book7/ch03
created 12-22-2000
revised last 04-26-2001

CHAPTER 3

Attacking Hilbert's Twenty-Fourth Problem

delim @@

As the preceding chapter shows in various ways, the curious mind is superb at discovering or inventing challenging problems to solve, problems that include the seeking of a single axiom. As additional challenges, the well-known mathematician Hilbert offered in a 1900 lecture in Paris problems meriting study. Those familiar with Hilbert's problems would have said they numbered twenty-three—until quite recently, thanks to Rudiger Thiele (the University of Leipzig, Institute for the History of Science Sudhoff). Specifically, in a notebook in file 600, Thiele found a twenty-fourth problem of interest to Hilbert. This problem focuses on finding ''simpler proofs''.

That the problem was of substantial interest to Hilbert is attested to by his including it in an index of his work, by his focusing on it almost two decades later in a lecture in Zurich, and by his discussion of it near the end of his life. Thiele's exciting discovery brought us intense satisfaction, in part because of its close relation to the material featured in this chapter and in part because more than eight years ago we had initiated a study concerned with the use of an automated reasoning program to find simpler proofs, to automate the activity of proof refinement.

Whereas Chapter 2 featured Lukasiewicz's three-year hunt in the early 1930s that culminated in finding a 23-letter single axiom for two-valued sentential (or propositional) calculus, followed by our successful hunt that discovered a missing proof for that result, this chapter focuses on proof refinement. Indeed, as Chapter 2 reports, we captured the prize, a 200-step proof found on the included CD-ROM, later replaced with a 56-step proof presented in that chapter. The latter proof is a most satisfying refinement of the former.

Especially in view of the recent unearthing of Hilbert's twenty-fourth problem, one might naturally wonder how that 56-step proof was completed. We answer that question by focusing on methodologies for refining proofs (with respect to various measures), with the main emphasis on producing shorter proofs. (The distinction between proof shortening and proof contraction—the latter being the subject of Section 2.4—was discussed in Dialogue 2.) The problem of discovering shorter proofs appeals to the unaided researcher as well as to the researcher interested in the automation of logical reasoning, and its solution can have dramatic effects for circuit design and program synthesis. But how natural is the interest in such a refinement, and what is its place in history?

When one first reaches a sought-after goal—for example, proving a theorem, designing a circuit, implementing an algorithm—often one does not immediately announce the success. Indeed, a refinement stage commences: The proof is shortened to increase elegance, the circuit redesigned to add to efficiency, the algorithm reprogrammed to save computer time. Such refinements are not confined to one's own work. Meredith and Prior, for example, published an abridgment (shortening) of a proof of Lukasiewicz. They were not alone, and, as one learns from Thiele's discovery, they were significantly addressing a problem of interest to Hilbert.

The masters in proof finding, designing circuits, writing computer programs, and the like produce refinements seldom if ever by applying an algorithm. Instead, they can explicitly employ methodology based on knowledge, experience, and intuition. The reasoning program OTTER can also produce refinements, where methodology of various types is the key. In this chapter, we focus on diverse approaches to refinement and on some of the successes that were obtained.

The first success we feature might have brought Meredith especial pleasure; indeed, we present an abridgment of his proof of his single axiom (the following expressed in clause notation) for two-valued sentential calculus.

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

In addition to being a shorter proof, 38 deduced steps through the use of condensed detachment rather than 41 steps, our proof (given in Section 3.3) exhibits some unusual properties. First, where the Meredith proof contains 17 steps relying on double negation, terms of the form $n(n(t))$ for some term t, our proof contains but 13 such steps. Nevertheless, both proofs complete by deducing the three-axiom system of Lukasiewicz cited in Chapter 2, Section 2.1.

The second unusual property of the 38-step proof may indeed be more intriguing. Ordinarily and understandably, one would expect the proof of the first of these three axioms (that follow) to be the most difficult and the longest, and hence found last, which is true of Meredith's proof.

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))).

In addition, Meredith uses the second axiom in the proof of the first. In contrast—and here one encounters a property for which we certainly were not prepared—our proof proves the first of the three axioms sooner and with fewer steps than the second, and the proof uses the first axiom to prove the second.

Rather than immediately presenting our methodology that (in three runs) yields the 38-step proof, we first focus (in Section 3.1) on the various attempts we made. We choose to do so because we have been told that various researchers profit from viewing the paths that enabled us to extract from our experiments the tidy approach found in Section 3.2. Our choice makes convenient the discussion of the various components shared by some, but not all, of the refinement methodologies, and it also provides a framework for focusing on the connection between abridgment and finding a proof where none existed before.

If one wishes to immediately examine the heart of the approach that yielded the 38-step proof, one can turn to the end of Section 3.1. One might keep in mind, however, that the tidiness of the approach found in Section 3.2 implies in no way that we formulated the methodology very early in our study. Of a broader scope, although perhaps far from obvious, reliance on various refinement methodologies can prove most profitable in an attempt to answer an open question, settle a conjecture, or find a proof where the previous literature offered none.

3.1. A Basis for Refinement Methodologies

In addition to proof-length reduction, proof refinement can have as its concern maximal complexity of the deduced steps (measured in symbol count), maximum number of distinct variables among the deduced steps, total number of symbols, avoidance of some type of term, freedom from the use of thought-to-be-indispensable lemmas, and more. Many of these concerns are relevant to circuit design and program synthesis. For example, the avoidance of a type of term might correspond to the absence of some nested set of components or instructions, and the absence of a frequently used lemma might correspond to the absence of some substructure. As will be seen, often an interplay occurs between refinement considerations.

One might be focusing mainly on finding a shorter proof and reach that goal by emphasizing as a secondary concern a reduction in proof complexity. Especially in our research, one finds ever-present the interplay of such concerns as well as the interplay of the weapons OTTER offers. Most likely this explicit and constant interplay results directly from our reliance on an automated reasoning program; such interplay is perhaps far less evident in the work of an unaided researcher who can bring other assets to a study.

When, more than six years ago, we began our attack on finding a shorter proof of the Meredith single axiom, we were aware of the difficulty of succeeding. After all, he was well known for offering elegant proofs, and his success in discovering short proofs was evident in his publications. If, apparently, Meredith was unable to abridge (shorten) his 41-step proof, the discovery (if even possible) of a proof with 40 or fewer applications of condensed detachment would require vast experimentation, fortune, and even unorthodoxy.

The most direct approach (which we tried) was to use ancestor subsumption, a procedure that compares derivation lengths to the same conclusion and prefers the strictly shorter. We were merely checking the obvious and expecting failure. We were not disappointed.

The next phase of our attack focused on instructing OTTER to block the use of the 41 steps of Meredith's proof, one at a time, to force the program to travel at least a slightly different path in search of a proof. Although different proofs were found, none were of interest in the context of finding an abridgment of the Meredith result. Nevertheless, this move often succeeds—and here we come to a rather subtle occurrence that merits discussion—sometimes because there exists within the proof in hand a subproof in the weakest sense. To explain what we mean by ''subproof in the weakest sense'', as well as to provide appropriate background for understanding one of the elements our refinement methodologies share, we consider for convenience the type of proof (found in Chapter 2) OTTER and programs of its type produce.

In such a proof, all of the axioms and auxiliary input clauses are given before the deduced clauses. An example of an auxiliary clause is that (together with the inference rule hyperresolution) which captures the inference rule condensed detachment. For each deduced clause, explicitly listed are its parents as well as the inference rule used to yield it. The ensemble is the ''proof'', not just the set of clauses or their equivalent in some other logical or mathematical notation.

Let *A* be such a proof. Informally, the proof *B* is a weak subproof of the proof *A* if and only if all of its deduced steps are among the deduced steps of *A*. In other words, some of the deduced steps of *A* may be absent in *B*; the order may be changed; and, most important, the history of the deduced steps of *B* may be different. We are not concerned about the steps that precede the deduced steps (the axioms), although the concept could be appropriately broadened.

The point about the history of deduced steps is worth elaborating because it continually plays a key role in our success. In particular, with the objective of discovering a weak subproof, we often take a given proof, include resonators with small assigned values that correspond to the steps of the proof, assign a value to max_weight not much larger than the value assigned to the resonators, and instruct the program (in effect) to focus on its steps and possibly others that match them if viewed as resonators and no others. To see how success can occur, we select from a recent study—not pertinent to the Meredith result—that found a 39-step weak subproof of a 40-step proof.

In the 40-step proof, clause (35) condensed detached with itself yields clause (51), and in the 39-step proof clause (42) condensed detached with itself yields clause (45). Clauses (51) and (45) are identical, but their respective parents are not. Further, clause (35) is absent from the 39-step proof.

35 [hyper,1,26,30] P(i(x,i(y,i(z,y)))).
51 [hyper,1,35,35] P(i(x,i(y,x))).
42 [hyper,1,8,34] P(i(i(i(x,y),z),i(y,z))).
45 [hyper,1,42,42] P(i(x,i(y,x))).

In other words, with a possible shuffling of the steps, one could block the use of clause (35) when focusing on the 40-step proof and instead find a 39-step proof.

The exhibited phenomenon is not dependent on the use of condensed detachment. It can remind one of the occurrence in which different paths of reasoning end at the same point. However, as sometimes happens, when the process of blocking steps one at a time to obtain a shorter proof from a longer is iterated, with the result of discovering a weak subproof that is four or more steps shorter than the original proof, one can (as we did in the beginning) register substantial surprise. Indeed, the discovery that one can refine a given proof (with respect to length) by confining one's attention to its deduced steps, requiring no other formulas or equations, can be most unexpected. Blocking the steps of the proof in hand two at a time can also produce a weak subproof and can occasionally be a more effective move than blocking the steps one at a time.

We conjecture that this deliberate seeking of weak subproofs—perhaps not known to many—might have some interesting consequences for design, synthesis, and proof refinement. For design and synthesis, a success might prove most economical. Indeed, stories exist of disassembling some system, followed by an analysis that leads to a reassembling that omits some of the parts, where the new system works quite well. Throughout years of attempts, the seeking of weak subproofs, although not sufficient on its own, turned out to be an important component in our attempt to abridge the Meredith result.

As for the components of the methodology featured in Chapter 2, neither lemma adjunction nor avoidance of double-negation terms ever played a role in our attempt to abridge Meredith's 41-step proof of his single axiom. Use of the former would most likely interfere with proof refinement (with respect to length); use of the latter, although it has served well in this context on occasion, was simply never considered, perhaps because it seemed to place too much of a constraint on the study. However, still in the realm of term avoidance, for efficiency OTTER was instructed to purge any deduced conclusion in which *triple* negation occurred. We almost always take this action in studies of the type under discussion; an unaided researcher would likely do the same automatically and without thought, assuming that triple-negation terms served no purpose.

On the other hand, the resonance strategy, which was so important to the success reported in Chapter 2, did play a vital role in our various attempts, and it is featured in the tidy approach we give in the next section. Indeed, with essentially no thought, we were certain immediately that the finding of a proof with fewer than 41 steps would demand advice giving, would require the inclusion of resonators to direct OTTER's reasoning. The problem, therefore, shifted to which if any set or sets of resonators offered sufficient promise. As is always the case, whether seeking a first proof or seeking a proof refinement, the situation was further complicated by the knowledge that the other options and parameter values dramatically affect the likelihood of reaching the chosen goal.

To taste the complexity, one need only conduct (with a program) a set of experiments aimed at a single goal or contemplate the possible paths of reasoning that can be taken by an unaided mind. For an illustration, although we could use the hypothetical wanderings of a graduate student or even an expert—focusing on missteps, misplaced emphases, and misfortune—we instead rely on the explicit offerings of a program like OTTER. (No doubt correspondents to these offerings exist for the researcher, but the degree to which explicit choices are made is unclear.) Each of these offerings played a continuous role in our thousands of experiments concerned with abridgment.

Interplay of Program Offerings

First is the concern for the complexity of deduced-and-retained conclusions measured in symbol count. Whereas some proofs can be completed with formulas or equations whose complexity does not exceed 20, other proofs may require a far, far greater complexity. Indeed, in an early 1990s study of groups in which the 19th power of x is the identity e (groups of exponent 19), equations were needed whose complexity did in fact exceed 700; in other words, we were forced to assign the max_weight a value in excess of 700, where the max_weight places an upper bound on the complexity of deduced-and-retained conclusions. When proofs with such long equations were shown to the well-known set-theorist Ken Kunen, he commented that notation to sharply reduce their length would help or perhaps, for readability, be necessary. Mathematicians and logicians continually and wisely introduce notation and definitions that certainly add to readability but, more important, appear to dramatically increase reasoning power. In contrast, automated reasoning programs cope poorly with definition expansion and definition contraction. This inability (in effect) poses a research question for those interested in automation.

Note that OTTER measures complexity in terms of symbol count unless instructed otherwise, which can be done with resonators or with weight templates such as the following.

weight(n($(1)),2).

Inclusion of the given template will penalize deduced conclusions according to the number of occurrences of the function n (here for negation). We often include this template when permitting the use of double negation but wishing to discourage the focus on conclusions in which negation occurs frequently.

In our intensive study whose goal was a shortening of the Meredith 41-step proof, we relied on various assignments to max_weight. In some experiments, we chose a very small value (such as 4) to force the program to focus mainly on conclusions that matched an included resonator. In other experiments, we chose a large value (such as 48) to enable the program to retain complex formulas that might lead to a breakthrough.

A second complicating factor for experimentation rests with OTTER's mechanism for choosing where next to focus to initiate applications of inference rules. OTTER ordinarily bases its choice of the assigned complexity of retained conclusions not yet used as the focus of attention. However, by means of OTTER's offering of the *ratio strategy*, one can instruct the program to intermingle its choice of focus, sometimes based on complexity, and sometimes based on breadth first (first-retained first-chosen). Specifically, if the value *k* is assigned to the pick_given_ratio parameter, then the program will choose *k* clauses based on complexity, 1 based on breadth first, *k*, 1, and so forth. The chosen value has in some studies proved crucial.

For example, in studies of Boolean algebra aimed at proof refinement (with respect to length), a value of 10 and a value of 12 each failed to produce progress, whereas a value of 11 did. In other words, occasionally the window that leads to an important result is extremely narrow. This phenomenon is by no means limited to automated reasoning, as can be attested to by engineers, chemists, physicists, and other scientists.

As one might predict, the interplay of the ratio strategy with the upper bound placed on the complexity of deduced-and-retained clauses appears to defy a complete analysis. If the max_weight is too small, then a vital conclusion might be purged upon deduction; if too large, then the efficiency of the program may be sharply reduced. If the value assigned to the pick_given_ratio is too small, the program will spend too much of its time on conclusions retained early, and if the value is too large, a complex conclusion retained rather early might not ever be chosen to drive the program's reasoning. Our heavy experimentation focused on numerous combinations of the two values. Of course, even if wishing to do so, an expert would seldom attack by hand a question in this experimental manner. However, if a program like OTTER were added to the effort, then such a wish could easily be granted. Indeed, we use two additional programs with OTTER to automate some of our experimentation. The first, otter-loop, automates our blocking of steps of a proof one at a time, and a variation of it, otter-loopn, automates blocking steps more than one at a time. The second program, super-loop, has OTTER try various assigned values to the parameters in diverse combinations, where the choices are made by the user. We know of no way as yet to automate experiments in which the sets of resonators vary. If transported to the year 2001, would Hilbert or Lukasiewicz or Meredith or Tarski, for example, find the use of OTTER irresistible?

A third complicating factor in our experimentation concerns the use of the *hot list strategy* and its extension the *dynamic hot list strategy*, each of which merits some discussion. When the program decides to retain a new conclusion, *before* another conclusion is chosen from list(sos) as the focus of attention to drive the program's reasoning, the hot list strategy causes the new conclusion to initiate applications of the inference rules in use, with the remaining hypotheses (or parents) that are needed all chosen from the hot list. Members of the hot list, whether input or adjoined during a run, are not subject to back demodulation or back subsumption; in other words, clauses in the hot list are never rewritten with the discovery of new demodulators (rewrite rules), nor are they purged because of being captured by a newly retained clause.

If one wishes the program *during* the run to adjoin new members to the hot list, one uses the dynamic hot list strategy, which McCune formulated as an extension of the hot list strategy). If that is the choice, then one assigns an integer, positive or negative, to the dynamic_heat_weight parameter with the intention that conclusions that are retained and that have a *pick_given weight* (complexity) less than or equal to the assigned value will be adjoined to the hot list. (A clause, or its equivalent, 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.) The dynamic_heat_weight assignment places an upper bound on the pick_given weight of clauses that can be adjoined to the hot list during the run.

When the heat parameter is assigned the value 1, conclusions that result from consulting the hot list (whether one is using the hot list strategy or the dynamic hot list strategy) have *heat level* 1. (By definition, input clauses have level 0, and a deduced clause has a level one greater than the maximum of the levels of its parents.) If the program decides to retain a conclusion of heat level 1 and if the (input) heat parameter is assigned the value 2, then *before* another conclusion is chosen as the focus of attention, the heat-level-1 conclusion is used to initiate the search for conclusions of heat level 2 (with the hypotheses needed to complete the corresponding application of the inference rule chosen from the hot list). The heat parameter can be assigned to whatever positive integer the researcher chooses, with the objective of placing corresponding emphasis on the immediate use of the members of the hot list. If the parameter is assigned the value 0, then the hot list strategy will not be used.

With the static or dynamic hot list strategy, one is able to instruct a program to visit, revisit, re-revisit, and the like some chosen set of hypotheses immediately upon retention of a new conclusion. To illustrate why such visiting is advised in some cases, consider the program's retention of a quite complex clause, one, say, of length 40 (symbols). Assume that this new conclusion fails to match any included resonator or included template that might apply to a subexpression. Therefore, 40 is the value assigned to its complexity. If the cited conclusion is drawn after 1 or more CPU-hours, The likelihood is great that it will not be chosen for a long, long time to drive the reasoning—if ever. Now assume that the item under discussion, if considered with an input hypothesis that is placed in the hot list, would yield the vital step in a sought-after proof. If the hot list strategy were in use, and if the value assigned to the input heat parameter was 1 or greater, the vital step would be almost immediately obtained.

A glance at some of the proofs found in the literature concerning single axioms reveals the presence of such steps. Closer inspection reveals, sometimes, the presence of a *sequence* of such steps, where the first involves the single axiom and a deduced conclusion, the second involves the first step and the axiom, and the *n* involves the *n*-1st and the axiom. If the input heat parameter were assigned the value *n*, such a sequence would be easily within reach of a program. Our experimentation relied on various assignments to the heat parameter.

Finally, with the other options and parameters put aside, there is the question of which and how many sets of resonators to rely upon and which value or values to assign to the resonators. Whatever the value or values, of course, the value assigned to max_weight must be equal to or greater than the maximum of such values. Because the smaller the assigned value to a resonator, the higher the priority a matching conclusion is given for being chosen as the focus of attention, we were faced with the choice of which resonator sets (if more than one set were present) to give preference. Although not an obviously effective move, we often rely on what might be termed layered resonators, a set of resonator sets with members of the first set assigned a value, say, of 1, members of the next set a value of 2, and the like. Such a move is designed to give preference to one class of formulas or equations over another. We experimented with many sets, various values, and diverse orders of the resonator sets.

Two Significant Results

These numerous experiments (only barely sketched) did lead to two results that in turn will be seen to play a key role in the sought-after abridgment. The first concerns a shortening of a proof found by Deepak Kapur in the early 1990s. Kapur used OTTER to discover a 63-step proof that derives the three-axiom system of Lukasiewicz from the Meredith single axiom, and—here is the delightful aspect—the proof requires the use of no formulas in which seven or more distinct variables occur; his proof contains only two formulas relying on six distinct variables. Meredith's 41-step proof contains two formulas in which seven distinct variables occur. In the past two years, we used OTTER (and the various methodologies featured in this book) to discover a 49-step proof relying on no more than six distinct variables. The study of such proof refinements that focus on the number of distinct variables on which the deduced steps rely is made easy with OTTER by assigning the desired value to max_distinct_vars. The 49-step proof contains one step requiring six distinct variables; as noted, Kapur's 63-step proof contains two formulas of this type. For our success with abridging the Meredith 41-step proof, we relied on resonators corresponding to a 50-step proof of the six-variable class, a proof found months before finding the cited 49-step proof. We chose the 50-step proof over the 49-step proof based on intuition.

The second significant result yielded by our experimentation was a 51-step, seven-variable proof free of double negation. Its discovery relied again on our various methodologies. (The study of proof refinement centering on the avoidance of some type of term is greatly assisted by the use of appropriate demodulators to block the retention of unwanted conclusions and, alternately, by the use of weight templates that assign a weight higher than allowed by max_weight.) A set of resonators corresponding to its proof steps played a vital role, each assigned a common value, 1 greater than that assigned to the cited 50 resonators. In other words, we employed a layered-resonator approach. This layered-resonator approach was supplemented with the use of certain demodulators (rewrite rules) to block retention of conclusions proven to interfere with obtaining the desired abridgment. And, as the next section discusses, we were able to extract (from our years of experimentation that culminated in successes) a neat three-run approach.

Those years demonstrated yet again that the space of conclusions that can be drawn, as well as the space of completable proofs, each present a labyrinth, a maze to be solved. Indeed, the search for a desired proof is reminiscent of wandering through a cave with tunnels that end abruptly in a proof of no interest, where other tunnels continue for many CPU-hours, getting narrower and narrower. One learns from such studies that the space of proofs is indeed rich—far, far larger than one might surmise in graduate school and even beyond. The problem is to find a means to mine the gold, which may take the form of a first proof (as was the case in Chapter 2) or (as in this chapter) the form of an abridgment of a proof of a master.

Especially from exposure to the efforts of a program like OTTER, one also learns why neither logic nor mathematics easily lends itself to an algorithmic approach, whether (at one end of the spectrum) the task is settling a conjecture or (at the other end) the task is proof refinement. In place of algorithms is the use of methodologies, certainly concrete and explicit when a program is involved. Just as logic and mathematics continue to accrue theorems as part of their growth, the methodologies we give continue to grow and expand, both in nature and in application. Indeed, the mind naturally seeks (as part of hard-problem solving) to extend the use of strategies originally formulated for proof refinement to include the context of answering open questions. The evidence we offer in this volume suggests partial success in this regard.

3.2. A Tidy Approach for Abridging the Meredith Proof

To set the stage, we begin with a brisk review of the highlights of the preceding section. The intense experimentation with the Meredith single axiom paid fine dividends, especially that focusing on finding a proof relying on a limit of six distinct variables and that focusing on finding a double-negation-free proof. The successes in each effort are (at least to us) interesting in themselves in that Meredith's proof contains two formulas in which seven distinct variables are present and also relies on seventeen formulas in which double negation is present. Perhaps more important, among the proofs the separate efforts discovered, two enabled the abridging of the 41-step proof. From our experiments that yielded the abridgment, we extracted a three-run approach, which we now present.

The first of the three runs relies on two sets of resonators. The first set corresponds to a six-variable 50-step proof showing that the Meredith axiom suffices to derive the three-axiom system of Lukasiewicz, a proof containing 19 steps in which double negation occurs. Each of the resonators is assigned the value 0. The second set of resonators corresponds to a seven-variable 51-step proof in which double negation is absent and in which two of its steps exhibit the use of seven distinct variables, also deriving the three-axiom system of Lukasiewicz. Each resonator in this set is assigned the value 1, thus giving retained conclusions that match a resonator in the first set preference (for being chosen to initiate applications of condensed detachment) over those that match a resonator in the second but not in the first. Retained conclusions that match a resonator neither in the first set nor in the second are given less preference.

A value of 36 was assigned to max_weight to permit the program to retain numerous conclusions whose weight in symbol count does not exceed 36. The intention was to give the program the opportunity of using many, many conclusions that do not match a resonator. A value of 3 was assigned to the pick_given_ratio, instructing the program to choose for driving the reasoning three clauses by weight (complexity), one by breadth first, then three, then one, and the like. The program was thus encouraged to occasionally focus on clauses retained reasonably early, a move that might be important because complex clauses might otherwise be delayed for an interminable amount of CPU time before being chosen to drive the reasoning. A value of 7 was assigned to max_distinct_vars to give the program latitude in this parameter. Also, double-negation terms were deemed acceptable. As a result of our choices, all of the steps in the 50-step proof and in the 51-step proof were acceptable for possible inclusion in the sought-after proof.

In addition to the demodulators that were included to ensure that triple-negation terms would not be present, one other demodulator (rewrite rule) was included, a demodulator to prevent the program from retaining the conclusion $i(x,x)$. The cited formula occurs as the seventh step in the 50-step proof. Blocking its presence might block an entire path of reasoning, even though different pairs of formulas can yield with condensed detachment the same conclusion. Experiments had demonstrated that progress would occur if this formula were purged upon generation.

To have the program provide possibly important assistance in finding shorter proofs, ancestor subsumption was employed. When that procedure is used, one should also employ back subsumption, a procedure that purges an already-retained conclusion when a newly retained conclusion properly subsumes it (is strictly more general). The use of ancestor subsumption is in no way a panacea in the context of finding shorter and ever-shorter proofs. Unfortunately and not so obvious, the aphorism ‘‘shorter subproofs do not necessarily a shorter total proof make’’ nicely summarizes the complexity of seeking proof abridgment.

For a small taste of what can go wrong, consider the following example. Let **P** be a 30-step proof of the theorem **T**; let *A* be the twentieth step; let the proof **S** of *A* within **P** consist of various axioms and the fifteenth through the twentieth step inclusive. The length of the proof **S** is 6, and **S** is a subproof of **P**. Now let **R** be a proof of *A* of length 5 such that the only deduced step present in both **R** and **S** is *A*. If all of the steps of **S** are used as parents for steps later than *A* in **P**, and if the first four steps of **R** are of no use in this context, then replacing **S** by **R** will in many cases require for a proof of **T** a sequence of steps whose length is strictly greater than 30. One thus sees how the aphorism applies, whether the shorter subproof is obtained by hand or by a program through the use of ancestor subsumption.

Typically, when ancestor subsumption is in use and the program is allowed to complete one proof after another, the lengths do not vary wildly, getting shorter, longer, shorter, longer, and the like. Indeed, in the context of proof length, once an interesting proof has been completed and the proof following it is rather longer, one can expect to encounter little of value by letting the program continue. Such, however, was not the case with the run under discussion. In particular, the first five proofs discovered were, respectively, of length 51, 70, 45, 55, and—the prize—40. The Meredith proof had been abridged.

Naturally, one wonders about the relation of the 40-step proof to each of the proofs (respectively of lengths 50 and 51) used as resonators. Of the forty steps, eleven are not among the fifty, and thirteen are not among the fifty-one. Further, if one makes the corresponding comparisons at the resonator level (to determine how many of the forty steps match a resonator), the same two figures are obtained. Clearly, the two sets of resonators do not tell the full story; extra latitude was required, given by a decent max_weight (36) and by a reasonable ratio (3). But did the two resonators sets tell any story of significance?

One experiment sheds light on that question. When the 50 resonators corresponding to a six-variable proof of length 50 were replaced by 49 resonators corresponding to a six-variable proof of length 49, each deducing the Lukasiewicz three-axiom system, no prize was won; no proof shorter than Meredith’s was discovered.

If one enjoys open questions to attack, the following two may offer appeal.

Open question: Does there exist a six-variable proof of length strictly less than 49 (applications of condensed detachment) that derives some known axiom system for two-valued sentential calculus from the Meredith single axiom?

Open question: With condensed detachment as the only rule of inference, does there exist a six-variable proof free of double negation that derives from the Meredith single axiom some known axiom system for this area of logic? Or, can one show that such a proof cannot be found, especially when one is constrained to focus on a well-known axiom system as the target?

We did not terminate our research when we obtained the 40-step proof. Instead, because of the excitement engendered by the success with abridging a result of a master who had published abridgments of results of others, and because of the wish to test one of our standard methodologies for proof refinement in the context of length, we attempted to seek a more impressive abridgment.

The cited methodology centers on using a newly discovered and shorter proof in the context of the resonance strategy. In particular, the second of the three runs that provide a tidy approach relies on resonators that correspond to the 40-step proof discovered in the first run and ignores the use of any other resonators. We also included an additional demodulator, the following, to block the corresponding formula, knowledge gained from earlier experiments.

(P(i(x,i(i(i(y,i(n(n(z)),z)),u),u))) = junk).

Additional demodulators are included that eventually cause the unwanted formula to demodulate to $T (**trueafR); OTTER automatically discards clauses that have the form $T. The given demodulator does not correspond to any step of the 40-step proof. In all other respects, the input for run 2 was identical to that for run 1.**

The second run yielded two proofs, the first of length 40, the second of length 39. Very often, with this methodology, the program first reproduces the proof guiding the search, and sometimes nothing else. Had the additional demodulator corresponded to a step of the 40-step proof, one should expect (but not with certainty) that the 39-step proof consists of formulas all of which are among the 40. In the case under discussion, one of the steps of the 39-step proof is not present in the 40-step proof.

For the third run, resonators corresponding to the 39-step proof of run 2 were used and no others; no demodulators were used to block formulas, other than blocking triple-negation conclusions; all other aspects were unchanged. As expected, the first proof completed was the known 39-step proof; the second was a 46-step proof; the third—most satisfying to us—was a 38-step proof. The 38-step proof contains one formula not present in the 39-step proof and two not present in the 40-step proof. (Our efforts to find a proof shorter than length 38 yielded nothing of interest.)

Often, after a satisfying advance occurs and resonators from the new and shorter proof are relied upon, one is advised in many of the methodologies to remove all demodulators that led to progress. Otherwise, additional advances may be missed because of the search being too constricted. Of course, more than occasionally one or more of the dismissed demodulators must be reinstalled to make further progress.

For an open question, we offer the following.

Open question: Can one find a proof of strictly less than 38 applications of condensed detachment showing that the Meredith formula is a single axiom for two-valued sentential calculus. Or can one show that such a proof does not exist? Although in most of our research, proving the three-axiom system of Lukasiewicz seems to be easiest and shortest, perhaps another target axiom system will serve one better.

3.3. A Most Intriguing Proof

Although we have commented earlier on some of the properties of the 38-step proof, closer inspection may prove profitable.

An Abridgment of Meredith's Proof

```
----- Otter 3.0.6, Feb 2000 -----
The process was started by fitelson on lythande.mcs.anl.gov, Mon Dec 11 19:06:51 2000
The command was "/homes/fitelson/otter-mccune/otter".  The process ID is 7803.

-----> EMPTY CLAUSE at 2495.66 sec ----> 217226 [hyper,2,216598,216969,79]
$ANS(Luka123).
```

Length of proof is 38. Level of proof is 25.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] -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))) |
$ANS(Luka123).
3 [] P(i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x)))).
9 [hyper,1,3,3] P(i(i(i(i(x,y),i(z,y)),i(y,u)),i(v,i(y,u)))).
11 [hyper,1,3,9] P(i(i(i(x,i(n(y),z)),u),i(y,u))).
13 [hyper,1,3,11] P(i(i(i(x,x),y),i(z,y))).
15 [hyper,1,13,13] P(i(x,i(y,i(z,z)))).
31 [hyper,1,3,15] P(i(i(i(x,i(y,y)),z),i(u,z))).
37 [hyper,1,3,31] P(i(i(i(x,y),z),i(y,z))).
65 [hyper,1,3,37] P(i(i(i(i(n(x),n(y)),x),z),i(y,z))).
68 [hyper,1,37,3] P(i(x,i(i(x,y),i(z,y)))).
78 [hyper,1,3,65] P(i(i(i(x,y),n(i(n(y),n(z)))),i(z,n(i(n(y),n(z)))))).
79 [hyper,1,65,37] P(i(x,i(n(x),y))).
85 [hyper,1,37,68] P(i(x,i(i(i(y,x),z),i(u,z)))).
105 [hyper,1,68,78] P(i(i(i(i(i(x,y),n(i(n(y),n(z)))),i(z,n(i(n(y),n(z))))),u),i(v,u))).
116 [hyper,1,37,79] P(i(x,i(n(i(y,x)),z))).
122 [hyper,1,11,79] P(i(x,i(n(i(y,i(n(x),z))),u))).
151 [hyper,1,3,85] 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))).
171 [hyper,1,3,105] P(i(i(i(x,y),i(z,u)),i(i(n(u),n(n(y))),i(z,u)))).
205 [hyper,1,37,116] P(i(x,i(n(i(y,i(z,x))),u))).
302 [hyper,1,3,151] 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))))).
319 [hyper,1,171,105] P(i(i(n(x),n(n(x))),i(y,x))).
325 [hyper,1,171,3] P(i(i(n(i(x,y)),n(n(z))),i(i(z,y),i(x,y)))).
409 [hyper,1,302,68] P(i(x,i(i(i(y,z),u),i(i(i(z,v),i(n(u),n(y))),u)))).
418 [hyper,1,37,319] P(i(n(n(x)),i(y,x))).
432 [hyper,1,409,409] P(i(i(i(x,y),z),i(i(i(y,u),i(n(z),n(x))),z))).
484 [hyper,1,432,3] P(i(i(i(x,y),i(n(i(i(x,z),i(u,z))),n(i(i(i(z,v),i(n(w),n(u))),w)))),i(i(x,z),i(u,z)))).
490 [hyper,1,484,205] P(i(i(x,i(x,y)),i(z,i(x,y)))).
491 [hyper,1,484,122] P(i(i(x,y),i(n(i(x,z)),y))).
507 [hyper,1,490,418] P(i(x,i(n(n(y)),y))).
583 [hyper,1,68,507] P(i(i(i(x,i(n(n(y)),y)),z),i(u,z))).
621 [hyper,1,490,583] P(i(x,i(i(i(y,i(n(n(z)),z)),u),u))).
216067 [hyper,1,621,621] P(i(i(i(x,i(n(n(y)),y)),z),z)).
216096 [hyper,1,491,216067] P(i(n(i(i(i(x,i(n(n(y)),y)),z),u)),z)).
216097 [hyper,1,432,216067] P(i(i(i(x,y),i(n(x),n(i(z,i(n(n(u)),u))))),x)).
216143 [hyper,1,325,216096] P(i(i(x,y),i(i(i(z,i(n(n(u)),u)),n(n(x))),y))).
216261 [hyper,1,216143,216143] 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))).
216467 [hyper,1,3,216261] P(i(i(i(i(i(x,i(n(n(y)),y)),n(n(z))),u),v),i(i(z,u),v))).
216598 [hyper,1,216467,3] P(i(i(x,y),i(i(y,z),i(x,z)))).
216600 [hyper,1,216598,216598] P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))).
216969 [hyper,1,216600,216097] P(i(i(n(x),x),x)).

The following comparison of the given 38-step proof with Meredith's 41-step proof perhaps sheds much light on what made the abridgment possible. To begin with, sixteen formulas are present in the 38-step proof that are absent in the 41-step proof. The 38-step proof contains thirteen formulas in which double-negation terms are present; the 41-step proof contains seventeen such steps. Whereas the order in the 41-step proof of the deductions of the three Lukasiewicz axioms is the third, the second, and the first, the order in the 38-step proof is the third, the first, and the second. In the 41-step proof the cited subproofs in order are of lengths 20, 26, and 40; in the 38-step proof, they are, respectively, 8, 35, and 38.

One sees that the subproof of the second Lukasiewicz axiom is longer in the 38-step proof than in the 41-step proof, but the other two subproofs have reduced lengths. Whereas the 41-step proof uses the second Lukasiewicz axiom in its subproof of the first, the situation is reversed in the 38-step proof. Finally, the 38-step proof is *compact*, meaning the entire proof is also a proof of one of the members of the target axiom system. The 41-step proof comes as close to being compact as possible without being compact. The level of Meredith's proof is 30, whereas the level of the given 38-step proof is 25.

In addition, the 38-step proof contains two formulas (the 18th and the 24th) whose complexity measured in symbol count is 34. The 41-step proof contains one formula (the 12th) of complexity 34. Where *size* of a proof is measured in terms of the symbols present in the deduced steps (counting the predicate symbol, but not parentheses or commas), the 41-step proof has size 694, the given 38-step proof has size 625, and the 38-step proof of level 24 has size 634. As for relying on the maximum number of distinct variables, whereas the 41-step proof contains two formulas (the 10th and 21st) requiring seven distinct variables, the 38-step proof contains one such formula (the 15th). Therefore, at least on the surface, one might have more success with attempting to abridge the cited 38-step proof rather than the 41-step proof. As for a relation to the work of Lukasiewicz, three steps of the 41-step proof are among theses 4 through 71, and two steps of the 38-step proof are among those theses. In summary, the abridgment we offer in this section is dramatically different from Meredith's proof.

Some observations are in order that show how the 38-step abridgment was actually discovered. One of our methodologies for refinement in the context of proof length has the user of the program take actions that force the program to find a sharply different proof from that in hand. Although more details will be given later, a small taste of possibly profitable moves provide some insight.

One can instruct the program to avoid using a sequence of four or more steps in the middle of the proof in hand. One can prevent the program from using some thought-to-be-key lemma. One can designate a class of terms as undesirable and prevent the program from retaining a new conclusion if one of the undesirable terms is present in it. One can take two quite different proofs and, rather than taking their intersection to be used as resonators, use as resonators their symmetric difference. The idea is to have the program explore a part of the search space perhaps not visited before, by itself or by an unaided researcher.

Although at this time we cannot give any explanation, we conjecture that the likelihood of refining a proof with respect to length is increased when the difference between the level and the length is greater rather than smaller. Based on that conjecture, if one intends to attack the question of the existence of a proof of length strictly less than 38, one might begin by focusing on a 38-step proof of level 24 that we discovered before completing the given 38-step proof. That (to us better in some senses) level-24 proof is found on the included CD-ROM; it does share many properties with the 38-step level-25 proof. Unaided researchers seem to prefer focusing on proofs of lower level, whereas the program cares not at all.

3.4. Other Methodologies, Other Types of Refinement

Although—amazingly enough—success resulted with the theorem featured here, the nature of the methodology we discuss next strongly suggests that its likelihood of success is small, especially because of certain additional constraints. That theorem concerns the following shortest single axiom (found and proved to be such by Lukasiewicz in 1936) for the implicational fragment of two-valued sentential calculus (which is the subtheory of the entire two-valued calculus, featured in Chapter 2, where the only connective is that for implication), expressed in clause notation.

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

His proof of its sufficiency completes by deriving the following three-axiom system supplied in 1921 by Tarski and Bernays.

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

Meredith and Prior published a ‘‘very slight abridgment’’ of the Lukasiewicz proof; their proof consists of 33 applications of condensed detachment. They were able to eliminate one step of Lukasiewicz's proof, yielding a 33-step condensed detachment proof, four of whose steps are not present in the reconstructed Lukasiewicz 34-step proof.

We were eager to attempt a climb greater than Mount Everest, enthralled about trying to find an abridgment of the Meredith-and-Prior abridgment. Therefore, we began a set of experiments. We did find a 34-step proof, essentially that of Lukasiewicz. (Actually, the Lukasiewicz proof consists of 28 steps, but not all obtained with condensed detachment, which explains in part the use of the term abridgment by Meredith and Prior.) Our 34-step proof contains all but the last step of the 28-step proof, that not obtained with condensed detachment (the only inference rule allowed). We used resonators corresponding to the 34-step proof, blocked with demodulation various steps, and applied other techniques we ordinarily use for proof shortening.

Although near the end of our effort we still had not succeeded in finding a proof shorter than length 33 (applications of condensed detachment) of the entire Tarski-Bernays system, we did find a 30-step proof of the third member of this system. We then made a rather wild conjecture: Perhaps one could extend this 30-step proof by ‘‘cramming’’ in the remaining two members of the target system, thus completing a 32-step proof. But how could one instruct OTTER to make such an attempt?

The answer was soon forthcoming: Formulate and use a new methodology (the following), which just might work even though it was clearly naive. First, we placed in the initial set of support the steps of the 30-step proof to be used as lemmas together with the Lukasiewicz shortest single axiom. Second, we used thirty-two resonators with an assignment of 0 to each—one corresponding to the first Tarski-Bernays axiom, one corresponding to the second axiom, and thirty corresponding to the 30-step proof of the third axiom—and used the assigned value of 4 to max_weight. Third, we instructed the program to make a breadth-first search, with the command set(sos_queue). Our hope was that the first and second of the Tarski-Bernays axioms would each be deduced in one step. In other words, to succeed in completing a 32-step proof, there must exist pairs of hypotheses among the thirty-one hypotheses (the Lukasiewicz single axiom and the thirty adjoined steps of the proof of the third axiom of Tarski and Bernays) such that two applications of condensed detachment would yield, respectively, the first and second axioms. To put it another way, perhaps the program could cram the needed two axioms into a 32-step proof, 30 steps of which were already used.

OTTER was victorious. Condensed detachment applied to the ninth step of the 30-step proof used as both major and minor premiss yielded the first axiom. Its application to the twentieth-eighth as the major premiss and the twenty-sixth as minor premiss yielded the second axiom. The program had been able to cram into a proof of length 32 (the following) the 30-step proof of the third axiom as well as deductions of the first and second axioms.

A 32-Step Abridgment of the Meredith-and-Prior Proof

```
----- Otter 3.0.5b, March 1998 -----
The process was started by wos on soot.mcs.anl.gov, Wed May 24 16:24:24 2000
The command was "otter". The process ID is 5369.

-----> EMPTY CLAUSE at   0.23 sec ----> 82 [hyper,34,53,77,79]
$ANS(TARSKI_BERNAYS).

Length of proof is 32.  Level of proof is 29.

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

33 [] -P(i(x,y)) | -P(x) | P(y).
34 [] -P(i(p,i(q,p))) | -P(i(i(i(p,q),p),p)) | -P(i(i(p,q),i(i(q,r),i(p,r)))) |
$ANS(TARSKI_BERNAYS).
35 [] P(i(i(i(x,y),z),i(i(z,x),i(u,x)))).
44 [hyper,33,35,35] P(i(i(i(i(x,y),i(z,y)),i(y,u)),i(v,i(y,u)))).
45 [hyper,33,35,44] P(i(i(i(x,i(y,z)),i(i(u,y),i(v,y))),i(w,i(i(u,y),i(v,y))))).
46 [hyper,33,45,35] P(i(x,i(i(i(y,z),y),i(u,y)))).
47 [hyper,33,46,46] P(i(i(i(x,y),x),i(z,x))).
```

48 [hyper,33,35,47] P(i(i(i(x,y),i(y,z)),i(u,i(y,z)))).
49 [hyper,33,35,48] P(i(i(i(x,i(y,z)),i(u,y)),i(v,i(u,y)))).
50 [hyper,33,35,49] P(i(i(i(x,i(y,z)),i(u,i(z,v))),i(w,i(u,i(z,v))))).
51 [hyper,33,50,35] P(i(x,i(i(i(y,z),u),i(z,u)))).
52 [hyper,33,51,51] P(i(i(i(x,y),z),i(y,z))).
53 [hyper,33,52,52] P(i(x,i(y,x))).
55 [hyper,33,52,35] P(i(x,i(i(x,y),i(z,y)))).
56 [hyper,33,35,55] P(i(i(i(i(i(x,y),z),i(u,z)),x),i(v,x))).
57 [hyper,33,35,56] P(i(i(i(x,y),i(i(i(y,z),u),i(v,u))),i(w,i(i(i(y,z),u),i(v,u))))).
58 [hyper,33,35,57] P(i(i(i(x,i(i(i(y,z),u),i(v,u))),i(w,y)),i(v6,i(w,y)))).
59 [hyper,33,35,58] P(i(i(i(x,i(y,z)),i(u,i(i(i(z,v),w),i(v6,w)))),i(v7,i(u,i(i(i(z,v),w),i(v6,w)))))).
60 [hyper,33,59,35] P(i(x,i(i(i(y,z),i(u,v)),i(i(i(z,w),v),i(u,v))))).
61 [hyper,33,60,60] P(i(i(i(x,y),i(z,u)),i(i(i(y,v),u),i(z,u)))).
62 [hyper,33,61,35] P(i(i(i(x,y),i(z,u)),i(i(x,u),i(z,u)))).
63 [hyper,33,62,55] P(i(i(x,i(y,z)),i(i(i(x,u),z),i(y,z)))).
64 [hyper,33,63,35] P(i(i(i(i(i(x,y),z),u),i(v,x)),i(i(z,x),i(v,x)))).
65 [hyper,33,35,64] P(i(i(i(i(x,y),i(z,y)),i(i(i(y,u),x),v)),i(w,i(i(i(y,u),x),v)))).
66 [hyper,33,65,65] P(i(x,i(i(i(i(y,z),u),i(v,z)),i(i(i(z,w),v),i(y,z))))).
67 [hyper,33,66,66] P(i(i(i(i(x,y),z),i(u,y)),i(i(i(y,v),u),i(x,y)))).
68 [hyper,33,61,67] P(i(i(i(i(x,y),z),i(u,y)),i(i(i(y,v),x),i(u,y)))).
69 [hyper,33,67,68] P(i(i(i(i(x,y),z),i(i(y,u),v)),i(i(v,y),i(x,y)))).
70 [hyper,33,68,62] P(i(i(i(i(x,y),z),u),i(i(u,y),i(x,y)))).
71 [hyper,33,69,64] P(i(i(i(x,y),z),i(i(i(y,u),z),z))).
73 [hyper,33,64,71] P(i(i(x,y),i(i(i(x,z),y),y))).
75 [hyper,33,73,73] P(i(i(i(i(x,y),z),i(i(i(x,u),y),y)),i(i(i(x,u),y),y))).
76 [hyper,33,70,75] P(i(i(i(i(i(x,y),z),z),u),i(i(x,z),u))).
77 [hyper,33,75,71] P(i(i(i(x,y),x),x)).
79 [hyper,33,76,70] P(i(i(x,y),i(i(y,z),i(x,z)))).

Two steps of this proof are not present in the 33-step proof of Meredith and Prior, and four of its steps are not present in the Lukasiewicz 34-step proof.

In view of the two given successes with finding shorter proofs than were found by one of the masters interested in such matters, one might naturally ask for an intuitive discussion of what gets in the way of such discoveries. For example, one might conjecture that a breadth-first search is a more than viable approach. Actually, in the majority of cases, such a search is impractical because, as the levels are traversed, one can encounter an exponential growth in the size of conclusions to be examined. Some of the shortest proofs we have found have a level greater than 20 (such as our abridgment of the Meredith 41-step proof), which forces such proofs to be out of reach of a breadth-first search. Also, by the way, one can find examples where the level of a first proof is less than that of the second, but the length of the second is less than that of the first, and the differences can be most impressive. Other obstacles exist to the refinement of proofs in the context of length.

In a somewhat unusual sense, another obstacle rests with encountering some type of cul de sac. We say ''unusual'' because the term cul de sac brings to mind a dead end, a path that permits no additional steps forward. Of course, such a case can occur, a case in which no additional conclusions can be drawn of any type, often because of the choice of parameter values and options. But a rather different type of cul de sac is typically the problem.

Indeed, the attack often encounters some case in which all newly retained conclusions simply produce no advance. For one example, the methodology that emphasizes the use of a given proof and blocks its steps one at a time often yields shorter and shorter proofs until any additional blockage results in a longer proof. And sometimes this methodology yields no advance. To escape this type of cul de sac, occasionally the blocking of steps two at a time succeeds.

For a different example, the program can get trapped within a set of conclusions most or all of which have a rather small weight, say, twelve or fewer symbols. In particular, the program chooses one such conclusion to drive its reasoning and then deduces more conclusions of the same weight or smaller. The program can remain in such a trap for an inordinate amount of CPU time. When encountered, this type of cul de sac can prevent the program from answering an open question or settling a conjecture. Use of the ratio strategy can be of some assistance, and use of resonators that assign small weight to complex conclusions (those of high weight if measured in symbol count) can provide an escape route.

A third type of cul de sac takes the form of a vast desert of conclusions of various types and classes and weights. The program wanders throughout this barren land, finding no proof of interest or, in the case of an open question, no proof of any type. A close look sometimes unearths a startling and perhaps disappointing fact: The desert being traversed by the program has been explored heavily by an expert. One might then decide that the program will not find anything of value missed by the well experienced. For the program to explore potentially far more verdant land can require a radical move, one that might indeed be counterintuitive.

Such a move can feature the avoidance of some type of term, double-negation terms or tautologous terms, for example. The latter concerns conclusions such as those that contain as a subterm $i(t,t)$ for some term t. Term avoidance can often transport the program to a land that a person would find strange or worse. Note that the avoidance of some type of term does not prevent the program from drawing conclusions that would otherwise be drawn; after all, many unrelated paths can lead to the same destination. Indeed, for a selected conclusion free of double negation, one might find a path totally free of double-negation conclusions that culminates in the selected conclusion, and one might also find a path strewn with double negation that culminates in the selected item. Nevertheless, (as noted) such a move has proved most profitable in our research.

An even less intuitive move concerns the use of symmetric difference. For this action, one first finds two proofs of reasonably similar length but that share perhaps half of their deduced steps. A natural position to take asserts that the half-proof in common contains crucial information, steps or lemmas that must be used to reach the desired objective. Contrary to expectation and indeed counterintuitive, the symmetric-difference methodology has the program rely on resonators that correspond to the steps *not* shared by the two proofs, avoids the intersection for guidance. Indeed, although the program might just happen to deduce and use a step in the intersection of the two proofs, no encouragement of this type is given. The use of that methodology can, of course, lead to temporarily finding a rather longer proof, but a proof that can then be used to eventually make a breakthrough. One could go further—which we have not studied—and force the program to avoid shared steps.

We have had success with reliance on a symmetric-difference methodology. The context was Meredith’s single axiom for the <C,I> calculus, two-valued propositional logic in terms of the pair of connectives *C* and *I*, where *C* stands for implication and *I* stands for any tautology in this logic. The target was the Tarski-Bernays axiom system for this area of logic. OTTER discovered a 38-step proof through the use of symmetric difference, and this proof then led to the discovery of a 34-step proof (given shortly). Meredith did not offer any proof that his single axiom implies the Tarski-Bernays system, suggesting that yet another proof was missing. We thus have some evidence that refinement strategies can be of use where no proof was previously offered. Meredith did offer a 38-step proof that begins with his axiom and deduces Lukasiewicz’s single axiom. With our various methodologies, we have found a 31-step refinement (found on the included CD-ROM).

A Proof Based in Part on the Symmetric-Difference Methodology

```
----- Otter 3.0.5b, March 1998 -----
The process was started by wos on soot.mcs.anl.gov, Sun Jun 11 11:10:33 2000
The command was "otter". The process ID is 22747.

-----> EMPTY CLAUSE at   0.36 sec ----> 103 [hyper,2,101,61,98] $ANS(TB_ALL).

Length of proof is 34. Level of proof is 25.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] -P(i(i(a,b),i(i(b,c),i(a,c)))) | -P(i(a,i(b,a))) | -P(i(i(i(a,b),a),a)) | $ANS(TB_ALL).
```

3 [] P(i(i(i(x,y),i(z,i(I,u))),i(i(u,x),i(z,i(v,x))))).
48 [hyper,1,3,3] P(i(i(i(x,y),i(y,z)),i(i(u,y),i(v,i(y,z))))).
49 [hyper,1,3,48] P(i(i(i(x,y),i(z,x)),i(i(u,x),i(v,i(z,x))))).
50 [hyper,1,48,3] P(i(i(x,i(y,i(I,y))),i(z,i(i(y,i(I,y)),i(y,i(u,i(I,y))))))).
51 [hyper,1,49,50] P(i(i(x,i(i(y,i(I,y)),i(y,i(z,i(I,y))))),i(u,i(v,i(i(y,i(I,y)),i(y,i(z,i(I,y)))))))).
52 [hyper,1,3,51] P(i(i(i(i(x,i(I,x)),i(x,i(y,i(I,x)))),z),i(u,i(v,z)))).
54 [hyper,1,48,52] P(i(i(x,y),i(z,i(y,i(u,y))))).
56 [hyper,1,49,54] P(i(i(x,i(y,i(z,y))),i(u,i(v,i(y,i(z,y)))))).
58 [hyper,1,56,52] P(i(x,i(y,i(z,i(u,z))))).
59 [hyper,1,58,58] P(i(x,i(y,i(z,y)))).
61 [hyper,1,59,59] P(i(x,i(y,x))).
63 [hyper,1,3,59] P(i(i(x,y),i(x,i(z,y)))).
65 [hyper,1,63,61] P(i(x,i(y,i(z,x)))).
66 [hyper,1,63,65] P(i(x,i(y,i(z,i(u,x))))).
69 [hyper,1,3,66] P(i(i(i(x,i(y,z)),y),i(u,i(v,y)))).
70 [hyper,1,3,69] P(i(i(x,i(y,i(x,z))),i(u,i(v,i(y,i(x,z)))))).
73 [hyper,1,70,63] P(i(x,i(y,i(z,i(i(z,u),u))))).
74 [hyper,1,73,73] P(i(x,i(y,i(i(y,z),z)))).
75 [hyper,1,74,74] P(i(x,i(i(x,y),y))).
76 [hyper,1,63,75] P(i(x,i(y,i(i(x,z),z)))).
77 [hyper,1,3,75] P(i(i(x,y),i(i(i(y,z),i(I,x)),i(u,y)))).
81 [hyper,1,61,76] P(i(x,i(y,i(z,i(i(y,u),u))))).
83 [hyper,1,3,77] P(i(i(x,y),i(i(i(x,z),i(I,y)),i(u,y)))).
85 [hyper,1,3,81] P(i(i(i(i(x,y),y),z),i(x,i(u,z)))).
86 [hyper,1,70,83] P(i(x,i(y,i(i(i(z,u),i(I,v)),i(i(z,v),v))))).
88 [hyper,1,85,63] P(i(x,i(y,i(i(x,z),i(u,z))))).
89 [hyper,1,86,86] P(i(x,i(i(i(y,z),i(I,u)),i(i(y,u),u)))).
90 [hyper,1,63,88] P(i(x,i(y,i(z,i(i(x,u),i(v,u)))))).
92 [hyper,1,89,89] P(i(i(i(x,y),i(I,z)),i(i(x,z),z))).
94 [hyper,1,92,76] P(i(i(x,i(i(i(x,y),z),z)),i(i(i(x,y),z),z))).
96 [hyper,1,94,90] P(i(i(i(x,y),i(z,i(i(x,u),i(v,u)))),i(z,i(i(x,u),i(v,u))))).
97 [hyper,1,94,88] P(i(i(i(x,y),i(i(x,z),i(u,z))),i(i(x,z),i(u,z)))).
98 [hyper,1,94,61] P(i(i(i(x,y),x),x)).
100 [hyper,1,3,97] P(i(i(x,i(y,z)),i(i(y,x),i(u,i(y,z))))).
101 [hyper,1,96,100] P(i(i(x,y),i(i(y,z),i(x,z)))).

One can summarize with the following aphorism what is occasionally required.

Aphorism: To mine the treasure, the program may be forced to go where no person nor program has gone before.

In addition to the refinement concerning proof shortening, and in the context of his newly discovered twenty-fourth problem, Hilbert might also have been interested in the following refinements. The first, focusing on lemma avoidance (the dispensing with the use of a thought-to-be-indispensable lemma), can easily be addressed with OTTER. One can include appropriate demodulators that block the use of an unwanted formula or equation as well as all instances of such, or one can rely on weighting. If the latter, one includes a resonator with an assigned value strictly greater than the value assigned to max_weight. With this choice, in addition to blocking the use of an unwanted lemma, one also blocks the use of conclusions that are similar in form to the unwanted element (where all variables are treated as indistinguishable). In other words, an entire equivalence class is prevented from participating in the sought-after proof. Of course, if one wishes to admit elements of the equivalence class other than the singled out element, the approach based on demodulation is the choice. One of these mechanisms might come in handy if an expert decides the program is wasting its time pursuing judged-to-be-unprofitable lines of inquiry.

Also of interest, one can attempt to refine a proof by limiting (with the use of an appropriate assignment to max_weight) the complexity of deduced steps, in other words, use the *expression* *complexity strategy*. Instead or in addition, one can use the *variable richness strategy* by assigning an appropriate value for max_distinct_vars to attempt to refine a proof in the context of the maximum number of distinct variables permitted in a deduced step. (We did just that, discovering a five-variable proof for Meredith's single axiom for two-valued sentential calculus, a proof of length 69, given on the CD-ROM.) And, of course, one can make a study whose goal is to produce a proof refinement in many respects by combining various methodologies. Sometimes with a single methodology and sometimes with a combination, we have obtained a single refinement or simultaneous refinements.

We have discussed some of our refinements with Rudiger Thiele, the discoverer of Hilbert's twenty-fourth problem. In his view, they do indeed address various incarnations of that problem and in diverse contexts. We find this situation especially piquant, even amusing, for these studies (as noted) commenced more than eight years ago, a time at which no knowledge concerning Hilbert's twenty-fourth problem was extant. Many, many additional examples of progress with the Hilbert problem can be found on the included CD-ROM.

Dialogue 3

This chapter, as its two predecessors did, produced a fervent dialogue among our six protagonists: Ardent Proponent (AP), Strongly Opposed (SO), Curious Interrogator (CI), Interested Mathematician (IM), Practicing Logician (PL), and Reasonable Skeptic (RS).

RS: We meet again, as scheduled, to discuss aspects of this third chapter. To start things off, I must say somewhat begrudgingly that my skepticism is beginning to wane. The authors of this book present a rather persuasive case, especially in view of their success with shortening proofs offered by experts who are themselves known for their achievements in this area.

IM: Perhaps you are also influenced by the fact that Hilbert was sufficiently interested in proof simplification that he almost offered the problem (as his twenty-fourth) in his 1900 Paris lecture. Problems known as Hilbert's have justifiably received much attention and study throughout the twentieth century.

RS: I don't doubt that is a factor. How amusing that the reported studies of proof refinement commenced more than eight years ago and just happen to be so relevant to Hilbert's twenty-fourth!

PL: Probably because of my concern for aspects of logic, even more than the pertinence to Hilbert's work, I find satisfaction in their successes with the two Meredith theorems. The differences between their 38-step proof and Meredith's 41-step proof for his single axiom are particularly interesting. For example, most unexpected is the proof of the first, and in some sense most powerful, of the three Lukasiewicz axioms before proving the second. I also chuckled at the term ''cramming'' in their treatment of abridging the Meredith-and-Prior abridgment. Rather astounding indeed that OTTER could cram proofs of the first and second Tarski-Bernays axioms into a 32-step proof, thirty steps of which were already used for proving the third.

SO: The comments by the authors concerning the impracticality of a breadth-first search make me suspicious. After all, they used it briefly in their cramming methodology. Just have the program try for a short proof by traversing the levels and take the first one that is completed.

AP: Here is a semantically uninteresting example (consisting of eight clauses) showing why the first proof may not be the shortest.

```
-P.
P | Q.
P | R.
-Q | S.
-R | -S.
P | T.
-T | U.
-U | P.
```

If one focuses on just the first five clauses, one can find a 4-step level-2 proof, completing with the deduction of *S* and *-S*. If one focuses instead on the sixth through the eighth, one can complete a 3-

step level-3 proof, completing with the deduction of *P* which contradicts the first clause. Regarding impracticality, one can imagine elaborating on the given example in a manner that produces a proof requiring two clauses at level 1, four at level 2, and the like. If the shortest proof has level 24—as is the case for one the abridgments cited for the Meredith single axiom— the program could easily drown before completing the desired proof.

CI: Actually, although the specific results should be the items of concern, I prefer the methodologies. I can't help wondering how and when they will appear in later chapters and, more exciting in the long run, whether chip designers and those interested in program synthesis will eventually put the methodologies to use. Their research sure does rely on a wide variety of OTTER's arsenal of weapons.

AP: I am certain that you especially, IM, find interesting the use of resonators to enable the program to rely on successes in one context to increase its chance of succeeding in a related context. In particular, they discovered a shorter six-variable proof of the sufficiency of the Meredith single axiom and also a shorter double-negation-free proof in that context. Then they used their new proofs to direct the program's reasoning, eventually discovering a 38-step abridgment of Meredith's 41-step proof. Are any of you five familiar with a fine alternative to the resonance strategy for directing the program's attack, namely, Veroff's hints strategy?

CI: I know that the authors of this book use it profitably, but I don't really know how it differs.

AP: Well, in contrast to a resonator that treats all variables as indistinguishable and, therefore, focuses on equivalence classes of formulas or equations, a hint treats the variables precisely as written and focuses on items that are identical to the hint, subsume the hint, or are subsumed by the hint, depending on the included options. In general, the hints strategy works rather faster than does the resonance strategy. On the other hand, when seeking a shorter proof with an assignment of a small max_weight, use of the hints strategy is more likely to return (proofcheck) the proof in hand when compared with the resonance strategy. They each serve the researcher well, and the choice between them is best based on the objective; they also work well together.

SO: Hilbert, Meredith, Prior, and the rest notwithstanding, if shorter proofs are so preferred, why is the task left to those other than the originator of a result?

RS: Cynical though it may sound—and perhaps putting it the way you would, SO—life is simply too short to stop and clean up one's new proof. Besides, one does not notice the length of a proof of the key theorem that relies on a sequence of proved lemmas (that precede the main theorem) or, worse, that relies on lemmas and theorems proved in other publications. Ordinarily, one would not stop to count all of the steps used explicitly and implicitly. And, as an aside, who can afford to start from scratch the way OTTER does, ignoring the vast amount of research already completed by others? Obviously—and somewhat upsetting—the program has proved quite effective even though it always begins its attack with little or no knowledge, other than the influence provided by included resonators or included hints.

SO: Again we return to an absurdity, the lack of use of a large database of accrued results. Somebody ought to address this aspect, if it can be effectively addressed, which I sincerely doubt. Then there is the unrelated agony of plowing through a gigantic output file to see if any of the desired conditions (such as a member of a target axiom system) have been proved; of course, I am talking about the case in which the entire system has not yet been deduced.

AP: As for your last criticism, you lose this round SO; OTTER wins another. Specifically, the passive list can be put to good use in locating partial successes, whether the objective has been reached or not. For example, if the goal is to deduce the Tarski-Bernays system, one places its negation in list(usable). In other words, if the goal is viewed as seeking a proof of *P* **and** *Q* **and** *r*, one places in list(usable) the clause -P | -Q | -R. To monitor progress and capture partial successes, one places in list(passive) each of -P, -Q, and -R. If and when, say, *P* is proved, OTTER sends a message to your terminal that a proof has been discovered. If, as was the case with the attempt by the authors to abridge the Meredith-and-Prior abridgment, a potentially interesting proof of one of three axioms is found, that proof is automatically offered to the researcher.

CI: This monitoring use of the passive list can be put to less obvious uses, I would guess.

AP: Indeed so. One can place the negation (denial) of various lemmas expressible as unit clauses in list(passive) with the intention of checking the program's emphasis on the appropriate area of conclusion deduction. Such lemmas may or may not be relevant to the theorem under attack. They serve to indicate that good things are happening. For a not-so-obvious example, one might have in hand a proof in which double-negation terms are present or in which some supposedly indispensable lemma is present. The goal might be to find a double-negation-free proof of the theorem of concern or to find a proof in which the unwanted lemma is absent. For monitoring progress, one might place in the passive list the negations of the steps of the proof in hand and judge progress or even the nearness to reaching the goal by noting which and how many of the steps get proved.

SO: What about my comment about the lack of use of the results of years and years of study by the fine minds?

IM: I'd like to try this one, AP. Fine minds are indeed adept at selecting from a huge set of lemmas and theorems presented in books, papers, and lectures just those that are likely to lead to the discovery of a sought-after proof or the answer to a posed question. I suspect, at least for now, the means for a program to make such wise choices from the corresponding database are unknown. The situation may be reminiscent of the problem faced by a person versus that faced by a reasoning program when, say, confronted with proving commutativity for groups in which the square of every element x is the identity e. A person might quickly start with $(yz)(yz) = e$ by instantiating x to yz; a program such as OTTER ordinarily would not draw this conclusion, and, if it did, the conclusion would be immediately discarded because of being subsumed by $xx = e$. OTTER does not offer instantiation because the means for effective choices among the myriad of instances of an equation or a formula is lacking. How did I do, AP?

AP: You may become a user of OTTER if you are not careful. You get an A. Indeed, how a person makes good selections from a vast pool of knowledge is unclear, and currently nobody has suggested a powerful means for the program to wisely choose from a huge database of lemmas and theorems. Just a thought: Perhaps included resonators in the input could be used to select matching formulas or equations from such a database.

CI: The subjects of huge arrays of already-proved theorems and lemmas and the myriad of instances suggests to me, for some odd reason, the question of axiom dependence. Is it wise to supply an automated reasoning program with an independent set of axioms when embarking on a study, or are dependencies of use?

IM: When one is introduced, for example, to group theory, one is generally presented with a set of axioms in which dependency is present. Indeed, one is told that a two-sided identity exists, although the axiom asserting the existence of a left identity is sufficient. In other words, the addition of an axiom asserting the existence of a right identity introduces a dependency.

PL: Logicians are often concerned with axiom dependence, enjoying a system in which no dependency is present. What a program prefers, however, I don't know.

AP: Although far from a uniform practice, many users of a reasoning program claim that effectiveness is increased when the type of dependency cited for group theory is included. The naturalness of asserting a two-sided identity hints at its value in the context of automation. As for areas of logic as opposed to mathematics, I have no data and no opinion regarding this aspect of so-called redundancy in the context of automated reasoning.

RS: Since AP is not too subtly looking for converts, seeking to add to the users of OTTER, let's return to the feature in this chapter, proof refinement. Don't fret, SO; I'm not going to become ZC (Zealous Convert). I've simply been reviewing the possible value of refined proofs, whether the refinement is of length, complexity, term avoidance, lemma avoidance, or whatever. Ceteris paribus, refined proofs offer quite a lot. I can see pedagogical advantages, the advantage that a refinement might more easily have practical application, or might lead to a breakthrough of some type (such as settling a conjecture or answering an open question or prompting a new discovery). Proof refinements have aesthetic appeal, offer elegance, and—especially for the lover of Mozart—possess beauty. And then there is Hilbert's twenty-fourth, as we have discussed.

PL: Whether refined or not, the explicit listing of parents and inference rule does add to the appeal of an OTTER proof, especially for various logicians.

IM: And those properties bring to mind a remark Paul Halmos once made to a graduate student who was looking for the ''rules that must be satisfied'' for something to be a proof. Halmos replied that, when a proof is sent to three referees, one might say the proof is too detailed, one might offer that it is incomplete and requires fuller explanation, and one might find it acceptable as is—too much, not enough, perfect. OTTER's proofs are presented in a canonical form, axioms and clauses needed for inference rules such as condensed detachment, followed by the deduced steps, with no step omitted.

AP: None omitted if you allow for a list of demodulators (rewrite rules) for simplification and canonicalization.

CI: One can easily try crazy ideas with such a program; it's cheap, just a small amount of effort to present the problem. The worst that can occur is that you learn nothing. Distantly related and as evidence of the power of the cited methodologies, I have heard from a most reliable source that at least one of the authors is frequently successful in obtaining a refinement *without* ever reading the clauses contained in an input file received by e-mail. That suggests that the methodologies are independent of the area of logic or mathematics under investigation.

PL: The term ''refinement'' is in some cases odd because the refinement can be a proof sharply different from that being refined, witness the Meredith single axiom study. Incidentally, Ulrich has offered two other interesting measures of refinement, the first of which was not addressed in this chapter, and the second only briefly in the context of the success with finding a 38-step refinement of Meredith's 41-step proof. He studied (with a program he wrote) axiom systems in terms of minimizing the distance in terms of applications of condensed detachment from the axiom set to significant theorems, which might be quite germane to the aspect of proof refinement concerned with proof length. He also discussed at a workshop the measure of refinement that reflects the total number of symbols in a proof, namely, size.

AP: I cannot think of a way to have OTTER easily provide assistance in either of Ulrich's measures.

SO: By the way, I bet nobody has thought about the obstacle encountered when subsumption is used. In particular, if one wishes a deduction of a specific formula or equation, I bet that OTTER (in the presence of subsumption) can fail, instead deducing a more general result. How annoying!

PL: That remark reminds me of the interesting property of D-completeness possessed by two-valued sentential calculus. Specifically, if **A** is an axiom system for this area of logic and T a tautology (theorem) in the logic, then there always exists a condensed-detachment proof of T from **A**. Now if R is strictly more general than T, and if OTTER deduces R before it deduces T, subsumption will prevent the retention of T, which will make it difficult for the researcher to get a condensed-detachment proof of T.

AP: A trick that sometimes helps is that of demodulating T to some constant c not in the remainder of the input and placing $-c$ in the passive list. Since OTTER applies demodulation before it applies subsumption, sometimes the needed proof is (in effect) obtained; instead, if other occurrences of subsumption of needed intermediate conclusions are encountered, the sought-after proof may not be found. Of course, if such a proof is completed, by hand one is forced to so-to-speak undemodulate the constant to the desired formula T. This pyrrhic loss seems inconsequential, when all works.

IM: The material of this chapter, all in all, seems to lay the groundwork for studying what might be termed a lattice of proofs, or different lattices of proofs, each based on the type of refinement under consideration. That topic should be sufficiently heavy to cause the six of us to adjourn, meeting again, after reading the next chapter, for a fourth dialogue.