35_49).

-P(i(i(i(p,q),r),i(q,r))) | -P(i(i(i(p,q),r),i(n(p),r))) |
-P(i(i(n(p),r),i(i(q,r),i(i(p,q),r)))) | $ANSWER(step_allLuka_x_19_37_59).

-P(i(i(i(p,q),r),i(q,r))) | -P(i(i(i(p,q),r),i(n(p),r))) |
-P(i(i(s,i(n(p),r)),i(s,i(i(q,r),i(i(p,q),r))))) | $ANSWER(step_allWos_x_19_37_60).

-P(i(i(p,q),i(i(q,r),i(p,r)))) | -P(i(i(n(p),p),p)) |
-P(i(p,i(n(p),q))) | $ANSWER(step_allLuka_1_2_3).

-P(i(p,p)) | -P(i(p,i(q,p))) | -P(i(i(p,i(q,r)),i(q,i(p,r)))) |
-P(i(i(i(p,q),p),p)) | -P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) |
-P(i(n(n(p)),p)) | -P(i(p,n(n(p)))) | -P(i(i(p,q),i(n(q),n(p)))) |
```

$ANSWER(step_allScott_orig_16_18_21_24_35_39_40_46).

-P(i(p,p)) | -P(i(p,i(q,p))) | -P(i(i(p,i(q,r)),i(q,i(p,r)))) |
-P(i(i(i(p,q),p),p)) | -P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) |
-P(i(n(n(p)),p)) | -P(i(p,n(n(p)))) | -P(i(i(n(p),n(q)),i(q,p))) |
$ANSWER(step_allScott_orig0_16_18_21_24_35_39_40_49).

end_of_list.

list(sos).
% The following three are Luka, 1 2 3.
P(i(i(x,y),i(i(y,z),i(x,z)))).
P(i(i(n(x),x),x)).
P(i(x,i(n(x),y))).

% The following are from Frege, 18 35 21 46 39 40, with 21 dependent.
% P(i(x,i(y,x))). % axiom F1.
% P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))). % axiom F2.
% P(i(i(x,i(y,z)),i(y,i(x,z)))). % axiom F3.
% P(i(i(x,y),i(n(y),n(x)))). % axiom F4.
% P(i(n(n(x)),x)). % axiom F5.
% P(i(x,n(n(x)))). % axiom f6.
end_of_list.

list(passive).
% -P(i(i(i(i(q,r),i(p,r)),s),i(i(p,q),s))) | $ANS(neg_th_04).
% -P(i(i(p,i(q,r)),i(i(s,q),i(p,i(s,r))))) | $ANS(neg_th_05).
% -P(i(i(p,q),i(i(i(p,r),s),i(i(q,r),s)))) | $ANS(neg_th_06).
% -P(i(i(t,i(i(p,r),s)),i(i(p,q),i(t,i(i(q,r),s))))) | $ANS(neg_th_07).
% -P(i(i(q,r),i(i(p,q),i(i(r,s),i(p,s))))) | $ANS(neg_th_08).
% -P(i(i(i(n(p),q),r),i(p,r))) | $ANS(neg_th_09).
% -P(i(p,i(i(i(n(p),p),p),i(i(q,p),p)))) | $ANS(neg_th_10).
% -P(i(i(q,i(i(n(p),p),p)),i(i(n(p),p),p))) | $ANS(neg_th_11).
% -P(i(t,i(i(n(p),p),p))) | $ANS(neg_th_12).
% -P(i(i(n(p),q),i(t,i(i(q,p),p)))) | $ANS(neg_th_13).
% -P(i(i(i(t,i(i(q,p),p)),r),i(i(n(p),q),r))) | $ANS(neg_th_14).
% -P(i(i(n(p),q),i(i(q,p),p))) | $ANS(neg_th_15).
% -P(i(p,p)) | $ANS(neg_th_16).
% -P(i(p,i(i(q,p),p))) | $ANS(neg_th_17).
-P(i(q,i(p,q))) | $ANS(neg_th_18).
-P(i(i(i(p,q),r),i(q,r))) | $ANS(neg_th_19).
% -P(i(p,i(i(p,q),q))) | $ANS(neg_th_20).
-P(i(i(p,i(q,r)),i(q,i(p,r)))) | $ANS(neg_th_21).
-P(i(i(q,r),i(i(p,q),i(p,r)))) | $ANS(neg_th_22).
% -P(i(i(i(q,i(p,r)),s),i(i(p,i(q,r)),s))) | $ANS(neg_th_23).
% -P(i(i(i(p,q),p),p)) | $ANS(neg_th_24).
% -P(i(i(i(p,r),s),i(i(p,q),i(i(q,r),s)))) | $ANS(neg_th_25).
% -P(i(i(i(p,q),r),i(i(r,p),p))) | $ANS(neg_th_26).
% -P(i(i(i(p,q),q),i(i(q,p),p))) | $ANS(neg_th_27).
% -P(i(i(i(i(r,p),p),s),i(i(i(p,q),r),s))) | $ANS(neg_th_28).
% -P(i(i(i(p,q),r),i(i(p,r),r))) | $ANS(neg_th_29).
-P(i(i(p,i(p,q)),i(p,q))) | $ANS(neg_th_30).
% -P(i(i(p,s),i(i(i(p,q),r),i(i(s,r),r)))) | $ANS(neg_th_31).
% -P(i(i(i(p,q),r),i(i(p,s),i(i(s,r),r)))) | $ANS(neg_th_32).
% -P(i(i(p,s),i(i(s,i(q,i(p,r))),i(q,i(p,r))))) | $ANS(neg_th_33).

```
% -P(i(i(s,i(q,i(p,r))),i(i(p,s),i(q,i(p,r))))) | $ANS(neg_th_34).
-P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) | $ANS(neg_th_35).
% -P(i(n(p),i(p,q))) | $ANS(neg_th_36).
-P(i(i(i(p,q),r),i(n(p),r))) | $ANS(neg_th_37).
% -P(i(i(p,n(p)),n(p))) | $ANS(neg_th_38).
-P(i(n(n(p)),p)) | $ANS(neg_th_39).
-P(i(p,n(n(p)))) | $ANS(neg_th_40).
% -P(i(i(p,q),i(n(n(p)),q))) | $ANS(neg_th_41).
% -P(i(i(i(n(n(p)),q),r),i(i(p,q),r))) | $ANS(neg_th_42).
% -P(i(i(p,q),i(i(q,n(p)),n(p)))) | $ANS(neg_th_43).
% -P(i(i(s,i(q,n(p))),i(i(p,q),i(s,n(p))))) | $ANS(neg_th_44).
% -P(i(i(s,i(q,p)),i(i(n(p),q),i(s,p)))) | $ANS(neg_th_45).
-P(i(i(p,q),i(n(q),n(p)))) | $ANS(neg_th_46).
% -P(i(i(p,n(q)),i(q,n(p)))) | $ANS(neg_th_47).
% -P(i(i(n(p),q),i(n(q),p))) | $ANS(neg_th_48).
-P(i(i(n(p),n(q)),i(q,p))) | $ANS(neg_th_49).
% -P(i(i(i(n(q),p),r),i(i(n(p),q),r))) | $ANS(neg_th_50).
% -P(i(i(p,i(q,r)),i(p,i(n(r),n(q))))) | $ANS(neg_th_51).
% -P(i(i(p,i(q,n(r))),i(p,i(r,n(q))))) | $ANS(neg_th_52).
% -P(i(i(n(p),q),i(i(p,q),q))) | $ANS(neg_th_53).
-P(i(i(p,q),i(i(n(p),q),q))) | $ANS(neg_th_54).
% -P(i(i(p,q),i(i(p,n(q)),n(p)))) | $ANS(neg_th_55).
% -P(i(i(i(i(p,q),q),r),i(i(n(p),q),r))) | $ANS(neg_th_56).
% -P(i(i(n(p),r),i(i(p,q),i(i(q,r),r)))) | $ANS(neg_th_57).
% -P(i(i(i(i(p,q),i(i(q,r),r)),s),i(i(n(p),r),s))) | $ANS(neg_th_58).
-P(i(i(n(p),r),i(i(q,r),i(i(p,q),r)))) | $ANS(neg_th_59).
-P(i(i(s,i(n(p),r)),i(s,i(i(q,r),i(i(p,q),r))))) | $ANS(neg_th_60).
% -P(i(i(p,r),i(i(q,r),i(i(n(p),q),r)))) | $ANS(neg_th_61).
% -P(i(i(n(n(p)),q),i(p,q))) | $ANS(neg_th_62).
% -P(i(q,i(p,p))) | $ANS(neg_th_63).
% -P(i(n(i(p,p)),q)) | $ANS(neg_th_64).
% -P(i(i(n(q),n(i(p,p))),q)) | $ANS(neg_th_65).
% -P(i(n(i(p,q)),p)) | $ANS(neg_th_66).
% -P(i(n(i(p,q)),n(q))) | $ANS(neg_th_67).
% -P(i(n(i(p,n(q))),q)) | $ANS(neg_th_68).
% -P(i(p,i(n(q),n(i(p,q))))) | $ANS(neg_th_69).
% -P(i(p,i(q,n(i(p,n(q)))))) | $ANS(neg_th_70).
% -P(n(i(i(p,p),n(i(q,q))))) | $ANS(neg_th_71).
end_of_list.

list(demodulators).
%  (n(n(x)) = junk).
 (n(n(n(x))) = junk).
%  (i(i(x,x),y) = junk).
%  (i(y,i(x,x)) = junk).
(i(junk,x) = junk).
(i(x,junk) = junk).
(n(junk) = junk).
(P(junk) = $T).
end_of_list.

list(hot).
%  The following clause is used with hyperresolution for condensed detachment.
-P(i(x,y)) | -P(x) | P(y).
%  The following three are Luka, 1 2 3.
```

P(i(i(x,y),i(i(y,z),i(x,z)))).
P(i(i(n(x),x),x)).
P(i(x,i(n(x),y))).
end_of_list.

Proof 6 of the Church Axiom System

-----> EMPTY CLAUSE at 0.92 sec ----> 84 [hyper,4,56,78,53]
$ANSWER(step_allBEH_Church_FL_18_35_49).

Length of proof is 21. Level of proof is 15.

---------- PROOF ----------

1 [] -P(i(x,y)) | -P(x) | P(y).
4 [] -P(i(p,i(q,p))) | -P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) | -P(i(i(n(p),n(q)),i(q,p))) | $
$ANSWER(step_allBEH_Church_FL_18_35_49).
10 [] P(i(i(x,y),i(i(y,z),i(x,z)))).
11 [] P(i(i(n(x),x),x)).
12 [] P(i(x,i(n(x),y))).
32 [] -P(i(x,y)) | -P(x) | P(y).
33 [] P(i(i(x,y),i(i(y,z),i(x,z)))).
34 [] P(i(i(n(x),x),x)).
35 [] P(i(x,i(n(x),y))).
---
36 [hyper,1,10,10] P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))).
40 [hyper,1,10,12] P(i(i(i(n(x),y),z),i(x,z))).
41 [hyper,1,12,11] P(i(n(i(i(n(x),x),x)),y)).
42 (heat=1) [hyper,32,33,41] P(i(i(x,y),i(n(i(i(n(z),z),z)),y))).
43 [hyper,1,36,36] P(i(i(x,i(y,z)),i(i(u,y),i(x,i(u,z))))).
44 (heat=1) [hyper,32,43,34] P(i(i(x,y),i(i(n(i(y,z)),i(y,z)),i(x,z)))).
45 [hyper,1,40,42] P(i(x,i(n(i(i(n(y),y),y)),z))).
46 [hyper,1,43,44] P(i(i(x,i(n(i(y,z)),i(y,z))),i(i(u,y),i(x,i(u,z))))).
47 [hyper,1,46,45] P(i(i(x,i(n(y),y)),i(z,i(x,y)))).
49 [hyper,1,36,47] P(i(i(n(x),y),i(z,i(i(y,x),x)))).
52 [hyper,1,46,49] P(i(i(x,i(y,z)),i(i(n(z),y),i(x,z)))).
53 (heat=1) [hyper,32,52,35] P(i(i(n(x),n(y)),i(y,x))).
55 [hyper,1,43,52] P(i(i(x,i(n(y),z)),i(i(u,i(z,y)),i(x,i(u,y))))).
56 [hyper,1,40,53] P(i(x,i(y,x))).
58 (heat=1) [hyper,32,33,56] P(i(i(i(x,y),z),i(y,z))).
60 [hyper,1,36,55] P(i(i(n(x),y),i(i(z,i(u,x)),i(i(y,u),i(z,x))))).
62 [hyper,1,58,53] P(i(n(x),i(x,y))).
67 [hyper,1,52,62] P(i(i(n(x),y),i(n(y),x))).
72 [hyper,1,67,62] P(i(n(i(x,y)),x)).
76 [hyper,1,60,72] P(i(i(x,i(y,i(z,u))),i(i(z,y),i(x,i(z,u))))).
78 (heat=1) [hyper,32,76,33] P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).
84 [hyper,4,56,78,53] $ANSWER(step_allBEH_Church_FL_18_35_49).

The following proof and its earlier 23-step relative are the shortest proofs known to me of the Hilbert axiom system (consisting of theses, 3, 18, 21, 22, 30, and 54, of which thesis 30 is dependent), when using the Lukasiewicz axiom system (consisting of *L*1, *L*2, and *L*3) as the hypotheses and condensed detachment as the inference rule.

The Later 23-Step Proof of the Hilbert Axiom System

-----> EMPTY CLAUSE at 3.16 sec ----> 95 [hyper,3,57,60,63,12,89,91] $ANSWER(step_allHilbert_18_21_22_3_54_30).

Length of proof is 23. Level of proof is 16.

---------- PROOF ----------

1 [] -P(i(x,y)) | -P(x) | P(y).
3 [] -P(i(p,i(q,p))) | -P(i(i(p,i(q,r)),i(q,i(p,r)))) | -P(i(i(q,r),i(i(p,q),i(p,r)))) | -P(i(p,i(n(p),q))) | $
-P(i(i(p,q),i(i(n(p),q),q))) | -P(i(i(p,i(p,q)),i(p,q))) | $
$ANSWER(step_allHilbert_18_21_22_3_54_30).
10 [] P(i(i(x,y),i(i(y,z),i(x,z)))).
11 [] P(i(i(n(x),x),x)).
12 [] P(i(x,i(n(x),y))).
32 [] -P(i(x,y)) | -P(x) | P(y).
33 [] P(i(i(x,y),i(i(y,z),i(x,z)))).
34 [] P(i(i(n(x),x),x)).
35 [] P(i(x,i(n(x),y))).
---
36 [hyper,1,10,10] P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))).
40 [hyper,1,10,12] P(i(i(i(n(x),y),z),i(x,z))).
41 [hyper,1,12,11] P(i(n(i(i(n(x),x),x)),y)).
42 (heat=1) [hyper,32,40,34] P(i(x,x)).
43 (heat=1) [hyper,32,33,41] P(i(i(x,y),i(n(i(i(n(z),z),z)),y))).
44 [hyper,1,36,36] P(i(i(x,i(y,z)),i(i(u,y),i(x,i(u,z))))).
45 (heat=1) [hyper,32,44,34] P(i(i(x,y),i(i(n(i(y,z)),i(y,z)),i(x,z)))).
46 [hyper,1,40,43] P(i(x,i(n(i(i(n(y),y),y)),z))).
47 [hyper,1,44,45] P(i(i(x,i(n(i(y,z)),i(y,z))),i(i(u,y),i(x,i(u,z))))).
48 [hyper,1,47,46] P(i(i(x,i(n(y),y)),i(z,i(x,y)))).
50 [hyper,1,36,48] P(i(i(n(x),y),i(z,i(i(y,x),x)))).
51 [hyper,1,47,50] P(i(i(x,i(y,z)),i(i(n(z),y),i(x,z)))).
54 [hyper,1,44,51] P(i(i(x,i(n(y),z)),i(i(u,i(z,y)),i(x,i(u,y))))).
56 (heat=1) [hyper,32,54,35] P(i(i(x,i(y,z)),i(z,i(x,z)))).
57 (heat=2) [hyper,32,56,35] P(i(x,i(y,x))).
60 [hyper,1,54,57] P(i(i(x,i(y,z)),i(y,i(x,z)))).
62 (heat=1) [hyper,32,60,35] P(i(n(x),i(x,y))).
63 (heat=1) [hyper,32,60,33] P(i(i(x,y),i(i(z,x),i(z,y)))).
71 [hyper,1,51,62] P(i(i(n(x),y),i(n(y),x))).
78 [hyper,1,54,71] P(i(i(x,i(y,z)),i(i(n(y),z),i(x,z)))).
86 [hyper,1,78,42] P(i(i(n(x),y),i(i(x,y),y))).
89 [hyper,1,60,86] P(i(i(x,y),i(i(n(x),y),y))).
91 [hyper,1,86,62] P(i(i(x,i(x,y)),i(x,y))).
95 [hyper,3,57,60,63,12,89,91] $ANSWER(step_allHilbert_18_21_22_3_54_30).

Next, I give the shortest proof of which I know for the Frege axiom system (consisting of theses 18, 21, 35, 39, 40, and 46, of which thesis 21 is dependent on the other five), when using the Lukasiewicz axiom system (consisting of *L*1, *L*2, and *L*3) as the hypotheses and condensed detachment as the inference rule.

A 28-Step Proof of the Frege Axiom System

```
----- Otter 3.0.2b+, Aug 1994 -----
The job was started by wos on altair.mcs.anl.gov, Wed Mar 22 11:12:34 1995
The command was "otter302c".

-----> EMPTY CLAUSE at   1.96 sec ----> 82 [hyper,2,44,72,60,67,77,46]
$ANSWER(step_allFrege_18_35_39_40_46_21).
Length of proof is 28.  Level of proof is 17.

---------- PROOF ----------

1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] -P(i(p,i(q,p))) | -P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) | -P(i(n(n(p)),p)) | $
-P(i(p,n(n(p)))) | -P(i(i(p,q),i(n(q),n(p)))) | -P(i(i(p,i(q,r)),i(q,i(p,r)))) | $
$ANSWER(step_allFrege_18_35_39_40_46_21).
8 [] P(i(i(x,y),i(i(y,z),i(x,z)))).
9 [] P(i(i(n(x),x),x)).
10 [] P(i(x,i(n(x),y))).
----------
25 [hyper,1,8,8] P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))).
27 [hyper,1,8,10] P(i(i(i(n(x),y),z),i(x,z))).
28 [hyper,1,10,9] P(i(n(i(i(n(x),x),x)),y)).
29 [hyper,1,25,25] P(i(i(x,i(y,z)),i(i(u,y),i(x,i(u,z))))).
30 [hyper,1,25,8] P(i(i(x,y),i(i(i(x,z),u),i(i(y,z),u)))).
31 [hyper,1,29,30] P(i(i(x,i(i(y,z),u)),i(i(y,v),i(x,i(i(v,z),u))))).
32 [hyper,1,30,28] P(i(i(i(n(i(i(n(x),x),x)),y),z),i(i(u,y),z))).
33 [hyper,1,32,9] P(i(i(x,i(i(n(y),y),y)),i(i(n(y),y),y))).
34 [hyper,1,27,33] P(i(x,i(i(n(y),y),y))).
35 [hyper,1,31,34] P(i(i(n(x),y),i(z,i(i(y,x),x)))).
36 [hyper,1,8,35] P(i(i(i(x,i(i(y,z),z)),u),i(i(n(z),y),u))).
37 [hyper,1,36,9] P(i(i(n(x),y),i(i(y,x),x))).
38 [hyper,1,29,37] P(i(i(x,i(y,z)),i(i(n(z),y),i(x,z)))).
39 [hyper,1,31,38] P(i(i(n(x),y),i(i(z,i(u,x)),i(i(y,u),i(z,x))))).
40 [hyper,1,29,38] P(i(i(x,i(n(y),z)),i(i(u,i(z,y)),i(x,i(u,y))))).
41 [hyper,1,38,10] P(i(i(n(x),n(y)),i(y,x))).
44 [hyper,1,27,41] P(i(x,i(y,x))).
46 [hyper,1,40,44] P(i(i(x,i(y,z)),i(y,i(x,z)))).
51 [hyper,1,46,10] P(i(n(x),i(x,y))).
53 [hyper,1,38,51] P(i(i(n(x),y),i(n(y),x))).
54 [hyper,1,8,51] P(i(i(i(x,y),z),i(n(x),z))).
58 [hyper,1,53,51] P(i(n(i(x,y)),x)).
60 [hyper,1,54,9] P(i(n(n(x)),x)).
65 [hyper,1,39,58] P(i(i(x,i(y,i(z,u))),i(i(z,y),i(x,i(z,u))))).
67 [hyper,1,41,60] P(i(x,n(n(x)))).
69 [hyper,1,39,60] P(i(i(x,i(y,n(z))),i(i(z,y),i(x,n(z))))).
72 [hyper,1,65,8] P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))).
77 [hyper,1,69,51] P(i(i(x,y),i(n(y),n(x)))).
82 [hyper,2,44,72,60,67,77,46] $ANSWER(step_allFrege_18_35_39_40_46_21).
```

For those interested in the alternate Lukasiewicz axiom system (consisting of theses 19, 37, 59), I give the following proof. I know of no shorter proof, when using the Lukasiewicz axiom system (consisting of *L*1, *L*2, and *L*3) as the hypotheses and condensed detachment as the inference rule.

The Later 24-Step Proof of the Alternate Lukasiewicz Axiom System

-----> EMPTY CLAUSE at 3.14 sec ----> 100 [hyper,5,62,67,90]
$ANSWER(step_allLuka_x_19_37_59).

Length of proof is 24. Level of proof is 16.

---------- PROOF ----------

1 [] -P(i(x,y)) | -P(x) | P(y).
5 [] -P(i(i(i(p,q),r),i(q,r))) | -P(i(i(i(p,q),r),i(n(p),r))) | $
-P(i(i(n(p),r),i(i(q,r),i(i(p,q),r)))) | $ANSWER(step_allLuka_x_19_37_59).
10 [] P(i(i(x,y),i(i(y,z),i(x,z)))).
11 [] P(i(i(n(x),x),x)).
12 [] P(i(x,i(n(x),y))).
32 [] -P(i(x,y)) | -P(x) | P(y).
33 [] P(i(i(x,y),i(i(y,z),i(x,z)))).
34 [] P(i(i(n(x),x),x)).
35 [] P(i(x,i(n(x),y))).
---
36 [hyper,1,10,10] P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))).
40 [hyper,1,10,12] P(i(i(i(n(x),y),z),i(x,z))).
41 [hyper,1,12,11] P(i(n(i(i(n(x),x),x)),y)).
42 (heat=1) [hyper,32,33,41] P(i(i(x,y),i(n(i(i(n(z),z),z)),y))).
43 [hyper,1,36,36] P(i(i(x,i(y,z)),i(i(u,y),i(x,i(u,z))))).
44 (heat=1) [hyper,32,43,34] P(i(i(x,y),i(i(n(i(y,z)),i(y,z)),i(x,z)))).
45 [hyper,1,40,42] P(i(x,i(n(i(i(n(y),y),y)),z))).
46 [hyper,1,43,44] P(i(i(x,i(n(i(y,z)),i(y,z))),i(i(u,y),i(x,i(u,z))))).
47 [hyper,1,46,45] P(i(i(x,i(n(y),y)),i(z,i(x,y)))).
50 [hyper,1,36,47] P(i(i(n(x),y),i(z,i(i(y,x),x)))).
51 [hyper,1,46,50] P(i(i(x,i(y,z)),i(i(n(z),y),i(x,z)))).
54 [hyper,1,43,51] P(i(i(x,i(n(y),z)),i(i(u,i(z,y)),i(x,i(u,y))))).
55 (heat=1) [hyper,32,54,35] P(i(i(x,i(y,z)),i(z,i(x,z)))).
56 (heat=2) [hyper,32,55,35] P(i(x,i(y,x))).
58 [hyper,1,54,50] P(i(i(x,i(i(i(y,z),z),u)),i(i(n(z),y),i(x,u)))).
60 [hyper,1,54,56] P(i(i(x,i(y,z)),i(y,i(x,z)))).
62 [hyper,1,10,56] P(i(i(i(x,y),z),i(y,z))).
64 (heat=1) [hyper,32,60,35] P(i(n(x),i(x,y))).
65 (heat=1) [hyper,32,60,33] P(i(i(x,y),i(i(z,x),i(z,y)))).
67 (heat=2) [hyper,32,33,64] P(i(i(i(x,y),z),i(n(x),z))).
69 (heat=2) [hyper,32,33,65] P(i(i(i(i(x,y),i(x,z)),u),i(i(y,z),u))).
76 [hyper,1,54,67] P(i(i(x,i(y,z)),i(i(i(z,u),y),i(x,z)))).
82 [hyper,1,69,76] P(i(i(x,y),i(i(i(y,z),u),i(i(u,x),y)))).
90 [hyper,1,58,82] P(i(i(n(x),y),i(i(z,y),i(i(x,z),y)))).
100 [hyper,5,62,67,90] $ANSWER(step_allLuka_x_19_37_59).

For the final item regarding axiom systems deducible from the Lukasiewicz system, I give the shortest proof known to me of my axiom system, that consisting of theses 19, 37, and 60.

The Later 25-Step Proof of the Wos Axiom System

-----> EMPTY CLAUSE at 2.20 sec ----> 117 [hyper,6,62,67,106]
$ANSWER(step_allWos_x_19_37_60).

Length of proof is 25. Level of proof is 17.

---------- PROOF ----------

1 [] -P(i(x,y)) | -P(x) | P(y).
6 [] -P(i(i(i(p,q),r),i(q,r))) | -P(i(i(i(p,q),r),i(n(p),r))) | $
-P(i(i(s,i(n(p),r)),i(s,i(i(q,r),i(i(p,q),r))))) | $ANSWER(step_allWos_x_19_37_60).
10 [] P(i(i(x,y),i(i(y,z),i(x,z)))).
11 [] P(i(i(n(x),x),x)).
12 [] P(i(x,i(n(x),y))).
32 [] -P(i(x,y)) | -P(x) | P(y).
33 [] P(i(i(x,y),i(i(y,z),i(x,z)))).
34 [] P(i(i(n(x),x),x)).
35 [] P(i(x,i(n(x),y))).
----
36 [hyper,1,10,10] P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))).
40 [hyper,1,10,12] P(i(i(i(n(x),y),z),i(x,z))).
41 [hyper,1,12,11] P(i(n(i(i(n(x),x),x)),y)).
42 (heat=1) [hyper,32,33,41] P(i(i(x,y),i(n(i(i(n(z),z),z)),y))).
43 [hyper,1,36,36] P(i(i(x,i(y,z)),i(i(u,y),i(x,i(u,z))))).
44 (heat=1) [hyper,32,43,34] P(i(i(x,y),i(i(n(i(y,z)),i(y,z)),i(x,z)))).
45 [hyper,1,40,42] P(i(x,i(n(i(i(n(y),y),y)),z))).
46 [hyper,1,43,44] P(i(i(x,i(n(i(y,z)),i(y,z))),i(i(u,y),i(x,i(u,z))))).
47 [hyper,1,46,45] P(i(i(x,i(n(y),y)),i(z,i(x,y)))).
50 [hyper,1,36,47] P(i(i(n(x),y),i(z,i(i(y,x),x)))).
51 [hyper,1,46,50] P(i(i(x,i(y,z)),i(i(n(z),y),i(x,z)))).
54 [hyper,1,43,51] P(i(i(x,i(n(y),z)),i(i(u,i(z,y)),i(x,i(u,y))))).
55 (heat=1) [hyper,32,54,35] P(i(i(x,i(y,z)),i(z,i(x,z)))).
56 (heat=2) [hyper,32,55,35] P(i(x,i(y,x))).
58 [hyper,1,54,50] P(i(i(x,i(i(i(y,z),z),u)),i(i(n(z),y),i(x,u)))).
60 [hyper,1,54,56] P(i(i(x,i(y,z)),i(y,i(x,z)))).
62 [hyper,1,10,56] P(i(i(i(x,y),z),i(y,z))).
64 (heat=1) [hyper,32,60,35] P(i(n(x),i(x,y))).
65 (heat=1) [hyper,32,60,33] P(i(i(x,y),i(i(z,x),i(z,y)))).
67 (heat=2) [hyper,32,33,64] P(i(i(i(x,y),z),i(n(x),z))).
69 (heat=2) [hyper,32,33,65] P(i(i(i(i(x,y),i(x,z)),u),i(i(y,z),u))).
76 [hyper,1,54,67] P(i(i(x,i(y,z)),i(i(i(z,u),y),i(x,z)))).
84 [hyper,1,69,76] P(i(i(x,y),i(i(i(y,z),u),i(i(u,x),y)))).
93 [hyper,1,58,84] P(i(i(n(x),y),i(i(z,y),i(i(x,z),y)))).
106 [hyper,1,65,93] P(i(i(x,i(n(y),z)),i(x,i(i(u,z),i(i(y,u),z))))).
117 [hyper,6,62,67,106] $ANSWER(step_allWos_x_19_37_60).

I close my focus on two-valued sentential calculus with an input file that can be used to begin one's attack on finding a means for an automated reasoning program to prove, definitely not in a proof checking mode, that Meredith's single axiom suffices for an axiom system for two-valued sentential calculus. To aid one's research, I also include (essentially) Meredith's proof; it was produced with a cursory proof-checking run, using his steps as resonators and assigning max_weight the value 2.

Input File for Studying Meredith's Single Axiom

```
set(hyper_res).
assign(max_weight, 28).
assign(change_limit_after, 2000).
assign(new_max_weight, 20).
assign(max_proofs, -1).
clear(print_kept).
clear(back_sub).
assign(max_mem, 110000).
assign(report, 1800).
assign(max_distinct_vars, 7).
assign(pick_given_ratio, 3).
assign(heat,1).
set(order_history).
set(input_sos_first).

weight_list(pick_given).
%  The following is Meredith's single axiom.
weight(P(i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x)))),2).
%  Following are the 17 from the known axiom systems; using resonance.
weight(P(i(i(x,y),i(i(y,z),i(x,z)))),2).
weight(P(i(i(n(x),x),x)),2).
weight(P(i(x,i(n(x),y))),2).
weight(P(i(x,i(y,x))),2).
weight(P(i(i(i(x,y),z),i(y,z))),2).
weight(P(i(i(x,i(y,z)),i(y,i(x,z)))),2).
weight(P(i(i(x,y),i(i(z,x),i(z,y)))),2).
weight(P(i(i(x,i(x,y)),i(x,y))),2).
weight(P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))),2).
weight(P(i(i(i(x,y),z),i(n(x),z))),2).
weight(P(i(n(n(x)),x)),2).
weight(P(i(x,n(n(x)))),2).
weight(P(i(i(x,y),i(n(y),n(x)))),2).
weight(P(i(i(n(x),n(y)),i(y,x))),2).
weight(P(i(i(x,y),i(i(n(x),y),y))),2).
weight(P(i(i(n(x),y),i(i(z,y),i(i(x,z),y)))),2).
weight(P(i(i(x,i(n(y),z)),i(x,i(i(u,z),i(i(y,u),z))))),2).
%  Following is for recursive tail strategy.
weight(i($(1),$(2)),1).
end_of_list.

list(usable).
%  Following is for condensed detachment.
-P(i(x,y)) | -P(x) | P(y).

%  The following disjunctions are known axiom systems.

-P(i(q,i(p,q))) | -P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) | -P(i(n(n(p)),p)) | $
-P(i(p,n(n(p)))) | -P(i(i(p,q),i(n(q),n(p)))) | -P(i(i(p,i(q,r)),i(q,i(p,r)))) | $
$ANSWER(step_allFrege_18_35_39_40_46_21).  % 21 is dependent.

-P(i(q,i(p,q))) | -P(i(i(p,i(q,r)),i(q,i(p,r)))) | -P(i(i(q,r),i(i(p,q),i(p,r)))) | $
-P(i(p,i(n(p),q))) | -P(i(i(p,i(q,r)),i(q,i(p,r)))) | -P(i(i(q,r),i(i(p,q),i(p,r)))) | $
$ANSWER(step_allHilbert_18_21_22_3_54_30).  % 30 is dependent.
```

```
-P(i(q,i(p,q))) | -P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) | -P(i(i(n(p),n(q)),i(q,p))) | $
$ANSWER(step_allBEH_Church_FL_18_35_49).

-P(i(i(i(p,q),r),i(q,r))) | -P(i(i(i(p,q),r),i(n(p),r))) | -P(i(i(n(p),r),i(i(q,r),i(i(p,q),r)))) | $
$ANSWER(step_allLuka_x_19_37_59).

-P(i(i(i(p,q),r),i(q,r))) | -P(i(i(i(p,q),r),i(n(p),r))) | $
-P(i(i(s,i(n(p),r)),i(s,i(i(q,r),i(i(p,q),r))))) | $ANSWER(step_allWos_x_19_37_60).

-P(i(i(p,q),i(i(q,r),i(p,r)))) | -P(i(i(n(p),p),p)) | -P(i(p,i(n(p),q))) | $
$ANSWER(step_allLuka_1_2_3).
end_of_list.

list(sos).
%  The following is Meredith's single axiom.
P(i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x)))).  % CN-CAM
%  The following three are Luka, 1 2 3.
% P(i(i(x,y),i(i(y,z),i(x,z)))).
% P(i(i(n(x),x),x)).
% P(i(x,i(n(x),y))).
end_of_list.

list(passive).
-P(i(i(p,q),i(i(q,r),i(p,r)))) | $ANSWER(step_L1).
-P(i(i(n(p),p),p)) | $ANSWER(step_L2).
-P(i(p,i(n(p),q))) | $ANSWER(step_L3).
-P(i(q,i(p,q))) | $ANSWER(step_18).
-P(i(i(i(p,q),r),i(q,r))) | $ANSWER(step_19).
-P(i(i(p,i(q,r)),i(q,i(p,r)))) | $ANSWER(step_21).
-P(i(i(q,r),i(i(p,q),i(p,r)))) | $ANSWER(step_22).
-P(i(i(p,i(p,q)),i(p,q))) | $ANSWER(step_30).
-P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) | $ANSWER(step_35).
-P(i(i(i(p,q),r),i(n(p),r))) | $ANSWER(step_37).
-P(i(n(n(p)),p)) | $ANSWER(step_39).
-P(i(p,n(n(p)))) | $ANSWER(step_40).
-P(i(i(p,q),i(n(q),n(p)))) | $ANSWER(step_46).
-P(i(i(n(p),n(q)),i(q,p))) | $ANSWER(step_49).
-P(i(i(p,q),i(i(n(p),q),q))) | $ANSWER(step_54).
-P(i(i(n(p),r),i(i(q,r),i(i(p,q),r)))) | $ANSWER(step_59).
-P(i(i(s,i(n(p),r)),i(s,i(i(q,r),i(i(p,q),r))))) | $ANSWER(step_60).
-P(i(n(n(a)),a)) | $ANSWER(lemma_24).
-P(i(a,n(n(a)))) | $ANSWER(lemma_29).
-P(i(i(a,b),i(i(c,a),i(c,b)))) | $ANSWER(lemma_25).
-P(i(i(a,b),i(n(b),n(a)))) | $ANSWER(lemma_36).
end_of_list.

 list(demodulators).
 % (n(n(x)) = junk).
   (n(n(n(x))) = junk).
% (i(i(x,x),y) = junk).
% (i(y,i(x,x)) = junk).
% (i(n(i(x,x)),y) = junk).
% (i(y,n(i(x,x))) = junk).
 (i(junk,x) = junk).
```

```
 (i(x,junk) = junk).
 (n(junk) = junk).
 (P(junk) = $T).
 end_of_list.

list(hot).
%  Following is for condensed detachment.
-P(i(x,y)) | -P(x) | P(y).
%  Following is Meredith's single axiom.
P(i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x)))).  % CN-CAM
end_of_list.
```

Meredith's Proof

```
-----> EMPTY CLAUSE at    1.35 sec ----> 58 [hyper,6,57,47,38]
$ANSWER(Luka,[1,2,3]).

Length of proof is 41.  Level of proof is 30.

---------- PROOF ----------

1 [] -P(i(x,y)) | -P(x) | P(y).
6 [] -P(i(i(p,q),i(i(q,r),i(p,r)))) | -P(i(i(n(p),p),p)) | -P(i(p,i(n(p),q))) | $
$ANSWER(Luka,[1,2,3]).
7 [] P(i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x)))).
------------------------------------
8 [hyper,1,7,7] P(i(i(i(i(x,y),i(z,y)),i(y,u)),i(v,i(y,u)))).
9 [hyper,1,7,8] P(i(i(i(x,i(n(y),z)),u),i(y,u))).
10 [hyper,1,7,9] P(i(i(i(x,x),y),i(z,y))).
11 [hyper,1,10,10] P(i(x,i(y,i(z,z)))).
13 [hyper,1,7,11] P(i(i(i(x,i(y,y)),z),i(u,z))).
15 [hyper,1,7,13] P(i(i(i(x,y),z),i(y,z))).
17 [hyper,1,15,7] P(i(x,i(i(x,y),i(z,y)))).
18 [hyper,1,15,17] P(i(x,i(i(i(y,x),z),i(u,z)))).
19 [hyper,1,17,9] P(i(i(i(i(i(x,i(n(y),z)),u),i(y,u)),v),i(w,v))).
20 [hyper,1,7,18] P(i(i(i(i(i(x,i(i(i(y,z),i(n(u),n(v))),u)),w),i(v6,w)),y),i(v,y))).
21 [hyper,1,7,19] P(i(i(i(x,y),i(z,i(n(n(y)),u))),i(v,i(z,i(n(n(y)),u))))).
22 [hyper,1,7,20] P(i(i(i(x,y),i(z,i(i(i(y,u),i(n(v),n(x))),v))),i(w,i(z,i(i(i(y,u),i(n(v),n(x))),v))))).
23 [hyper,1,21,7] P(i(x,i(i(y,z),i(n(n(y)),z)))).
24 [hyper,1,22,17] P(i(x,i(i(i(y,z),u),i(i(i(z,v),i(n(u),n(y))),u)))).
25 [hyper,1,23,23] P(i(i(x,y),i(n(n(x)),y))).
26 [hyper,1,7,23] P(i(i(i(i(x,y),i(n(n(x)),y)),z),i(u,z))).
27 [hyper,1,24,24] P(i(i(i(x,y),z),i(i(i(y,u),i(n(z),n(x))),z))).
29 [hyper,1,10,25] P(i(x,i(n(n(y)),y))).
30 [hyper,1,27,18] P(i(i(i(x,y),i(n(i(i(i(z,i(u,x)),v),i(w,v))),n(u))),i(i(i(z,i(u,x)),v),i(w,v)))).
31 [hyper,1,17,29] P(i(i(i(x,i(n(n(y)),y)),z),i(u,z))).
32 [hyper,1,7,30] P(i(i(i(i(i(x,i(y,i(z,u))),v),i(w,v)),z),i(v6,z))).
33 [hyper,1,7,32] P(i(i(i(x,y),i(z,i(u,i(y,v)))),i(w,i(z,i(u,i(y,v)))))).
34 [hyper,1,33,7] P(i(x,i(i(y,i(y,z)),i(u,i(y,z))))).
35 [hyper,1,34,34] P(i(i(x,i(x,y)),i(z,i(x,y)))).
36 [hyper,1,35,35] P(i(x,i(i(y,i(y,z)),i(y,z)))).
37 [hyper,1,36,36] P(i(i(x,i(x,y)),i(x,y))).
38 [hyper,1,9,37] P(i(x,i(n(x),y))).
39 [hyper,1,37,31] P(i(i(i(x,i(n(n(y)),y)),z),z)).
40 [hyper,1,37,26] P(i(i(i(i(x,y),i(n(n(x)),y)),z),z)).
```

42 [hyper,1,25,39] P(i(n(n(i(i(x,i(n(n(y)),y)),z))),z)).
43 [hyper,1,7,40] P(i(i(n(x),x),i(y,x))).
46 [hyper,1,18,42] P(i(i(i(x,i(n(n(i(i(y,i(n(n(z)),z)),u))),u)),v),i(w,v))).
47 [hyper,1,37,43] P(i(i(n(x),x),x)).
48 [hyper,1,7,46] P(i(i(i(x,n(i(i(y,i(n(n(z)),z)),n(u)))),v),i(u,v))).
49 [hyper,1,48,47] P(i(x,n(i(i(y,i(n(n(z)),z)),n(x))))).
50 [hyper,1,18,49] P(i(i(i(x,i(y,n(i(i(z,i(n(n(u)),u)),n(y))))),v),i(w,v))).
52 [hyper,1,37,50] P(i(i(i(x,i(y,n(i(i(z,i(n(n(u)),u)),n(y))))),v),v)).
53 [hyper,1,7,52] P(i(i(x,y),i(i(i(z,i(n(n(u)),u)),n(n(x))),y))).
54 [hyper,1,53,53] P(i(i(i(x,i(n(n(y)),y)),n(n(i(z,u)))),i(i(i(v,i(n(n(w)),w)),n(n(z))),u))).
55 [hyper,1,7,54] P(i(i(i(i(i(x,i(n(n(y)),y)),n(n(z))),u),v),i(i(z,u),v))).
57 [hyper,1,55,7] P(i(i(x,y),i(i(y,z),i(x,z)))).
58 [hyper,6,57,47,38] $ANSWER(Luka,[1,2,3]).

To end this report, for those interested in equivalential calculus, I give two short proofs. The first is the shortest I know that the formula *XHN* implies the formula *UM*, and the second is the shortest of which I know that the formula *XHK* implies the formula *YQL*.

A Short Proof for *XHN* Implies *UM*

----> UNIT CONFLICT at 0.30 sec ----> 38 [binary,37.1,6.1] $ANSWER(P4_UM).
Length of proof is 19. Level of proof is 14.
---------- PROOF ----------

1 [] -P(e(x,y)) | -P(x) | P(y).
2 [] P(e(x,e(e(y,z),e(e(z,x),y)))).
6 [] -P(e(e(e(a,b),c),e(b,e(c,a)))) | $ANSWER(P4_UM).
----
18 [hyper,1,2,2] P(e(e(x,y),e(e(y,e(z,e(e(u,v),e(e(v,z),u)))),x))).
19 [hyper,1,18,18] P(e(e(e(e(x,e(y,e(e(z,u),e(e(u,y),z)))),v),e(w,e(e(v6,v7),e(e(v7,w),v6)))),e(v,x))).
20 [hyper,1,18,2] P(e(e(e(e(x,y),e(e(y,z),x)),e(u,e(e(v,w),e(e(w,u),v)))),z)).
21 [hyper,1,19,20] P(e(e(x,e(e(y,z),e(e(z,x),y))),e(e(e(u,v),e(e(v,e(w,e(v6,e(e(v7,v8),
e(e(v8,v6),v7)))))),u)),w))).
22 [hyper,1,18,20] P(e(e(x,e(y,e(e(z,u),e(e(u,y),z)))),e(e(e(v,w),e(e(w,x),v)),
e(v6,e(e(v7,v8),e(e(v8,v6),v7))))).
23 [hyper,1,21,2] P(e(e(e(x,y),e(e(y,e(z,e(u,e(e(v,w),e(e(w,u),v)))))),x)),z)).
24 [hyper,1,2,23] P(e(e(x,y),e(e(y,e(e(e(z,u),e(e(u,e(v,e(w,e(e(v6,v7),e(e(v7,w),v6)))))),z)),v)),x))).
25 [hyper,1,24,2] P(e(e(e(e(x,y),e(e(y,z),x)),e(e(e(u,v),e(e(v,e(w,e(v6,e(e(v7,v8),
e(e(v8,v6),v7)))))),u)),w)),z)).
26 [hyper,1,20,25] P(e(e(x,e(e(x,y),e(z,e(e(u,v),e(e(v,z),u))))),e(y,e(w,e(e(v6,v7),e(e(v7,w),v6))))).
27 [hyper,1,26,22] P(e(e(e(e(x,e(e(y,z),e(e(z,x),y))),u),u),e(v,e(e(w,v6),e(e(v6,v),w))))).
28 [hyper,1,22,27] P(e(e(e(x,y),e(e(y,e(e(e(z,e(e(u,v),e(e(v,z),u))),w),w)),x)),e(v6,
e(e(v7,v8),e(e(v8,v6),v7))))).
30 [hyper,1,20,28] P(e(e(e(x,e(e(y,z),e(e(z,x),y))),u),u)).
31 [hyper,1,19,28] P(e(e(e(e(x,e(e(y,z),e(e(z,x),y))),e(e(e(u,e(e(v,w),e(e(w,u),v))),v6),v6)),v7),v7)).
32 [hyper,1,25,30] P(e(e(e(e(x,y),z),e(y,e(u,e(e(v,w),e(e(w,u),v))))),e(z,x))).
33 [hyper,1,30,31] P(e(e(x,y),e(e(y,e(e(z,e(e(u,v),e(e(v,z),u))),e(e(e(w,e(e(v6,v7),
e(e(v7,w),v6))),v8),v8))),x))).
34 [hyper,1,20,32] P(e(e(x,e(e(y,z),e(e(z,e(u,x)),y))),u)).
35 [hyper,1,32,33] P(e(e(e(e(e(x,e(e(y,z),e(e(z,x),y))),u),u),v),e(e(w,v6),e(e(v6,v),w)))).
36 [hyper,1,32,35] P(e(x,e(e(y,e(e(z,u),e(e(u,y),z))),e(e(e(v,w),e(e(w,e(v6,x)),v)),v6)))).
37 [hyper,1,34,36] P(e(e(e(x,y),z),e(y,e(z,x)))).
38 [binary,37.1,6.1] $ANSWER(P4_UM).

A Short Proof for *XHK* Implies *YQL*

----> UNIT CONFLICT at 380.90 sec ----> 4789 [binary,4788.1,3.1] $ANSWER(P1_YQL).
Length of proof is 23. Level of proof is 19.
---------- PROOF ----------

1 [] -P(e(x,y)) | -P(x) | P(y).
2 [] P(e(x,e(e(y,z),e(e(x,z),y)))).
3 [] -P(e(e(a,b),e(e(c,b),e(a,c)))) | $ANSWER(P1_YQL).
16 [] -P(e(x,y)) | -P(x) | P(y).
17 [] P(e(x,e(e(y,z),e(e(x,z),y)))).
---
18 [hyper,1,2,2] P(e(e(x,y),e(e(e(z,e(e(u,v),e(e(z,v),u))),y),x))).
19 (heat=1) [hyper,16,17,18] P(e(e(x,y),e(e(e(e(z,u),e(e(e(v,e(e(w,v6),e(e(v,v6),w))),u),z)),y),x))).
20 (heat=1) [hyper,16,18,17] P(e(e(e(x,e(e(y,z),e(e(x,z),y))),e(e(u,v),e(e(w,v),u))),w)).
21 [hyper,1,18,18] P(e(e(e(x,e(e(y,z),e(e(x,z),y))),e(e(e(u,e(e(v,w),e(e(u,w),v))),v6),v7)),e(v7,v6))).
27 [hyper,1,19,18] P(e(e(e(e(x,y),e(e(e(z,e(e(u,v),e(e(z,v),u))),y),x)),e(e(e(w,e(e(v6,v7),
e(e(w,v7),v6))),v8),v9)),e(v9,v8))).
35 (heat=1) [hyper,16,27,17] P(e(e(e(e(e(x,y),e(e(e(z,e(e(u,v),e(e(z,v),u))),y),x)),w),
e(v6,e(e(v7,v8),e(e(v6,v8),v7)))),w)).
47 [hyper,1,21,20] P(e(e(e(x,y),e(e(e(e(z,e(e(u,v),e(e(z,v),u))),w),y),x)),w)).
58 [hyper,1,35,47] P(e(e(e(e(x,e(e(y,z),e(e(x,z),y))),e(u,e(e(v,w),e(e(u,w),v)))),e(e(e(v6,e(e(v7,v8),
e(e(v6,v8),v7))),v9),v10)),e(v10,v9))).
77 [hyper,1,58,21] P(e(x,e(e(e(e(y,e(e(z,u),e(e(y,u),z))),x),e(e(v,w),e(e(v6,w),v))),v6))).
81 [hyper,1,77,47] P(e(e(e(e(x,e(e(y,z),e(e(x,z),y))),e(e(e(u,v),e(e(e(e(w,e(e(v6,v7),e(e(w,v7),v6))),
v8),v),u)),v8)),e(e(v9,v10),e(e(v11,v10),v9))),v11)).
91 [hyper,1,20,81] P(e(e(e(x,y),e(e(z,e(e(u,v),e(e(z,v),u))),x)),y)).
93 [hyper,1,77,91] P(e(e(e(e(x,e(e(y,z),e(e(x,z),y))),e(e(e(u,v),e(e(w,e(e(v6,v7),
e(e(w,v7),v6))),u)),v)),e(e(v8,v9),e(e(v10,v9),v8))),v10)).
97 [hyper,1,91,47] P(e(e(e(e(x,e(e(y,z),e(e(x,z),y))),e(e(u,e(e(v,w),e(e(u,w),v))),e(v6,v7))),v7),v6)).
121 [hyper,1,20,93] P(e(e(e(x,e(e(y,z),e(e(u,z),y))),u),x)).
133 [hyper,1,121,121] P(e(x,e(e(y,z),e(e(e(e(u,v),e(e(x,v),u)),z),y)))).
154 [hyper,1,18,133] P(e(e(e(x,e(e(y,z),e(e(x,z),y))),e(e(u,v),e(e(e(e(w,v6),e(e(v7,v6),w)),v),u))),v7)).
177 [hyper,1,97,154] P(e(e(e(x,y),e(e(z,y),x)),e(e(u,v),e(e(z,v),u)))).
181 [hyper,1,121,177] P(e(e(e(e(e(x,y),e(e(z,y),x)),u),z),u)).
197 [hyper,1,121,181] P(e(e(e(x,y),e(e(e(e(z,u),e(e(v,u),z)),y),x)),v)).
219 [hyper,1,181,197] P(e(e(e(e(x,y),e(e(z,y),x)),e(e(z,u),v)),e(v,u))).
240 [hyper,1,18,219] P(e(e(e(x,e(e(y,z),e(e(x,z),y))),e(u,v)),e(e(e(w,v6),e(e(v7,v6),w)),e(e(v7,v),u)))).
4776 [hyper,1,219,240] P(e(e(e(x,y),e(e(z,u),e(e(v,w),e(e(y,w),v)))),e(e(x,u),z))).
4788 (heat=1) [hyper,16,4776,17] P(e(e(x,y),e(e(z,y),e(x,z)))).
4789 [binary,4788.1,3.1] $ANSWER(P1_YQL).

References

[Boyer79] Boyer, R., and Moore, J, *A Computational Logic*, Academic Press: New York, 1979.

[Boyer98] Boyer, R. S., and Moore, J S., *A Computational Logic Handbook*, 2nd ed., Academic Press: New York, 1998 (also Web information ftp://ftp.cs.utexas.edu/pub/boyer/nqthm/index.html).

[Hart95] Hart, J., and Kunen, K. ''Single Axioms for Odd Exponent Groups''. *J. Automated Reasoning* **14**, no. 3 (June 1995): 383-412.

[Henkin71] Henkin, L., Monk, J., and Tarski, A., *Cylindric Algebras, Part I*, North-Holland: Amsterdam, 1971.

[Jech94] Jech, T., ''OTTER Experiments in a System of Combinatory Logic'', *J. Automated Reasoning* **14,** no. 3 (1995): 413-426.

[Kalman78] Kalman, J., ''A Shortest Single Axiom for the Classical Equivalential Calculusrq, *Notre Dame J. Formal Logic* **19,** (1978): 141-144.

[Kalman83] Kalman, J., ''Condensed Detachment as a Rule of Inference'', *Studia Logica* **42** (1983): 443-451.

[Kunen95] Kunen, K., ''The Shortest Single Axioms for Groups of Exponent 4'', *Computers and Mathematics with Applications* (special issue on automated reasoning) **29**, no. 2 (February 1995): 1-12.

[Lukasiewicz63] Lukasiewicz, J., *Elements of Mathematical Logic*, Pergamon Press: Oxford, 1963.

[Lukasiewicz70] Lukasiewicz, J., ''The Equivalential Calculus'', pp. 250-277 in *Jan Lukasiewicz: Selected Works*, ed. L. Borkowski, North-Holland: Amsterdam, 1970.

[Lusk84] Lusk, E., and Overbeek, R., *The Automated Reasoning System ITP*, Technical Report ANL-84-27, Argonne National Laboratory, Argonne, Illinois, April 1984.

[McCune90] McCune, W., *OTTER 2.0 Users Guide*, Technical Report ANL-90/9, Argonne National Laboratory, Argonne, Illinois, 1990.

[McCune91] McCune, W., *What's New in OTTER 2.2*, Technical Memorandum ANL/MCS-TM-153, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois, 1991.

[McCune94] McCune, W., *OTTER 3.0 Reference Manual and Guide*, Technical Report ANL-94/6, Argonne National Laboratory, Argonne, Illinois, 1994.

[McCune96a] McCune, W., and Padmanabhan, R., *Automated Deduction in Equational Logic and Cubic Curves*, Lecture Notes in Computer Science, Vol. 1095, Springer-Verlag: Heidelberg, 1996. See http://www.mcs.anl.gov/home/mccune/ar/monograph/ for additional information.

[McCune96b] McCune, W., and Sands, A. D., ''Computer and Human Reasoning: Single Implicative Axioms for Groups and for Abelian Groups'', *American Mathematical Monthly* **103**, no. 10 (1996): 888-892.

[McCune97] McCune, W., ''Solution of the Robbins Problem'', *J. Automated Reasoning*, accepted for publication, 1997.

[Meredith53] Meredith, C. A., ''Single Axioms for the Systems (C,N), (C,0), and (A,N) of the Two-Valued Propositional Calculus'', *J. Computing Systems* **1** (1983): 155-164.

[Veroff92] Veroff, R., and Wos, L., ''The Linked Inference Principle, I: The Formal Treatment'', *J. Automated Reasoning* **8**, no. 2 (April 1982): 213-274.

[Winker90] Winker, S., ''Robbins Algebra: Conditions That Make a Near-Boolean Algebra Boolean'', *J. Automated Reasoning* **6**, no. 4 (December 1990): 465-489.

[Winker92] Winker, S., ''Absorption and Idempotency Criteria for a Problem wn Near-Boolean Algebras'', *J. Algebra* **153**, no. 2 (December 1992): 414-423.

[Wos83] Wos, L., Winker, S., Veroff, R., Smith, B., and Henschen, L., ''Questions concerning Possible Shortest Single Axioms in Equivalential Calculus: An Application of Automated Theorem Proving to Infinite Domains'', *Notre Dame J. Formal Logic* **24** (9183): 205-223.

[Wos84a] Wos, L., Winker, S., Veroff, R., Smith, B., and Henschen, L., ''A New Use of an Automated Reasoning Assistant: Open Questions in Equivalential Calculus and the Study of Infinite Domains'', *Artificial Intelligence* **22** (1984): 303-356.

[Wos84b] Wos, L. , Veroff, R., Smith, B., and McCune, W., ''The Linked Inference Principle, II: The User's Viewpoint'', pp. 316-332 in *Lecture Notes in Computer Science*, Vol. 170, ed. R. E. Shostak, Springer-Verlag: New York, 1984.

[Wos87] Wos, L, *Automated Reasoning: 33 Basic Research Problems*, Prentice-Hall: Englewood Cliffs, N.J., 1987.

[Wos92] Wos, L., Overbeek, R., Lusk, E., and Boyle, J., *Automated Reasoning: Introduction and Applications*, 2nd ed., McGraw-Hill: New York, 1992.

[Wos95a] Wos, L., ''The Resonance Strategy'', *Computers and Mathematics with Applications* (special issue on automated reasoning) **29**, no. 2 (February 1995): 133-178.

[Wos95b] Wos, L., ''Searching for Circles of Pure Proofs'', *J. Automated Reasoning* **15**, no. 3 (1995): 279-315.

[Wos96a] Wos, L., *The Automation of Reasoning: An Experimenter's Notebook with OTTER Tutorial*, Academic Press: New York, 1996. See http://www.mcs.anl.gov/people/wos/index.html for input files and information on shorter proofs.

[Wos96b] Wos, L., ''OTTER and the Moufang Identity Problem'', *J. Automated Reasoning* **17**, no. 2 (1996): 215-257.

[Wos96c] Wos, L., ''The Power of Combining Resonance with Heat'', *J. Automated Reasoning* **17**, no. 1 (1996): 23-81.

[Wos97a] Wos, L., ''Automating the Search for Elegant Proofs'', *J. Automated Reasoning*, accepted for publication, 1997.

[Wos97b] Wos, L., ''Experiments concerning the Automated Search for Elegant Proofs'', Technical Memorandum ANL/MCS-TM-221, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois, 1997.

[Wos97c] Wos, L., with Pieper, G. W., ''The Hot List Strategy'', Preprint ANL/MCS-P684-0897, Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, Illinois, 1997.