bck.bci
created 06-03-2006
revised last 06-11-2009

The Joy of Solving a Decades-Old Mystery: Success with the BCK and BCI Logics*

Larry Wos

Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439
wos@mcs.anl.gov

1. Challenges, Goals, Successes

The winds of research are many. From whence they come we sometimes do not know. But, sometimes, we do know the source of the refreshing breeze or delightful blast. The seeds such winds bring can be few, or they can be many. Share with me, via this notebook, an unexpected wind and the seeds it brought forth.

The title of this essay certainly expresses what I experienced when attempting to solve a decades-old mystery. Whoever reads the following (which can be viewed as an unpolished essay or, perhaps more accurately, as a researcher's notebook) will learn of a journey in search of a type of treasure that in turn solved a long-standing mystery. The journey was made with William (Bill) McCune's automated reasoning program OTTER as companion. That program typically is given a set of hypotheses, *axioms*, and asked to complete an assignment. The assignment usually concerns finding a proof. OTTER tirelessly and flawlessly draws one conclusion after another, flawlessly in the sense that each conclusion follows inevitably from the starting set of axioms. Especially if the problem under study is deep, much strategy is needed to enable a reasoning program to succeed in solving the problem, strategy to restrict its reasoning, and strategy to direct it. Sometimes, as data included here will suggest, before the assignment has been completed, ten million conclusions are drawn of which more than one million are retained; sometimes a few days of real time is required.

As you read of this journey, you will meet a few of my colleagues and learn of the nature of the treasure being sought. You will also be given information that might, just might, cause you to consider the intriguing and most challenging problem of automating much of the approach that is offered here. My colleague Ross Overbeek suggests that this activity merits serious thought.

This essay (or notebook) focuses on a mystery implied by a success of someone whom I consider to be one of the great logicians of all time, namely, C. A. Meredith. That researcher was known, among other things, for finding elegant proofs of interesting theorems, proofs that might be most difficult to improve upon. The most obvious improvement that can be made concerns the length of a proof, the number of deduced steps in the proof (not counting the hypotheses or axioms). Hence the implied mystery: Does there exist a shorter proof, one with fewer steps, than that given by Meredith for the area of logic under study, in the case featured here, a logic called *BCK*? What you will be offered here is a possible path to solving the mystery, namely, an approach for finding shorter proofs—and more.

However—if you have little or no interest in proof refinement, preferring instead to prove new theorems—as supported by evidence found in this essay, the approach also can be effective in finding a so-

This work was supported in part by the U.S. Department of Energy, under Contract DE-AC02-06CH11357.

called first proof. (You need not know much about logic or, for that matter, be a big fan of logic to enjoy what is to come; indeed, what you are about to encounter can justly be viewed as puzzle solving. Further, you are free to take any or all of what is presented here and run with it; in fact, I would be most pleased if you became at least somewhat interested in the automation of reasoning.)

Quite likely of concern, how might one go about finding that possibly elusive proof? More precisely, from the viewpoint of the automation of logical reasoning, how might one assist a reasoning program so effectively that success results? Of course, as you may have surmised, the assistance that can be provided takes the form of strategy of various types. For a small taste of such, you can suggest to OTTER that focusing on one class of formulas or equations over some other class is far more likely to yield useful information. But, before showing how the mystery was unraveled—had it not been unraveled, most likely this essay would not be being written—some background is in order.

Consistent with my preference for having access to most or all that is needed to understand the details that are given, I include a number of items, some or many of which may be familiar to you already. As in all games, as well as implied in puzzles, a rule was present, a rule for drawing conclusions. The rule used in this study, which was invented by Meredith in the early 1950s and which is now used in most modern computer-assisted studies of sentential logics, is called *condensed detachment*, and is represented by the following, where "-" denotes logical **not** and " | " denotes logical **or**.

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

You can think of "P" as meaning "is provable" and *i* as denoting implies, and you can view both "x" and "y" as variables that run over possible formulas. As for the axioms or hypotheses, the *BCK* logic can be captured (axiomatized) with the following three clauses, where the function *i* denotes logical implication.

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

They are, respectively, *B*, *C*, and *K*.

Meredith found a single, 17-symbol formula, the following, that provides a complete axiomatization for the *BCK* logic.

P(i(i(i(x,y),z),i(i(u,i(z,v)),i(y,i(u,v))))). % Meredith's known single axiom for BCK

Later, (my colleague) D. (Ted) Ulrich found seven more of this type. All eight of these single axioms for *BCK* are given on Ulrich's own web site at http://web.ics.purdue.edu/~dulrich/BCK-page.htm. In other words, from the given Meredith formula, you can deduce what you need to study the logic in focus, and, in particular, you can deduce, using condensed detachment, the three formulas known, respectively, as *B*, *C*, and *k*. In addition, the Meredith formula must, of course, be a theorem of the *BCK* logic, deducible from the three axioms, *B*, *C* and *K*.

Fortunately, from the viewpoint of this essay, Meredith provided the target, the proof at which to aim or, more precisely, the proof to be refined. Specifically, he presented a proof that derives, from his single axiom, each of *B*, *C*, and *K*, a proof consisting of thirty applications of condensed detachment. (I note that, throughout this essay, I have not addressed the aspect that shows that candidate single axioms are themselves theorems of the logic under study; they must be to be admissible as possible axioms.) For the historian, as my colleague Ulrich informs me, Meredith's proof is of length 29, omitting the step that derives *B*; indeed, Meredith is content to have his proof complete with the closely related formula *B*', the following.

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

With the foregoing, the game is on: The target is known, a 30-step proof; the rule(s) of the game are given, specifically, condensed detachment is to be used; and the challenge is issued. You are asked—I asked myself—to find a proof of length strictly less than thirty that derives each of *B*, *C*, and *K*, starting with the Meredith axiom. I had the advantage of knowing that a derivation of *B* will require more of a person, or of a program, than either the derivation of *C* or a derivation of *K*. More important, I had in hand Meredith's proof to (so-to-speak) shorten.

I now offer two closely related proofs. The first, supplied by Ulrich, is Meredith's, essentially, without derivation information that gives the precise history of each deduction; indeed, I asked Ulrich for the formulas alone (without their history), with the intention of using them in the approach to be presented. The second proof is more complete, giving the history of each deduction; it results from my attempt to have OTTER reproduce Meredith's proof, prompted by the suspicion that you might wish to see a fuller proof. The second differs from the first in that it includes a formula, *i*(*x*,*x*), that is not in Meredith's proof. Meredith, instead, includes an instance of that formula, *i*(*i*(*x*,*y*),*i*(*x*,*y*)). (OTTER ordinarily will not retain an instance of a formula or equation already present in that subsumption is typically invoked.)

After reading the two proofs, you might pause to see if you can take either proof and shorten it, *without* reading the approach to be presented in the next section. By doing so, you might gain insights into what the obstacles are to proof refinement in the context of proof length.

Meredith's Proof

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

OTTER's Similar Proof

```
-----> EMPTY CLAUSE at   .7 sec ----> 143 [hyper,2,79,106,140] $ANS(all).

Length of proof is 30.  Level of proof is 22.
```

```
---------------- PROOF ----------------

1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] -P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | -P(i(a1,i(b,a1))) |
   -P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(all).
3 [] P(i(i(i(u,v),w),i(i(x,i(w,y)),i(v,i(x,y))))).
61 [hyper,1,3,3] P(i(i(x,i(i(i(y,i(z,u)),i(v,i(y,u))),w)),i(z,i(x,w)))).
63 [hyper,1,61,3] P(i(x,i(i(i(y,z),u),i(z,i(i(v,i(x,w)),i(v,w)))))).
65 [hyper,1,61,63] P(i(x,i(y,i(i(x,z),i(i(u,i(y,v)),i(u,v)))))).
67 [hyper,1,65,3] P(i(x,i(i(i(i(i(y,z),u),i(i(v,i(u,w)),i(z,i(v,w)))),v6),
  i(i(v7,i(x,v8)),i(v7,v8))))).
69 [hyper,1,61,67] P(i(i(x,i(y,z)),i(u,i(i(v,i(u,w)),i(v,w))))).
71 [hyper,1,69,69] P(i(x,i(i(y,i(x,z)),i(y,z)))).
74 [hyper,1,71,71] P(i(i(x,i(i(y,i(i(z,i(y,u)),i(z,u))),v)),i(x,v))).
77 [hyper,1,74,74] P(i(i(i(x,i(i(y,i(x,z)),i(y,z))),i(i(u,i(i(v,i(u,w)),
  i(v,w))),v6)),v6)).
79 [hyper,1,74,71] P(i(i(x,i(y,z)),i(y,i(x,z)))).
81 [hyper,1,3,77] P(i(i(x,i(y,z)),i(i(i(u,i(i(v,i(u,w)),i(v,w))),y),i(x,z)))).
83 [hyper,1,3,79] P(i(i(x,i(i(y,i(z,u)),v)),i(i(y,u),i(x,v)))).
86 [hyper,1,3,81] P(i(i(x,i(i(i(i(y,i(i(z,i(y,u)),i(z,u))),v),i(w,v6)),v7)),
  i(i(v,v6),i(x,v7)))).
88 [hyper,1,83,71] P(i(i(x,y),i(z,i(x,y)))).
9 [hyper,1,86,71] P(i(i(x,y),i(z,i(i(i(u,i(i(v,i(u,w)),i(v,w))),x),y)))).
95 [hyper,1,3,88] P(i(i(x,i(i(y,i(z,u)),v)),i(u,i(x,v)))).
97 [hyper,1,88,3] P(i(x,i(i(i(y,z),u),i(i(v,i(u,w)),i(z,i(v,w)))))).
99 [hyper,1,74,9] P(i(i(x,y),i(i(i(z,i(i(u,i(z,v)),i(u,v))),x),y))).
11 [hyper,1,95,71] P(i(x,i(y,i(z,x)))).
13 [hyper,1,99,99] P(i(i(i(x,i(i(y,i(x,z)),i(y,z))),i(u,v)),
   i(i(i(w,i(i(v6,i(w,v7)),i(v6,v7))),u),v))).
16 [hyper,1,74,11] P(i(x,i(y,x))).
11 [hyper,1,3,13] P(i(i(x,i(i(i(i(y,i(i(z,i(y,u)),i(z,u))),v),w),v6)),
   i(i(v,w),i(x,v6)))).
114 [hyper,1,74,16] P(i(x,x)).
116 [hyper,1,11,97] P(i(i(x,y),i(z,i(i(u,i(y,v)),i(x,i(u,v)))))).
119 [hyper,1,74,116] P(i(i(x,y),i(i(z,i(y,u)),i(x,i(z,u))))).
122 [hyper,1,79,119] P(i(i(x,i(y,z)),i(i(u,y),i(u,i(x,z))))).
124 [hyper,1,122,114] P(i(i(x,y),i(x,i(i(y,z),z)))).
127 [hyper,1,124,124] P(i(i(x,y),i(i(i(x,i(i(y,z),z)),u),u))).
129 [hyper,1,79,127] P(i(i(i(x,i(i(y,z),z)),u),i(i(x,y),u))).
132 [hyper,1,129,79] P(i(i(x,y),i(i(y,z),i(x,z)))).
14 [hyper,1,79,132] P(i(i(x,y),i(i(z,x),i(z,y)))).
```

2. An Approach For Finding Shorter Proofs

To enable a person to independently verify my assertions and to enable one to deviate easily from the approach to be given here—other approaches of course can produce good results—I provide various input files. I also include in this notebook various proofs, some of which may provide researchers with new challenges in the form of seeking proofs of even shorter length. I note that, when a percent sign occurs on a line, from that point forward to the end of the line is considered by OTTER (the program I used) as comment.

Prior to providing the details of the approach taken in search of a shorter proof, one of length 29 or less (applications of condensed detachment), I first note that I cannot offer an algorithm. Even stronger, from what I know, no effective algorithm exists. Indeed, I select, based on previous experiences and experiments, from a variety of choices regarding various parameters and procedures. Further, I strongly suspect that an algorithm guaranteed to find shorter proofs, if such exist, is impractical at least. However, Overbeek is probably right on target when he suggests that much of what I do can be profitably automated.

Before presenting (in the form of phases) an approach that succeeded in finding a proof shorter than length 30 for the Meredith single axiom for the *BCK* logic, I include a bit of history. That episode, at least for me, captures some of the charm of mathematics and logic, as well as that of automated reasoning. Indeed, from episodes (of the type in focus) sometime spring new ideas, new challenges, and additional reasons for putting on paper what followed them.

My colleague Michael Beeson and I were discussing proof by induction versus proof by methods in first-order predicate calculus. I asserted that a proof of the latter type often taught one things that had not been evident in the proof by induction. He asked for an example, and, as usual, I turned to my colleague Ulrich for a good example. Rather than a detailed account of what occurred, I am content here with highlights.

Ulrich had found seven new single axioms for the *BCK* logic, at first only one of which had been brought to my attention. He sent me an e-mail, upon learning of my quest for an example showing that a first-order proof can teach things that the induction proof does not. By way of the briefest summary—his e-mail was far, far more informative than indicated here—he noted that one of his new single axioms could have been proven, by relying on induction, to be a theorem of the implicational fragment of intuitionistic logic. He next told me that, if one instead sought a proof with OTTER, one would find that but three of the axioms for that area of logic were used; in particular, the fourth, called *W*, was not used. Therefore, as he also noted, the formula in question is a theorem of a weaker logic, the *BCK* logic, introduced by C. A. Meredith. So, in the context of this part of the episode, one might not have realized that the formula was a theorem of the weaker logic and, as it turned out, an axiom for that weaker logic.

His e-mail was so instructive and so inspiring that I became interested in the *BCK* logic and, specifically, in the Meredith single axiom and the Meredith 30-step proof for that axiom. (I note that Ulrich's e-mails set a standard of clarity and stimulation that may be impossible to duplicate.) You thus have the beginnings of a discussion with others of the merits and properties of induction proofs versus OTTER-like proofs. You also see how and why I have put to paper the following approach (in phases), as well as related commentary.

Phase 1

The first step in my assault on the Meredith proof was to have OTTER seek proofs of each of *B*, *C*, and *K* with the goal of seeing how hard each is to prove for that invaluable assistant. I submitted the following input file to OTTER; I follow it with some commentary and some explanation of why the input file is structured the way it is, including justifications.

Input File 1, for Meredith's Single Axiom for BCK

```
set(hyper_res).
assign(max_weight,48).
% assign(change_limit_after,800).
% assign(new_max_weight,22).
assign(max_proofs,-1).
clear(print_kept).
% set(process_input).
% set(ancestor_subsume).
set(back_sub).
clear(print_back_sub).
clear(print_kept).
clear(print_new_demod).
clear(print_back_demod).
clear(print_back_sub).
assign(max_distinct_vars,12).
assign(pick_given_ratio,1).
assign(max_mem,7500).
assign(report,54).
set(order_history).
set(input_sos_first).
%  set(sos_queue).
%  assign(bsub_hint_wt,1).
%  set(keep_hint_subsumers).
%  assign(heat,0).

weight_list(pick_and_purge).
%  Following 31 include Meredith's single for BCK, and his 30-step proof.
weight(P(i(i(i(u,v),w),i(i(x,i(w,y)),i(v,i(x,y))))),2).
%  Meredith's single axiom for BCK
weight(P(i(i(u,i(i(i(v,i(w,x)),i(y,i(v,x))),z)),i(w,i(u,z)))),2).
weight(P(i(u,i(i(i(v,w),x),i(w,i(i(y,i(u,z)),i(y,z)))))),2).
weight(P(i(u,i(v,i(i(u,w),i(i(x,i(v,y)),i(x,y)))))),2).
weight(P(i(u,i(i(i(i(i(v,w),x),i(i(y,i(x,z)),i(w,i(y,z)))),v6),i(i(v7,i(u,v8)),
  i(v7,v8))))),2).
weight(P(i(i(u,i(v,w)),i(x,i(i(y,i(x,z)),i(y,z))))),2).
weight(P(i(u,i(i(v,i(u,w)),i(v,w)))),2).
weight(P(i(i(u,i(i(v,i(i(w,i(v,x)),i(w,x))),y)),i(u,y))),2).
weight(P(i(i(u,i(v,w)),i(v,i(u,w)))),2).  %  C
weight(P(i(i(i(u,i(i(v,i(u,w)),i(v,w))),i(i(x,i(i(y,i(x,z)),i(y,z))),v6)),v6)),2).
weight(P(i(i(u,i(v,w)),i(i(i(x,i(i(y,i(x,z)),i(y,z))),v),i(u,w)))),2).
weight(P(i(i(u,i(i(i(i(v,i(i(w,i(v,x)),i(w,x))),y),i(z,v6)),v7)),i(i(y,v6),
  i(u,v7)))),2).
weight(P(i(i(u,v),i(w,i(i(i(x,i(i(y,i(x,z)),i(y,z))),u),v)))),2).
weight(P(i(i(u,v),i(i(i(w,i(i(x,i(w,y)),i(x,y))),u),v))),2).
weight(P(i(i(i(u,i(i(v,i(u,w)),i(v,w))),i(x,y)),i(i(i(z,i(i(v6,i(z,v7)),
  i(v6,v7))),x),y))),2).
weight(P(i(i(u,i(i(i(i(v,i(i(w,i(v,x)),i(w,x))),y),z),v6)),i(i(y,z),i(u,v6)))),2).
weight(P(i(i(u,i(i(v,i(w,x)),y)),i(i(v,x),i(u,y)))),2).
weight(P(i(i(u,v),i(w,i(u,v)))),2).
weight(P(i(i(u,i(i(v,i(w,x)),y)),i(x,i(u,y)))),2).
weight(P(i(u,i(v,i(w,u)))),2).  %  K
weight(P(i(u,i(v,u))),2). % K
weight(P(i(u,i(i(i(v,w),x),i(i(y,i(x,z)),i(w,i(y,z)))))),2).
weight(P(i(i(u,v),i(w,i(i(x,i(v,y)),i(u,i(x,y)))))),2).
weight(P(i(i(u,v),i(i(w,i(v,x)),i(u,i(w,x))))),2).
weight(P(i(i(u,i(v,w)),i(i(x,v),i(x,i(u,w))))),2).
weight(P(i(i(u,v),i(u,v))),2).
weight(P(i(i(u,v),i(u,i(i(v,w),w)))),2).
weight(P(i(i(u,v),i(i(i(u,i(i(v,w),w)),x),x))),2).
weight(P(i(i(i(u,i(i(v,w),w)),x),i(i(u,v),x))),2).
weight(P(i(i(u,v),i(i(v,w),i(u,w)))),2).  %  B'
weight(P(i(i(u,v),i(i(w,u),i(w,v)))),2).  %  B
end_of_list.

list(usable).
-P(i(x,y)) | -P(x) | P(y).
```

```
end_of_list.

list(sos).
P(i(i(i(u,v),w),i(i(x,i(w,y)),i(v,i(x,y))))).  % Meredith's single axiom for BCK
end_of_list.

list(passive).
% Following 30, if unnegated, prove B,C,K from Meredith single.
-P(i(i(a1,i(i(i(b,i(a2,a3)),i(a4,i(b,a3))),a5)),i(a2,i(a1,a5)))) | $ANS(inter1).
-P(i(a1,i(i(i(b,a2),a3),i(a2,i(i(a4,i(a1,a5)),i(a4,a5)))))) | $ANS(inter1).
-P(i(a1,i(b,i(i(a1,a2),i(i(a3,i(b,a4)),i(a3,a4)))))) | $ANS(inter1).
-P(i(a1,i(i(i(i(i(b,a2),a3),i(i(a4,i(a3,a5)),i(a2,i(a4,a5)))),b6),
  i(i(b7,i(a1,b8)),i(b7,b8))))) | $ANS(inter1).
-P(i(i(a1,i(b,a2)),i(a3,i(i(a4,i(a3,a5)),i(a4,a5))))) | $ANS(inter1).
-P(i(a1,i(i(b,i(a1,a2)),i(b,a2)))) | $ANS(inter1).
-P(i(i(a1,i(i(b,i(i(a2,i(b,a3)),i(a2,a3))),a4)),i(a1,a4))) | $ANS(inter1).
-P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | $ANS(C).  % C
-P(i(i(i(a1,i(i(b,i(a1,a2)),i(b,a2))),i(i(a3,i(i(a4,i(a3,a5)),i(a4,a5))),b6)),
  b6)) | $ANS(inter1).
-P(i(i(a1,i(b,a2)),i(i(i(a3,i(i(a4,i(a3,a5)),i(a4,a5))),b),i(a1,a2)))) | $ANS(inter1).
-P(i(i(a1,i(i(i(i(b,i(i(a2,i(b,a3)),i(a2,a3))),a4),i(a5,b6)),b7)),i(i(a4,b6),
  i(a1,b7)))) | $ANS(inter1).
-P(i(i(a1,b),i(a2,i(i(i(a3,i(i(a4,i(a3,a5)),i(a4,a5))),a1),b)))) | $ANS(inter1).
-P(i(i(a1,b),i(i(i(a2,i(i(a3,i(a2,a4)),i(a3,a4))),a1),b))) | $ANS(inter1).
-P(i(i(i(a1,i(i(b,i(a1,a2)),i(b,a2))),i(a3,a4)),i(i(i(a5,i(i(b6,i(a5,b7)),
  i(b6,b7))),a3),a4))) | $ANS(inter1).
-P(i(i(a1,i(i(i(i(b,i(i(a2,i(b,a3)),i(a2,a3))),a4),a5),b6)),i(i(a4,a5),
  i(a1,b6)))) | $ANS(inter1).
-P(i(i(a1,i(i(b,i(a2,a3)),a4)),i(i(b,a3),i(a1,a4)))) | $ANS(inter1).
-P(i(i(a1,b),i(a2,i(a1,b)))) | $ANS(inter1).
-P(i(i(a1,i(i(b,i(a2,a3)),a4)),i(a3,i(a1,a4)))) | $ANS(inter1).
-P(i(a1,i(b,i(a2,a1)))) | $ANS(inter1).
-P(i(a1,i(b,a1))) | $ANS(K).
-P(i(a1,i(i(i(b,a2),a3),i(i(a4,i(a3,a5)),i(a2,i(a4,a5)))))) | $ANS(inter1).
-P(i(i(a1,b),i(a2,i(i(a3,i(b,a4)),i(a1,i(a3,a4)))))) | $ANS(inter1).
-P(i(i(a1,b),i(i(a2,i(b,a3)),i(a1,i(a2,a3))))) | $ANS(inter1).
-P(i(i(a1,i(b,a2)),i(i(a3,b),i(a3,i(a1,a2))))) | $ANS(inter1).
-P(i(i(a1,b),i(a1,b))) | $ANS(inter1).
-P(i(i(a1,b),i(a1,i(i(b,a2),a2)))) | $ANS(inter1).
-P(i(i(a1,b),i(i(i(a1,i(i(b,a2),a2)),a3),a3))) | $ANS(inter1).
-P(i(i(a1,b),i(i(b,a2),i(a1,a2)))) | $ANS(BP).  % B'
-P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(B).  % B
end_of_list.
```

Perhaps the first aspect to discuss concerns the lists of *clauses*; clauses are used to represent information and, in some cases, to assist in representing inference rules. In list(usable) is the clause (already seen in Section 1) to be used with the inference rule *hyperresolution* to capture condensed detachment. In list(sos), you will find but one clause, that for Meredith's single axiom, already seen in Section 1. In list(passive), you will find the negations, or denials, of various formulas, including that for each of *B*, *C*, and *K*. Items in list(passive) are used mainly to signal assignment completion, when two clauses are found that each consists of a single literal and that conflict, unit conflict. Its items also play a role in subsumption. Clauses in list(sos) begin driving the program's reasoning. The initial clauses in list(usable) are used to complete an application of an inference rule. Also used to complete an inference-rule application are clauses moved to list(usable) that have already been used to initiate such an application.

You can see that I need to include in the input the clause corresponding to condensed detachment, the clause (formula) to be studied and to derive other formulas, and the negations of each of *B*, *C*, and *K* (which are the goals). As for the remaining clauses in list(passive), they are included to monitor progress. As each is derived, or a generalization of such, OTTER signals a proof has been found. The more of the so-called intermediate steps that are proved, the closer to assignment completion.

OTTER quickly found proofs of each of the three targets: that for *C* of length 7, that for *K* of length 12, and that for *B* of length 29. But, you ask, what of the other list, weight_list(pick_and_purge)? Why was it included?

The elements of the pick_and_purge list are used to both restrict and direct OTTER's reasoning. The number that occurs in a line, the weight template, is treated as a measure of the significance of the formula or equation residing there. The smaller the value, the greater the significance. Any deduced item that is similar to the formula or equation residing there is automatically assigned the same value. The word "similar" refers to the fact that all variables in a weight template are considered to be indistinguishable, just treated as signifying a variable. As for restricting the reasoning, if the value occurring in a line is strictly greater than the value assigned to max_weight, any deduced item that is similar to the formula or equation residing there is automatically discarded. In the context of directing the reasoning, a retained item that matches a formula or equation within a weight template, where the corresponding assigned value is small (and not exceeding max_weight), will be given preference for initiating inference-rule application.

The inclusion of the thirty-one weight templates, each assigned a small value, directs the program's reasoning by having it prefer deduced formulas that are similar to any of the thirty-one steps of Meredith's proof, including his formula.

As for the *set* commands, the key command is set(order_history), a command that has the program place the numbers giving the history of a deduction in the order nucleus (for condensed detachment), major premiss, minor premiss. (For correctness, but perhaps for stubbornness, I prefer the given spelling of premiss; see Church on logic, who states that this spelling is the only acceptable.) If you know which formula plays the role of major and which of minor premiss, that information can sometimes be put to good use. Next in order among the *set* commands is that instructing OTTER to use the inference rule hyperresolution, a rule that deduces a conclusion only if it is free of a **not** sign.

Of the *assign* parameters, that for max_weight has the program discard any deduced item whose weight, or complexity, strictly exceeds 48. The command focusing on vars has the program discard any deduced item that relies on strictly more than twelve distinct variables. The assignment of the value −1 to max_proofs permits, even encourages, the program to find one proof after another, with no limit, except, of course, that placed on the program by time or memory. Finally, the pick_given_ratio parameter's value *n*, an integer, has the program choose to drive its reasoning *n* items based on complexity, 1 based on first-come first-serve, *n*, 1, and so on.

Phase 1 succeeded; indeed, the goal was reached, that of seeing how easy it was for OTTER to prove each of the three formulas in focus. As noted, each was quickly proved.

Phase 2

In the second phase of the approach under discussion, just two changes were made. The first change was that of including the following clause.

```
-P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | -P(i(a1,i(b,a1))) |
 -P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(all).  % BCK
```

this type of clause is precisely the type Dana Scott was after more than a decade ago. In particular, he wished to have, instead of three separate proofs (as in the case in hand), one single proof. In other words, he wished to have a proof in the spirit of logic and mathematics—one that avoids duplicate steps (that might occur in more than one so-called subproof)—a proof deriving the join of the targets, as Meredith did. The ANSWER literal, in various abbreviations, tells the user what specifically has been proved. You see that its argument is all, meaning all of the key targets, as you see from the literals of the corresponding clause, one each for *B*, *C*, and *K*.

The second key difference, when compared to Phase 1, was the inclusion of the command set(ancestor_subsume). The world owes much to McCune for his invention of that procedure, at least when seeking more elegant proofs. The presence of that command has the program compare, when available, two deductions of the same conclusion, preferring the strictly shorter when such exists. In other words, use of this command enables the partial automation of seeking shorter proofs. A word of warning: One can construct examples in which shorter subproofs do not necessarily make a shorter total proof. For one example, in the context of the logic under discussion, you might find a shorter proof of, say, *K*, whose use might force the proof of the join to be longer than necessary.

This second phase produced results that were promising for the future of the approach being taken. Specifically, OTTER produced a number of proofs of the join of *B*, *C*, and *K*, the first being of length 33 and the last of length 29. An inspection of the output provides an example of the phenomenon just cited, that referring to the fact that finding ever shorter proofs of members of a conjunction does not necessarily yield shorter proofs of the join of the members. In particular, the program found consecutive proofs of *B* of respective lengths 29 and 28 that, when used for a proof of the join, led to proofs of lengths 29 and 31. Just for total clarity, progress was apparently being made in that OTTER found a proof of length shorter than it had for *B*, 28 versus 29. However, the proof of the join of the three targets went from 29 to 31. This example, and the more striking generalization, strongly suggests that a simple and straightforward approach to finding ever-shorter proofs does not exist. Further, an algorithm of practical nature seems more than elusive. Nevertheless, as the following shows, progress was occurring. Indeed, before the run terminated, OTTER completed a 26-step proof of *B*. Its use led to a proof of length 29 of the join, not nearly what we were hoping for in our original excitement at (so-to-speak) competing with Meredith.

Phase 3

Two changes were made to the input file used in Phase 2. First, twenty-six weight templates were adjoined to the pick_and_purge list, each with an assigned value of 1, in contrast to those already present with an assigned value of 2. The new templates, called *resonators* (discussed in Section 3), were placed before those with assigned value 2. Each of the twenty-six corresponded to one of the 26 proof steps of the last proof of *B* found in Phase 2. The intention with the cited inclusion was to further direct the reasoning by keying on the shortest proof of *B* found so far.

The second change was to replace the assignment of the value 1 to the pick_given_ratio by a value of 2. This change causes the program to key in Phase 3 more on the complexity of deduced conclusions than in Phase 2. Indeed, the newly adjoined resonators (weight templates) would now play a more significant role than if the value assigned to the pick_given_ratio were 1.

Phase 3 was indeed rewarding. OTTER returned various proofs of *B*, including a 26-step proof; but, more important, the program also returned a proof for that member of length 22, then one of length 20. The 22-step proof led to a 28-step proof of the join of the three targets, and the 20-step proof led to a 26-step proof of the join.

Yes, I was now quite pleased. I had bested in a good sense Meredith's proof, having in hand a 26-step proof versus a 30-step proof. Could more be within reach?

Phase 4

At this point, Phase 4, I chose to rely on the *cramming strategy*, which will be exemplified with an input file later in this essay. In that strategy, the object is to force, or cram, the steps you choose into other proofs. In the case in focus, I tried to force steps of the 20-step proof into a proof of the join in such a manner that they would do double duty, triple duty, or more. The result, if all worked as planned, would be a shorter proof of the join of the three targets at the so-called cost of possibly longer proofs of either or both of *C* and *K*.

For the cramming strategy, in this instance, I placed in the initial set of support list twenty formulas, each corresponding to one step of the 20-step proof of *B* found in the preceding phase. I had the program conduct a level-saturation search, breadth first, by invoking the command set(sos_queue). I commented out the pick_given_ratio command in that it now became irrelevant. I also changed the value assigned to max_weight from 48 to 30, conjecturing that the smaller value might be required for the program to examine enough levels of clauses to complete a proof of the join. For the record, a level-0 clause is a clause in the input; a level-*n* clause is a clause at least one of whose parents has level *n*−1.

To aid you in duplicating, especially if that is your intent, what I am discussing, I now give an updated input file for Phase 4.

Input File 2, for Phase 4, Meredith and BCK

```
set(hyper_res).
assign(max_weight,30).
% assign(change_limit_after,800).
% assign(new_max_weight,22).
assign(max_proofs,-1).
clear(print_kept).
set(ancestor_subsume).
set(back_sub).
clear(print_back_sub).
clear(print_kept).
clear(print_new_demod).
clear(print_back_demod).
clear(print_back_sub).
assign(max_distinct_vars,12).
% assign(pick_given_ratio,2).
assign(max_mem,7500).
assign(report,54).
set(order_history).
set(input_sos_first).
set(sos_queue).

weight_list(pick_and_purge).
% Following 26 prove B of BCK from Meredith, temp.ulrich.bck.out2c1.
weight(P(i(i(x,i(i(i(y,i(z,u)),i(v,i(y,u))),w)),i(z,i(x,w)))),1).
weight(P(i(x,i(i(y,i(i(i(z,i(i(i(u,i(x,v)),i(w,i(u,v))),v6)),i(v7,i(z,v6))),v8)),
  i(y,v8)))),1).
weight(P(i(i(x,i(i(y,i(z,u)),v)),i(i(i(i(w,i(y,v6)),i(v7,i(w,v6))),u),i(x,v)))),1).
weight(P(i(x,i(y,i(i(z,i(x,u)),i(z,u))))),1).
weight(P(i(x,i(y,i(i(z,i(y,u)),i(z,u))))),1).
weight(P(i(i(i(i(x,i(y,z)),i(u,i(x,z))),v),i(w,i(y,v)))),1).
weight(P(i(x,i(i(y,i(x,z)),i(y,z)))),1).
weight(P(i(i(x,i(i(y,i(i(z,i(y,u)),i(z,u))),v)),i(x,v))),1).
weight(P(i(i(i(x,i(i(y,i(x,z)),i(y,z))),i(i(u,i(i(v,i(u,w)),i(v,w))),v6)),v6)),1).
weight(P(i(i(x,i(y,z)),i(y,i(x,z)))),1).
weight(P(i(i(x,i(y,z)),i(i(i(u,i(i(v,i(u,w)),i(v,w))),y),i(x,z)))),1).
weight(P(i(i(x,i(y,z)),i(i(i(u,v),y),i(v,i(x,z))))),1).
weight(P(i(i(x,i(i(i(y,i(i(z,i(y,u)),i(z,u))),v),i(w,v6)),v7)),i(i(v,v6),
  i(x,v7)))),1).
weight(P(i(i(x,y),i(z,i(i(i(u,i(i(v,i(u,w)),i(v,w))),x),y)))),1).
weight(P(i(i(x,i(i(y,i(z,u)),v)),i(u,i(x,v)))),1).
weight(P(i(i(x,y),i(i(i(z,i(i(u,i(z,v)),i(u,v))),x),y))),1).
weight(P(i(i(i(x,i(i(y,i(x,z)),i(y,z))),i(u,v)),i(i(w,i(i(v6,i(w,v7)),
  i(v6,v7))),u),v))),1).
weight(P(i(x,i(i(i(y,x),z),z))),1).
weight(P(i(i(x,i(i(i(y,i(i(z,i(y,u)),i(z,u))),v),w),v6)),i(i(v,w),i(x,v6)))),1).
weight(P(i(i(i(x,y),z),i(y,z))),1).
weight(P(i(i(x,y),i(i(z,i(y,u)),i(x,i(z,u))))),1).
weight(P(i(i(x,i(y,z)),i(i(u,y),i(u,i(x,z))))),1).
weight(P(i(i(x,i(y,i(z,u))),i(x,i(z,i(y,u))))),1).
weight(P(i(i(x,y),i(x,i(i(i(z,y),u),u)))),1).
weight(P(i(i(x,y),i(i(i(z,y),u),i(x,u)))),1).
weight(P(i(i(x,y),i(i(z,x),i(z,y)))),1).
% Following 31 include Meredith's single for BCK, and his 30-step proof.
weight(P(i(i(i(u,v),w),i(i(x,i(w,y)),i(v,i(x,y))))),2).
% Meredith's single axiom for BCK
weight(P(i(i(u,i(i(i(v,i(w,x)),i(y,i(v,x))),z)),i(w,i(u,z)))),2).
weight(P(i(u,i(i(i(v,w),x),i(w,i(i(y,i(u,z)),i(y,z)))))),2).
weight(P(i(u,i(v,i(i(u,w),i(i(x,i(v,y)),i(x,y)))))),2).
weight(P(i(u,i(i(i(i(i(v,w),x),i(i(y,i(x,z)),i(w,i(y,z)))),v6),
  i(i(v7,i(u,v8)),i(v7,v8))))),2).
weight(P(i(i(u,i(v,w)),i(x,i(i(y,i(x,z)),i(y,z))))),2).
weight(P(i(u,i(i(v,i(u,w)),i(v,w)))),2).
weight(P(i(i(u,i(i(v,i(i(w,i(v,x)),i(w,x))),y)),i(u,y))),2).
weight(P(i(i(u,i(v,w)),i(v,i(u,w)))),2).  % C
weight(P(i(i(i(u,i(i(v,i(u,w)),i(v,w))),i(i(x,i(i(y,i(x,z)),i(y,z))),v6)),v6)),2).
weight(P(i(i(u,i(v,w)),i(i(i(x,i(i(y,i(x,z)),i(y,z))),v),i(u,w)))),2).
weight(P(i(i(u,i(i(i(v,i(i(w,i(v,x)),i(w,x))),y),i(z,v6)),v7)),i(i(y,v6),
  i(u,v7)))),2).
weight(P(i(i(u,v),i(w,i(i(i(x,i(i(y,i(x,z)),i(y,z))),u),v)))),2).
weight(P(i(i(u,v),i(i(i(w,i(i(x,i(w,y)),i(x,y))),u),v))),2).
weight(P(i(i(i(u,i(i(v,i(u,w)),i(v,w))),i(x,y)),i(i(i(z,i(i(v6,i(z,v7)),
  i(v6,v7))),x),y))),2).
weight(P(i(i(u,i(i(i(i(v,i(i(w,i(v,x)),i(w,x))),y),z),v6)),i(i(y,z),i(u,v6)))),2).
weight(P(i(i(u,i(i(v,i(w,x)),y)),i(i(v,x),i(u,y)))),2).
weight(P(i(i(u,v),i(w,i(u,v)))),2).
weight(P(i(i(u,i(i(v,i(w,x)),y)),i(x,i(u,y)))),2).
weight(P(i(u,i(v,i(w,u)))),2).  % K
weight(P(i(u,i(v,u))),2). % K
weight(P(i(u,i(i(i(v,w),x),i(i(y,i(x,z)),i(w,i(y,z)))))),2).
weight(P(i(i(u,v),i(w,i(i(x,i(v,y)),i(u,i(x,y)))))),2).
weight(P(i(i(u,v),i(i(w,i(v,x)),i(u,i(w,x))))),2).
weight(P(i(i(u,i(v,w)),i(i(x,v),i(x,i(u,w))))),2).
weight(P(i(i(u,v),i(u,v))),2).
weight(P(i(i(u,v),i(u,i(i(v,w),w)))),2).
weight(P(i(i(u,v),i(i(i(u,i(i(v,w),w)),x),x))),2).
weight(P(i(i(i(u,i(i(v,w),w)),x),i(i(u,v),x))),2).
weight(P(i(i(u,v),i(i(v,w),i(u,w)))),2).  % B'
weight(P(i(i(u,v),i(i(w,u),i(w,v)))),2).  % B
end_of_list.

list(usable).
-P(i(x,y)) | -P(x) | P(y).
-P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | -P(i(a1,i(b,a1))) |
  -P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(all).  % BCK
end_of_list.

list(sos).
P(i(i(i(u,v),w),i(i(x,i(w,y)),i(v,i(x,y))))).  %  Meredith's single axiom for BCK
%  Following 26  prove, from Meredith single for BCK, B, temp.ulrich.bck.out2d1.
P(i(i(x,i(i(i(y,i(z,u)),i(v,i(y,u))),w)),i(z,i(x,w)))).
P(i(x,i(i(y,i(i(i(z,i(i(i(u,i(x,v)),i(w,i(u,v))),v6)),i(v7,i(z,v6))),v8)),
  i(y,v8)))).
P(i(x,i(y,i(i(z,i(x,u)),i(z,u))))).
P(i(x,i(y,i(i(z,i(y,u)),i(z,u))))).
P(i(x,i(i(y,i(x,z)),i(y,z)))).
P(i(i(x,i(i(y,i(i(z,i(y,u)),i(z,u))),v)),i(x,v))).
P(i(i(i(x,i(i(y,i(x,z)),i(y,z))),i(i(u,i(i(v,i(u,w)),i(v,w))),v6)),v6)).
P(i(i(x,i(y,z)),i(y,i(x,z)))).
P(i(i(x,i(y,z)),i(i(i(u,i(i(v,i(u,w)),i(v,w))),y),i(x,z)))).
P(i(i(x,i(i(i(y,i(z,u)),i(z,i(y,u))),v)),i(x,v))).
P(i(i(x,i(y,z)),i(i(i(u,v),y),i(v,i(x,z))))).
P(i(i(x,i(i(i(y,i(i(z,i(y,u)),i(z,u))),v),i(w,v6)),v7)),i(i(v,v6),i(x,v7)))).
P(i(i(x,y),i(i(z,i(i(u,y),v)),i(x,i(z,v))))).
P(i(i(x,y),i(z,i(i(i(u,i(i(v,i(u,w)),i(v,w))),x),y)))).
P(i(i(x,y),i(i(i(z,i(i(u,i(z,v)),i(u,v))),x),y))).
P(i(i(x,i(i(i(y,z),i(i(i(u,i(i(v,i(u,w)),i(v,w))),y),z)),v6)),i(x,v6))).
P(i(i(x,y),i(x,i(i(y,z),z)))).
P(i(i(x,y),i(i(i(x,i(i(y,z),z)),u),u))).
P(i(i(x,y),i(i(y,z),i(x,z)))).
P(i(i(x,y),i(i(z,x),i(z,y)))).
end_of_list.

list(passive).
%  Following negations of 4 can be proved in B,C,K from Meredith single.
-P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | $ANS(C).  %  C
-P(i(a1,i(b,a1))) | $ANS(K).  % K
-P(i(i(a1,b),i(i(b,a2),i(a1,a2)))) | $ANS(BP).  %  B'
-P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(B).  % B
end_of_list.
```

First I note that a proof of *C* is contained in the 20-step proof being used for the cramming strategy. Therefore, in the obvious sense, all that is sought in this phase is a proof of *K*. OTTER found three proofs of *K* of lengths 5, 4, and 3, each of level 3. Therefore, at least theoretically, a proof (obtainable with OTTER) existed of length 23 of the join of *B*, *C*, and *K*.

Before turning to the next phase, a bit more on cramming. The preceding experiments had consistently showed *B* to be the most difficult to prove and to require the longest proofs, when compared with the other two targets. I ordinarily cram on the most difficult to prove, especially if its proof is the longest proof of one of the members of a conjunction of two or more members. Intuitively, the longer proof offers more steps to do double, triple, or higher duty, and, so it seems, is more likely to prove useful in requiring fewer remaining steps to complete a proof of the join.

Phase 5

Surely some obvious moves would place in my hand a proof of length 23, resulting in immense satisfaction. Perhaps from haste, I returned to an earlier input file. Indeed, in this phase, the max_weight was again assigned the value 48, the pick_given_ratio was assigned the value 2, and the command set(sos_queue) was commented out (to avoid a breadth-first search). The crucial move, so I thought, was choosing the resonators to be included to direct the program's reasoning. I included two sets of resonators. The first set, consisting of twenty, corresponds to the steps of the 20-step proof of *B*, found in an earlier phase. Each was assigned the value −1, so that, when and if deduced, any formula resembling one of those proof steps would get much preference. I followed these twenty resonators with three, those corresponding to the 3-step proof of *K* yielded in the preceding phase, each assigned the value 0. My expectation was that OTTER would produce a 23-step proof of the join of the three targets, if not immediately, certainly eventually.

The experiment, as expected, produced a 7-step proof of *C* and a 20-step proof of *B*. As for *K*, the program found, in order, proofs of length 15, 13, 12, 11, 10, 9, and 8. On the surface, highly encouraging? The proofs found for the join were of respective lengths 25, 25, 27, and 26. Yes, again, you see an example of progress being made in the context of ever-shorter proofs of a member of a conjunction but, sometimes, at the cost of longer proofs of the entire conjunction.

Two questions come to mind. Most important, what happened to the expected 23-step proof of the join? Less important, but still of interest, what goes wrong with shorter subproofs? The second question can quickly be answered by noting that the shorter subproof may fail to include steps that were useful in other subproofs, hence an increase in the length of the entire proof of the conjunction. As for the more important question, I quickly guessed, with almost certainty, that too much latitude was given with the choice of max_weight, the assigned value of 48.

Phase 6, Eureka

I made but one change. I assigned max_weight the value 4, rather than 48, to virtually force OTTER to rely solely on the twenty-three steps that I was sure constituted a proof. A note of warning: Because of subsumption among other things, no guarantee exists that any proof will be found. However, a proof of the desired type is usually forthcoming. Indeed, OTTER produced the following 23-step proof of the join of *B*, *C*, and *K*.

A Fine 23-Step Proof Based on Meredith's Single Axiom for the BCK Logic

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on jaguar.mcs.anl.gov,
Mon Jun 26 18:19:27 2006
The command was "otter". The process ID is 8557.

-----> EMPTY CLAUSE at   0.06 sec ----> 89 [hyper,2,53,69,87] $ANS(all).

Length of proof is 23.  Level of proof is 15.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] -P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | -P(i(a1,i(b,a1))) |
  -P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(all).
3 [] P(i(i(i(u,v),w),i(i(x,i(w,y)),i(v,i(x,y))))).
39 [hyper,1,3,3] P(i(i(x,i(i(i(y,i(z,u)),i(v,i(y,u))),w)),i(z,i(x,w)))).
41 [hyper,1,39,39] P(i(x,i(i(y,i(i(i(z,i(i(i(u,i(x,v)),i(w,i(u,v))),v6)),
  i(v7,i(z,v6))),v8)),i(y,v8)))).
42 [hyper,1,39,41] P(i(x,i(y,i(i(z,i(x,u)),i(z,u))))).
44 [hyper,1,39,42] P(i(x,i(y,i(i(z,i(y,u)),i(z,u))))).
47 [hyper,1,44,44] P(i(x,i(i(y,i(x,z)),i(y,z)))).
49 [hyper,1,47,47] P(i(i(x,i(i(y,i(i(z,i(y,u)),i(z,u))),v)),i(x,v))).
51 [hyper,1,49,49] P(i(i(i(x,i(i(y,i(x,z)),i(y,z))),i(i(u,i(i(v,i(u,w)),
  i(v,w))),v6)),v6)).
```

```
53 [hyper,1,49,47] P(i(i(x,i(y,z)),i(y,i(x,z)))).
55 [hyper,1,3,51] P(i(i(x,i(y,z)),i(i(i(u,i(i(v,i(u,w)),i(v,w))),y),i(x,z)))).
57 [hyper,1,47,53] P(i(i(x,i(i(i(y,i(z,u)),i(z,i(y,u))),v)),i(x,v))).
58 [hyper,1,53,49] P(i(x,i(i(x,i(i(y,i(i(z,i(y,u)),i(z,u))),v)),v))).
59 [hyper,1,53,3] P(i(i(x,i(y,z)),i(i(i(u,v),y),i(v,i(x,z))))).
60 [hyper,1,3,55] P(i(i(x,i(i(i(i(y,i(i(z,i(y,u)),i(z,u))),v),i(w,v6)),v7)),
  i(i(v,v6),i(x,v7)))).
62 [hyper,1,59,58] P(i(i(i(x,y),i(z,i(i(u,i(i(v,i(u,w)),i(v,w))),v6))),
  i(y,i(z,v6)))).
63 [hyper,1,60,59] P(i(i(x,y),i(i(z,i(i(u,y),v)),i(x,i(z,v))))).
64 [hyper,1,60,58] P(i(i(x,y),i(i(i(z,i(i(u,i(z,v)),i(u,v))),x),y))).
66 [hyper,1,60,47] P(i(i(x,y),i(z,i(i(i(u,i(i(v,i(u,w)),i(v,w))),x),y)))).
68 [hyper,1,47,64] P(i(i(x,i(i(i(y,z),i(i(i(u,i(i(v,i(u,w)),i(v,w))),y),z)),
  v6)),i(x,v6))).
69 [hyper,1,62,66] P(i(x,i(y,x))).
75 [hyper,1,68,63] P(i(i(x,y),i(x,i(i(y,z),z)))).
83 [hyper,1,75,75] P(i(i(x,y),i(i(i(x,i(i(y,z),z)),u),u))).
85 [hyper,1,57,83] P(i(i(x,y),i(i(y,z),i(x,z)))).
87 [hyper,1,53,85] P(i(i(x,y),i(i(z,x),i(z,y)))).
89 [hyper,2,53,69,87] $ANS(all).
```

Retrospection and Look-Ahead

Meredith was clearly a master at finding first proofs but also at finding elegant proofs. To take one of his proofs and improve upon it is a singular achievement, not for me, but for OTTER and for the methodology. The 23-step proof just given produced, for me, much excitement of different kinds.

Of the twenty-three steps, nine are not among the thirty of the (in effect) Meredith proof. Both proofs contain one formula relying on nine distinct variables. The 23-step proof certainly punished the formula *K*. Indeed, rather than an 8-step proof, which you can find, the 23-step proof contains a 15-step proof of *K*. The proof of *C* contained in the 23-step proof has length 7, and the proof of *B* has length 20. In that the level of both the proof of *B* and the proof of the join is 15, perhaps you can improve upon the length of the proof of the join. The principle underlying this observation asserts that, when the length of a proof is reasonably (undefined) greater than its level, then some optimism is in order regarding the existence of a still shorter proof—even after methodology of the type just discussed has been employed.

As for the "look-ahead", with the given success in hand, I next wondered about the question of proof finding, in contrast to proof refining. Specifically, what might I have done if I did not have Meredith's proof to improve upon? Could I have found a proof showing that his axiom implied each of *B*, *C*, and *K*? By way of amplification, Lukasiewicz presented in the mid-1930s a 23-letter single axiom for the classical propositional calculus, but he did not offer a proof. In a book I wrote (with Gail Pieper), titled *Automated Reasoning and the Discovery of Missing and Elegant Proofs*, I included an approach that led to finding a first proof for the corresponding theorem. As far as I can learn, no proof had ever been published of the theorem, until OTTER and I considered the question. You can read of this and many other questions—as well as learning of open questions and challenges—by reading the just-cited book.

Next in order is a discussion of an approach for seeking a first proof of the Meredith theorem, *without* knowledge of his proof.

3. Proof Finding

The objective in this section is to find a so-called first proof; its length is of no concern. Overbeek believes that, although he is not personally interested in finding shorter proofs, the research with that objective can, and quite likely will, lead to methodologies for finding a first proof. The underlying principle he is adhering to, I think it fair to say, is that we need to know more about how to search the giant space of conclusions that can usually be drawn if actions are not taken to restrict and direct a program's reasoning. Robert (Bob) Veroff has made marvelous strides in automating a search for a first proof with his *sketches* *methodology*, a methodology that I have not as yet mastered. Indeed, to my knowledge, no researcher in automated reasoning comes close to the skill he exhibits in this regard. Nevertheless, more approaches are clearly of interest.

Given this preamble, here is an example of a technique for finding a first proof. The object is to show that the Meredith formula (the following), studied in Section 2, is in fact a single axiom for the *BCK* logic.

P(i(i(i(u,v),w),i(i(x,i(w,y)),i(v,i(x,y))))). % Meredith's single axiom

Immediately, you might justly wonder how this study can be made; after all, a proof is already in hand—more than one proof, as a matter of fact. The approach to be used *avoids relying on knowledge of Meredith's proof*; that is, the approach *does not* use *resonators* or *hints* taken from Meredith's proof to guide the program. A pause is, therefore, in order to briefly discuss resonators and (Veroff's) hints.

Both resonators and hints are included, if any, to direct a program's reasoning, to give preference to one or more classes of formulas or equations. Neither takes on **true** or **false** values. A resonator is treated as if the variables are indistinguishable from each other, and its assigned value is assigned to any deduced clause that matches it (ignoring the specific variables). A hint treats the variables as they are named, but you can instruct the program to assign a value to any deduced formula or equation that subsumes or is subsumed by a hint. By including a set of resonators, a set of hints, or both with small assigned values, you instruct OTTER to give preference to various deduced items when and if such are deduced. In the preceding section, you saw resonators used in the beginning that correspond to Meredith's proof, his derivation of each of B, C, and K from his single axiom. Eventually, you saw newer resonators used, corresponding to proof steps of the corresponding theorem, but where the proof was somewhat or quite different from Meredith's. The brief discussion that follows, showing that a first proof can be found *without* knowledge of Meredith's proof, presents an approach that relies on steps taken from elsewhere and the use of Veroff's hints.

Neither resonators nor hints need be theorems of the area under study. After all, as noted, they do not take on **true** or **false** values and are not treated as information describing the area under consideration or the goal to be reached. However, a visit to history suggests that, just perhaps, being theorems might be a wise choice. I have in mind the study of the Lukasiewicz 23-letter single axiom (the following) for classical propositional calculus (referred to earlier).

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

As noted, no proof was available. What was available was an attractive set of targets, namely, the three-axiom system of Lukasiewicz. More important in the context of resonators and hints was a set of sixty-eight theses (theorems) of Lukasiewicz. The approach I used keyed on theses 4 through 71 as resonators to direct OTTER in its attempt to find a first proof establishing the 23-letter formula powerful enough to derive theses 1 through 3 (the following), an axiom system for propositional calculus.

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))).

Those sixty-eight theses (the term used by Lukasiewicz) are indeed theorems of propositional calculus. What is not clear is how important is the fact that they, taken together, correspond to a quite tight line of reasoning.

What I needed for my attempt to find a first proof of the join of B, C, and K from the Meredith formula was a set of theorems of the *BCK* logic. Knowing nothing about this logic, I chose, to direct the reasoning, three sets of formulas, each corresponding to a tight line of reasoning, and each beginning with a new single axiom for the *BCK* logic found by Ulrich and sent to me by e-mail. Ulrich, as noted earlier, has found seven single axioms for this logic. As for his 31-step proof for one of them, he made no attempt to find a shorter proof; the proof was simply the first he found. In the input file I shall shortly supply, the third set corresponds to a proof also sent to me by Ulrich, a proof that derives Meredith's single axiom from the new Ulrich axiom. The second set corresponds to a proof of that same theorem found by an approach similar to that given in the preceding section, one aimed at finding a shorter proof. The first also corresponds to a proof of that theorem, an even shorter proof I found.

At this point, a reader might understandably wonder about the connection, at least syntactically, between Ulrich's single axiom and Meredith's. After all, perhaps unknowingly or unwittingly, I was using formulas to direct the program that are quite closely related to Meredith's own proof. Here is Meredith's axiom, followed by Ulrich's axiom.

```
P(i(i(i(x,y),z),i(i(u,i(z,v)),i(y,i(u,v))))).  Meredith's single axiom for BCK
P(i(i(i(x,y),z),i(y,i(i(u,i(z,v)),i(u,v))))).  % Ulrich's possibly new single axiom for BCK.
```

If I have done my homework correctly, twenty-eight formulas of the Meredith proof are *not* among the formulas I chose to use to direct OTTER's reasoning. If you would enjoy watching what occurred in my first attempt to find a first proof with OTTER with the approach just briefly discussed, here is the input file I used.

An Input File for Finding a First Proof for Meredith in the BCK Logic

```
set(hyper_res).
assign(max_weight,36).
% assign(change_limit_after,800).
% assign(new_max_weight,22).
assign(max_proofs,-1).
clear(print_kept).
% set(process_input).
% set(ancestor_subsume).
set(back_sub).
clear(print_back_sub).
clear(print_kept).
clear(print_new_demod).
clear(print_back_demod).
clear(print_back_sub).
assign(max_distinct_vars,12).
assign(pick_given_ratio,4).
assign(max_mem,7500).
assign(report,54).
set(order_history).
set(input_sos_first).
% set(sos_queue).
assign(bsub_hint_wt,1).
set(keep_hint_subsumers).
assign(heat,0).

weight_list(pick_and_purge).
end_of_list.

list(usable).
-P(i(x,y)) | -P(x) | P(y).
-P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | -P(i(a1,i(b,a1))) | -P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(all).  % BCK
% -P(i(q,i(p,q))) | -P(i(i(a,b),i(i(a,i(b,c)),i(a,c)))) | $ANS(theorem).
end_of_list.

list(sos).
P(i(i(i(u,v),w),i(i(x,i(w,y)),i(v,i(x,y))))).  % Meredith's single axiom for BCK
end_of_list.
```

```
list(passive).
-P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | $ANS(C).  %  C
-P(i(a1,i(b,a1))) | $ANS(K).  % K
-P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(B).  % B
end_of_list.

list(hints).
%  Following 20/13 prove Meredith, queue, temp.ulrich.intlog.out1e
P(i(x,i(i(y,i(i(z,i(i(u,i(x,v)),i(u,v))),w)),i(y,w)))).
P(i(i(x,i(i(y,i(i(z,i(i(i(i(u,v),w),i(v,i(i(v6,i(w,v7)),i(v6,v7)))),v8)),i(z,v8))),v9)),i(x,v9))).
P(i(x,i(y,i(z,i(i(i(u,x),v),i(i(w,i(v,v6)),i(w,v6))))))).
P(i(x,i(y,i(i(i(z,x),u),i(i(v,i(u,w)),i(v,w)))))).
P(i(x,i(i(i(y,x),z),i(i(u,i(z,v)),i(u,v))))).
P(i(i(i(x,i(y,i(i(i(z,y),u),i(i(v,i(u,w)),i(v,w))))),v6),i(i(v7,i(v6,v8)),i(v7,v8)))).
P(i(i(x,i(i(y,i(i(z,i(i(u,i(i(i(v,u),w),i(i(v6,i(w,v7)),i(v6,v7)))),v8)),i(z,v8))),v9)),i(x,v9))).
P(i(x,i(i(y,i(i(i(i(z,i(u,x)),v),i(i(w,i(v,v6)),i(w,v6))),v7)),i(y,v7)))).
P(i(i(x,i(i(i(y,i(i(z,i(i(i(u,z),v),i(i(w,i(v,v6)),i(w,v6)))),v7)),i(y,v7)),v8)),i(x,v8))).
P(i(x,i(y,i(i(i(z,i(u,x)),i(i(v,w),v6)),i(w,i(i(v7,i(v6,v8)),i(v7,v8))))))).
P(i(i(i(x,i(i(i(y,x),z),i(i(u,i(z,v)),i(u,v)))),w),i(i(v6,i(i(v7,w),v8)),i(v6,v8)))).
P(i(x,i(i(i(y,i(z,x)),i(i(u,v),w)),i(v,i(i(v6,i(w,v7)),i(v6,v7)))))).
P(i(x,i(i(y,i(i(i(z,i(i(u,x),v)),i(z,v)),w)),i(y,w)))).
P(i(x,i(y,i(i(z,i(x,u)),i(z,u))))).
P(i(x,i(i(i(i(y,i(i(z,x),u)),i(y,u)),i(i(v,i(i(i(w,v),v6),i(i(v7,i(v6,v8)),i(v7,v8)))),v9)),v9))).
P(i(i(i(x,i(y,i(z,i(u,i(i(v,i(z,w)),i(v,w)))))),i(i(v6,v7),v8)),i(v7,i(i(v9,i(v8,v1)),i(v9,v1))))).
P(i(x,i(i(y,i(x,z)),i(y,z)))).
P(i(i(i(x,y),z),i(i(u,i(i(y,i(i(v,i(z,w)),i(v,w))),v6)),i(u,v6)))).
P(i(i(x,i(i(y,i(i(z,i(y,u)),i(z,u))),v)),i(x,v))).
P(i(i(i(x,y),z),i(i(u,i(z,v)),i(y,i(u,v))))).
%  Following 22/12 prove the Meredith, from temp.ulrich.intlog.out1a.
P(i(x,i(i(y,i(i(z,i(i(u,i(x,v)),i(u,v))),w)),i(y,w)))).
P(i(i(x,i(i(y,i(i(z,i(i(i(i(u,v),w),i(v,i(i(v6,i(w,v7)),i(v6,v7)))),v8)),i(z,v8))),v9)),i(x,v9))).
P(i(x,i(y,i(z,i(i(i(u,x),v),i(i(w,i(v,v6)),i(w,v6))))))).
P(i(x,i(y,i(i(i(z,x),u),i(i(v,i(u,w)),i(v,w)))))).
P(i(x,i(i(y,i(i(z,i(u,i(i(i(v,i(w,x)),v6),i(i(v7,i(v6,v8)),i(v7,v8))))),v9)),i(y,v9)))).
P(i(x,i(i(i(y,x),z),i(i(u,i(z,v)),i(u,v))))).
P(i(x,i(y,i(z,i(i(i(u,i(v,x)),w),i(i(v6,i(w,v7)),i(v6,v7))))))).
P(i(i(i(x,i(y,i(i(i(z,y),u),i(i(v,i(u,w)),i(v,w))))),v6),i(i(v7,i(v6,v8)),i(v7,v8)))).
P(i(i(x,i(i(y,i(i(z,i(i(u,i(i(i(v,u),w),i(i(v6,i(w,v7)),i(v6,v7)))),v8)),i(z,v8))),v9)),i(x,v9))).
P(i(x,i(i(y,i(i(i(i(z,i(u,x)),v),i(i(w,i(v,v6)),i(w,v6))),v7)),i(y,v7)))).
P(i(x,i(y,i(i(i(z,i(u,x)),v),i(i(w,i(v,v6)),i(w,v6)))))).
P(i(i(x,i(i(i(y,i(i(z,i(i(i(u,z),v),i(i(w,i(v,v6)),i(w,v6)))),v7)),i(y,v7)),v8)),i(x,v8))).
P(i(x,i(y,i(i(i(z,i(u,x)),i(i(v,w),v6)),i(w,i(i(v7,i(v6,v8)),i(v7,v8))))))).
P(i(x,i(i(i(y,i(z,x)),u),i(i(v,i(u,w)),i(v,w))))).
P(i(x,i(i(i(y,i(z,x)),i(i(u,v),w)),i(v,i(i(v6,i(w,v7)),i(v6,v7)))))).
P(i(x,i(i(y,i(i(z,x),u)),i(y,u)))).
P(i(x,i(y,i(i(z,i(x,u)),i(z,u))))).
P(i(i(i(x,i(y,i(z,i(i(u,i(i(v,z),w)),i(u,w))))),i(i(v6,v7),v8)),i(v7,i(i(v9,i(v8,v1)),i(v9,v1))))).
P(i(x,i(i(y,i(x,z)),i(y,z)))).
P(i(i(i(x,y),z),i(i(u,i(i(y,i(i(v,i(z,w)),i(v,w))),v6)),i(u,v6)))).
P(i(i(x,i(i(y,i(i(z,i(y,u)),i(z,u))),v)),i(x,v))).
P(i(i(i(x,y),z),i(i(u,i(z,v)),i(y,i(u,v))))).
%  Following 31 correspond to Ulrich’s 31/12 proof of the Meredith in passive.
P(i(x,i(i(y,i(i(z,i(i(u,i(x,v)),i(u,v))),w)),i(y,w)))).
```

```
P(i(i(x,i(i(y,i(i(z,i(i(i(i(u,v),w),i(v,i(i(v6,i(w,v7)),i(v6,v7)))),v8)),i(z,v8))),v9)),i(x,v9))).
P(i(x,i(y,i(z,i(i(i(u,x),v),i(i(w,i(v,v6)),i(w,v6))))))).
P(i(x,i(y,i(i(i(z,x),u),i(i(v,i(u,w)),i(v,w)))))).
P(i(x,i(i(y,i(i(z,i(u,i(i(i(v,i(w,x)),v6),i(i(v7,i(v6,v8)),i(v7,v8))))),v9)),i(y,v9)))).
P(i(x,i(i(i(y,i(z,i(u,i(i(i(v,z),w),i(i(v6,i(w,v7)),i(v6,v7)))))),v8),i(i(v9,i(v8,v1)),i(v9,v1))))).
P(i(x,i(i(i(y,x),z),i(i(u,i(z,v)),i(u,v))))).
P(i(x,i(y,i(z,i(i(i(u,i(v,x)),w),i(i(v6,i(w,v7)),i(v6,v7))))))).
P(i(i(i(x,i(y,i(z,i(i(i(u,y),v),i(i(w,i(v,v6)),i(w,v6)))))),v7),i(i(v8,i(v7,v9)),i(v8,v9)))).
P(i(i(i(x,i(y,i(i(i(z,y),u),i(i(v,i(u,w)),i(v,w))))),v6),i(i(v7,i(v6,v8)),i(v7,v8)))).
P(i(i(x,i(i(y,i(i(z,i(i(u,i(i(i(v,u),w),i(i(v6,i(w,v7)),i(v6,v7)))),v8)),i(z,v8))),v9)),i(x,v9))).
P(i(x,i(i(y,i(i(i(i(z,i(u,x)),v),i(i(w,i(v,v6)),i(w,v6))),v7)),i(y,v7)))).
P(i(i(i(x,i(i(i(y,z),u),i(z,i(i(v,i(u,w)),i(v,w))))),v6),i(i(v7,i(v6,v8)),i(v7,v8)))).
P(i(x,i(y,i(i(i(z,i(u,x)),v),i(i(w,i(v,v6)),i(w,v6)))))).
P(i(x,i(i(y,i(i(i(z,i(x,u)),i(z,u)),v)),i(y,v)))).
P(i(i(x,i(i(i(y,i(i(z,i(i(i(u,z),v),i(i(w,i(v,v6)),i(w,v6)))),v7)),i(y,v7)),v8)),i(x,v8))).
P(i(i(i(x,y),z),i(u,i(v,i(y,i(i(w,i(z,v6)),i(w,v6))))))).
P(i(x,i(y,i(i(i(z,i(u,x)),i(i(v,w),v6)),i(w,i(i(v7,i(v6,v8)),i(v7,v8))))))).
P(i(x,i(i(i(y,i(z,x)),u),i(i(v,i(u,w)),i(v,w))))).
P(i(i(x,i(i(i(y,i(i(z,i(i(u,i(i(i(v,i(z,w)),i(v,w)),v6)),i(u,v6))),v7)),i(y,v7)),v8)),i(x,v8))).
P(i(i(i(x,i(i(i(y,x),z),i(i(u,i(z,v)),i(u,v)))),w),i(i(v6,i(i(v7,w),v8)),i(v6,v8)))).
P(i(x,i(y,i(z,i(i(u,i(i(i(v,i(z,w)),i(v,w)),v6)),i(u,v6)))))).
P(i(x,i(i(i(y,i(z,x)),i(i(u,v),w)),i(v,i(i(v6,i(w,v7)),i(v6,v7)))))).
P(i(x,i(i(y,i(i(z,x),u)),i(y,u)))).
P(i(i(x,i(i(y,i(z,i(u,i(i(v,i(i(i(w,i(u,v6)),i(w,v6)),v7)),i(v,v7))))),v8)),i(x,v8))).
P(i(x,i(y,i(i(z,i(x,u)),i(z,u))))).
P(i(i(i(x,i(y,i(z,i(i(u,i(i(v,z),w)),i(u,w))))),i(i(v6,v7),v8)),i(v7,i(i(v9,i(v8,v1)),i(v9,v1))))).
P(i(x,i(i(y,i(x,z)),i(y,z)))).
P(i(i(i(x,y),z),i(i(u,i(i(y,i(i(v,i(z,w)),i(v,w))),v6)),i(u,v6)))).
P(i(i(x,i(i(y,i(i(z,i(y,u)),i(z,u))),v)),i(x,v))).
P(i(i(i(x,y),z),i(i(u,i(z,v)),i(y,i(u,v))))).  %
end_of_list.
```

My expectation was that iteration would be required. Specifically, I thought that my first run would yield proof steps of some of the targets and that, then, I would use the proof steps in a later run as lemmas to reach the key targets, three in number. The already-cited book, published by Rinton Press, gives copious detail concerning this approach called *lemma adjunction*.

What actually occurred, however, was that the first phase yielded the required four proofs, proofs of each of *B*, *C*, and *K*, and, of course, of the join of the three in that I had included in list(usable) the negation of the join. As you would guess from the material presented in the preceding section, *C* was proved first, then *K*, then *B*, then the join. The proofs are of respective lengths 7, 11, 40, and 40. In other words, the proof of *B* contains as subproofs proofs of both *C* and *K*. I call proofs of this type *compact*, meaning that the proof of the conjunction is also a proof of one of the members of the conjunction. For the record, I find such proofs aesthetically pleasing. But, you may be growing impatient; indeed, you may wonder whether the proof that was found actually suggests that I inadvertently cheated. In particular, perhaps I relied on Meredith's proof, even though I claim that I did not. Or, perhaps the hints that I used based on Ulrich's e-mail and subsequent experimentation are closely connected to Meredith's original 30-step proof of the join of *B*, *C*, and *K*.

Well, the following data might dispel any such thoughts. Of the forty steps in the cited 40-step proof found in the approach under discussion, only eight are among Meredith's thirty. With an analysis that is clearly not deep, and might even be hasty, that data says to me that no cheating occurred, and, further, that the approach is promising for proof finding in addition to proof shortening. I now give you the proof, so that can make a more thorough study of it.

A 40-Step Proof Found with a Proof-Finding Approach

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on lemma.mcs.anl.gov,
Wed Jul  5 11:16:24 2006
The command was "otter".  The process ID is 14841.

-----> EMPTY CLAUSE at 1773.38 sec ----> 231339 [hyper,2,134,192,230005] $ANS(all).

Length of proof is 40.  Level of proof is 26.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] -P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | -P(i(a1,i(b,a1))) | -P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(all).
3 [] P(i(i(i(u,v),w),i(i(x,i(w,y)),i(v,i(x,y))))).
80 [hyper,1,3,3] P(i(i(x,i(i(i(y,i(z,u)),i(v,i(y,u))),w)),i(z,i(x,w)))).
81 [hyper,1,80,80] P(i(x,i(i(y,i(i(i(z,i(i(i(u,i(x,v)),i(w,i(u,v))),v6)),i(v7,i(z,v6))),v8)),i(y,v8)))).
101 [hyper,1,80,81] P(i(x,i(y,i(i(z,i(x,u)),i(z,u))))).
105 [hyper,1,80,101] P(i(x,i(y,i(i(z,i(y,u)),i(z,u))))).
110 [hyper,1,105,105] P(i(x,i(i(y,i(x,z)),i(y,z)))).
116 [hyper,1,110,110] P(i(i(x,i(i(y,i(i(z,i(y,u)),i(z,u))),v)),i(x,v))).
134 [hyper,1,116,110] P(i(i(x,i(y,z)),i(y,i(x,z)))).
139 [hyper,1,116,3] P(i(i(i(x,y),i(z,i(u,v))),i(y,i(u,i(z,v))))).
155 [hyper,1,110,139] P(i(i(x,i(i(i(y,z),i(u,i(v,w))),i(z,i(v,i(u,w))))),v6)),i(x,v6))).
161 [hyper,1,139,134] P(i(i(x,y),i(z,i(x,y)))).
172 [hyper,1,139,161] P(i(x,i(y,i(z,x)))).
192 [hyper,1,116,172] P(i(x,i(y,x))).
219 [hyper,1,134,192] P(i(x,i(y,y))).
220 [hyper,1,116,192] P(i(x,x)).
228 [hyper,1,134,220] P(i(x,i(i(x,y),y))).
230 [hyper,1,110,220] P(i(i(x,i(i(y,y),z)),i(x,z))).
279 [hyper,1,228,219] P(i(i(i(x,i(y,y)),z),z)).
602 [hyper,1,230,279] P(i(i(i(x,i(y,y)),i(i(z,z),u)),u)).
609 [hyper,1,230,3] P(i(i(i(x,y),z),i(y,i(i(z,u),u)))).
4925 [hyper,1,609,602] P(i(i(i(x,x),y),i(i(y,z),z))).
4969 [hyper,1,609,4925] P(i(x,i(i(i(i(x,y),y),z),z))).
4985 [hyper,1,228,4925] P(i(i(i(i(i(x,x),y),i(i(y,z),z)),u),u)).
4993 [hyper,1,134,4925] P(i(i(x,y),i(i(i(z,z),x),y))).
5000 [hyper,1,116,4925] P(i(i(i(x,x),y),i(i(z,i(y,u)),i(z,u)))).
5064 [hyper,1,134,4969] P(i(i(i(i(x,y),y),z),i(x,z))).
5405 [hyper,1,4993,4993] P(i(i(i(x,x),i(y,z)),i(i(i(u,u),y),z))).
5493 [hyper,1,4993,5064] P(i(i(i(x,x),i(i(i(y,z),z),u)),i(y,u))).
75653 [hyper,1,609,5000] P(i(x,i(i(i(i(y,i(x,z)),i(y,z)),u),u))).
80596 [hyper,1,609,5405] P(i(i(x,y),i(i(i(i(i(z,z),x),y),u),u))).
85314 [hyper,1,139,5493] P(i(i(i(i(x,y),y),i(z,u)),i(z,i(x,u)))).
166300 [hyper,1,134,75653] P(i(i(i(i(x,i(y,z)),i(x,z)),u),i(y,u))).
171816 [hyper,1,155,80596] P(i(i(x,i(y,i(z,u))),i(x,i(z,i(y,u))))).
206182 [hyper,1,166300,609] P(i(x,i(i(x,y),i(i(i(z,y),u),u)))).
206998 [hyper,1,134,206182] P(i(i(x,y),i(x,i(i(i(z,y),u),u)))).
218556 [hyper,1,171816,206998] P(i(i(x,y),i(i(i(z,y),u),i(x,u)))).
218767 [hyper,1,85314,218556] P(i(i(i(x,y),z),i(u,i(i(u,y),z)))).
226911 [hyper,1,4985,218767] P(i(x,i(i(x,y),i(i(y,z),z)))).
227344 [hyper,1,171816,226911] P(i(x,i(i(y,z),i(i(x,y),z)))).
```

```
228746 [hyper,1,134,227344] P(i(i(x,y),i(z,i(i(z,x),y)))).
230005 [hyper,1,171816,228746] P(i(i(x,y),i(i(z,x),i(z,y)))).
231339 [hyper,2,134,192,230005] $ANS(all).
```

I imagine that you may still wish for further evidence of the value of the proof-finding approach just discussed. Indeed, I suspect that some of you would like to know about the need to rely for directing the reasoning on theorems from the area of study in focus. In other words, I hypothesize that you would like to see what happens if a different logic, *L*, is studied and wonder what might occur if the resonators or hints are chosen from an area other than *L*. Actually, I wondered that myself, which brings this notebook to the next section.

4. A Brief Study of the BCI Logic

The *BCI* logic is weaker than the *BCK* logic, meaning that all theorems in the former are theorems in the latter and there exist theorems in the latter that are not theorems in the former. For an axiom system of the former, you can simply replace *K* by *I*, the following in clause notation.

```
P(i(x,x)).
```

Meredith found two single axioms for the *BCI* logic, the first of which is the following, in clause form.

```
P(i(i(x,i(y,z)),i(i(i(u,u),i(v,y)),i(v,i(x,z))))).  % M's BCI #1
```

Ulrich informed me by e-mail that he has found twenty-six additional single axioms of length 19 and that he has proved no shorter single axiom exists for the *BCI* logic. All twenty-eight known single axioms of length 19 for *BCI* are given on Ulrich's web site at http://web.ics.purdue.edu/~dulrich/BCI-page.htm.

Perhaps the following four questions will enable you to share what I experienced at this point. How easy do you think it would be for OTTER to complete a first proof, from the Meredith first single axiom for *BCI*, that derives *B*, *C*, and *I*? In what order, if successful, would the three be proved? How long would a so-called first proof be of the join of the three? And how might you approach this bit of research?

If you examine the following input file, you can answer the last question. If you run the input file with OTTER, or simply read the material following the file, you will answer the other three questions.

Input File for Studying BCI, Meredith 1, Single Axiom

```
set(hyper_res).
assign(max_weight,48).
% assign(change_limit_after,800).
% assign(new_max_weight,22).
assign(max_proofs,-1).
clear(print_kept).
%  set(ancestor_subsume).
set(back_sub).
% clear(for_sub).
clear(print_back_sub).
clear(print_kept).
clear(print_new_demod).
clear(print_back_demod).
clear(print_back_sub).
assign(max_distinct_vars,12).
assign(pick_given_ratio,1).
assign(max_mem,7500).
assign(report,54).
set(order_history).
set(input_sos_first).
```

```
% set(sos_queue).
assign(bsub_hint_wt,1).
set(keep_hint_subsumers).

weight_list(pick_and_purge).
% Following 31 include Meredith's single for BCK, and his 30-step proof.
weight(P(i(i(i(u,v),w),i(i(x,i(w,y)),i(v,i(x,y))))),2). % Meredith's single axiom for BCK
weight(P(i(i(u,i(i(i(v,i(w,x)),i(y,i(v,x))),z)),i(w,i(u,z)))),2).
weight(P(i(u,i(i(i(v,w),x),i(w,i(i(y,i(u,z)),i(y,z)))))),2).
weight(P(i(u,i(v,i(i(u,w),i(i(x,i(v,y)),i(x,y)))))),2).
weight(P(i(u,i(i(i(i(i(v,w),x),i(i(y,i(x,z)),i(w,i(y,z)))),v6),i(i(v7,i(u,v8)),i(v7,v8))))),2).
weight(P(i(i(u,i(v,w)),i(x,i(i(y,i(x,z)),i(y,z))))),2).
weight(P(i(u,i(i(v,i(u,w)),i(v,w)))),2).
weight(P(i(i(u,i(i(v,i(i(w,i(v,x)),i(w,x))),y)),i(u,y))),2).
weight(P(i(i(u,i(v,w)),i(v,i(u,w)))),2). % C
weight(P(i(i(i(u,i(i(v,i(u,w)),i(v,w))),i(i(x,i(i(y,i(x,z)),i(y,z))),v6)),v6)),2).
weight(P(i(i(u,i(v,w)),i(i(i(x,i(i(y,i(x,z)),i(y,z))),v),i(u,w)))),2).
weight(P(i(i(u,i(i(i(i(v,i(i(w,i(v,x)),i(w,x))),y),i(z,v6)),v7)),i(i(y,v6),i(u,v7)))),2).
weight(P(i(i(u,v),i(w,i(i(i(x,i(i(y,i(x,z)),i(y,z))),u),v)))),2).
weight(P(i(i(u,v),i(i(i(w,i(i(x,i(w,y)),i(x,y))),u),v))),2).
weight(P(i(i(i(u,i(i(v,i(u,w)),i(v,w))),i(x,y)),i(i(i(z,i(i(v6,i(z,v7)),i(v6,v7))),x),y))),2).
weight(P(i(i(u,i(i(i(v,i(i(w,i(v,x)),i(w,x))),y),z),v6)),i(i(y,z),i(u,v6)))),2).
weight(P(i(i(u,i(i(v,i(w,x)),y)),i(i(v,x),i(u,y)))),2).
weight(P(i(i(u,v),i(w,i(u,v)))),2).
weight(P(i(i(u,i(i(v,i(w,x)),y)),i(x,i(u,y)))),2).
weight(P(i(u,i(v,i(w,u)))),2).
weight(P(i(u,i(v,u))),2). % K
weight(P(i(u,i(i(i(v,w),x),i(i(y,i(x,z)),i(w,i(y,z)))))),2).
weight(P(i(i(u,v),i(w,i(i(x,i(v,y)),i(u,i(x,y)))))),2).
weight(P(i(i(u,v),i(i(w,i(v,x)),i(u,i(w,x))))),2).
weight(P(i(i(u,i(v,w)),i(i(x,v),i(x,i(u,w))))),2).
weight(P(i(i(u,v),i(u,v))),2).
weight(P(i(i(u,v),i(u,i(i(v,w),w)))),2).
weight(P(i(i(u,v),i(i(i(u,i(i(v,w),w)),x),x))),2).
weight(P(i(i(i(u,i(i(v,w),w)),x),i(i(u,v),x))),2).
weight(P(i(i(u,v),i(i(v,w),i(u,w)))),2). % B'
weight(P(i(i(u,v),i(i(w,u),i(w,v)))),2). % B
end_of_list.

list(usable).
-P(i(x,y)) | -P(x) | P(y).
-P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | -P(i(a1,a1)) | -P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(all). % BCI
end_of_list.

list(sos).
P(i(i(x,i(y,z)),i(i(i(u,u),i(v,y)),i(v,i(x,z))))). % M's BCI #1
end_of_list.

list(passive).
% Following is neg of I, of BCI.
-P(i(a,a)) | $ANS(I).
% Following prove B,C,K from Meredith single.
-P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | $ANS(C). % C
```

```
-P(i(a1,i(b,a1))). % K
% -P(i(i(a1,b),i(i(b,a2),i(a1,a2)))) | $ANS(BP). % B'
-P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(B). % B
end_of_list.
```

The given input file is the one I used in my attack on finding a proof that began with Meredith 1, his first of two single axioms for the *BCI* logic. I emphasize that I had never seen a proof of the corresponding theorem. As you see from the input file, the goal was to prove each of *B*, *C*, and *I* and, of course, prove the join of the three. I assumed, or guessed, that a series of runs would be required, each run bringing forth new results whose proof steps could be used as lemmas in the next run.

Well, such was indeed not the case. In the first run, OTTER quickly proved each of the three targets, as well as the conjunction. To my amusement, *B* was not the last of the three individual targets reached. They were proved in the order *C*, *B*, and *I*, counterintuitive to me (but, as is obvious, I know little about this area of logic). The proofs are, respectively, of length 24, 3, and 26. The proof of the join (conjunction) has length 38, and its level is 14. Because of my view that shorter proofs are far more likely to be discovered when the distance between the length and level is substantial, I immediately was almost certain that a far shorter proof must exist. Before discussing that aspect of my research, I note that the four questions have been answered. Here is the proof OTTER found.

A 38-Step Proof Based on Meredith's First Single Axiom for BCI

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on jaguar.mcs.anl.gov,
Fri Jul  7 09:45:37 2006
The command was "otter".  The process ID is 20988.

-----> EMPTY CLAUSE at   4.64 sec ----> 11083 [hyper,2,4119,10888,6824] $ANS(all).

Length of proof is 38.  Level of proof is 17.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] -P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | -P(i(a1,a1)) | -P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(all).
3 [] P(i(i(x,i(y,z)),i(i(u,u),i(v,y)),i(v,i(x,z))))).
8 [hyper,1,3,3] P(i(i(i(x,x),i(y,i(z,z),i(u,v)))),i(y,i(i(w,i(v,v6)),i(u,i(w,v6)))))).
9 [hyper,1,8,8] P(i(i(i(x,x),i(y,z)),i(i(u,i(i(z,v),v),w)),i(y,i(u,w))))).
11 [hyper,1,8,3] P(i(i(i(x,x),i(i(y,y),z)),i(i(u,i(v,w)),i(i(z,v),i(u,w))))).
12 [hyper,1,9,9] P(i(i(x,i(i(i(i(y,i(z,u)),v),v),w)),i(z,i(i(i(y,v6),v6),u)),i(x,w)))).
13 [hyper,1,8,9] P(i(i(x,i(i(i(i(y,y),z),z),u)),i(i(v,i(u,w)),i(x,i(v,w))))).
19 [hyper,1,8,11] P(i(i(x,i(i(y,y),z)),i(i(u,i(z,v)),i(x,i(u,v))))).
23 [hyper,1,11,3] P(i(i(x,i(y,z)),i(i(i(u,i(i(u,v),v)),y),i(x,z)))).
38 [hyper,1,19,3] P(i(i(x,i(i(y,i(z,u)),v)),i(i(z,i(y,u)),i(x,v)))).
44 [hyper,1,8,12] P(i(i(x,i(i(i(y,z),z),u)),i(i(v,i(w,v6)),i(i(i(y,i(x,u)),w),i(v,v6))))).
51 [hyper,1,23,23] P(i(i(i(x,i(i(x,y),y)),i(i(z,i(i(z,u),u)),v)),i(i(w,i(v,v6)),i(w,v6)))).
52 [hyper,1,19,23] P(i(i(x,i(i(y,z),u)),i(i(y,i(i(v,i(i(v,w),w)),z)),i(x,u)))).
81 [hyper,1,38,38] P(i(i(x,i(y,z)),i(i(u,i(i(x,i(y,z)),v)),i(u,v)))).
87 [hyper,1,8,38] P(i(i(x,i(y,z)),i(i(u,i(i(x,z),v)),i(y,i(u,v))))).
90 [hyper,1,38,19] P(i(i(x,i(y,z)),i(i(u,i(i(v,v),x)),i(u,i(y,z))))).
117 [hyper,1,38,81] P(i(i(i(x,i(y,z)),i(u,v)),i(i(x,i(y,z)),i(u,v)))).
161 [hyper,1,38,87] P(i(i(i(x,y),i(z,u)),i(i(x,i(v,y)),i(v,i(z,u))))).
209 [hyper,1,38,90] P(i(i(i(x,x),i(y,z)),i(i(z,i(u,v)),i(y,i(u,v))))).
270 [hyper,1,38,117] P(i(i(x,i(y,z)),i(i(i(y,i(x,z)),i(u,v)),i(u,v)))).
```

```
280 [hyper,1,8,117] P(i(i(i(x,x),i(y,z)),i(i(u,i(z,v)),i(y,i(u,v))))).
321 [hyper,1,38,161] P(i(i(x,i(y,z)),i(i(i(y,z),i(u,v)),i(x,i(u,v))))).
398 [hyper,1,209,161] P(i(i(i(x,i(y,z)),i(u,v)),i(i(y,i(x,z)),i(u,v)))).
472 [hyper,1,13,270] P(i(i(x,i(i(y,z),u)),i(i(y,i(i(v,v),z)),i(x,u)))).
635 [hyper,1,280,209] P(i(i(x,i(i(y,i(z,u)),v)),i(i(y,i(z,u)),i(x,v)))).
789 [hyper,1,398,321] P(i(i(x,i(y,z)),i(i(i(x,z),i(u,v)),i(y,i(u,v))))).
1049 [hyper,1,635,472] P(i(i(x,i(i(y,y),z)),i(i(u,i(i(x,z),v)),i(u,v)))).
2697 [hyper,1,44,270] P(i(i(x,i(y,z)),i(i(i(u,i(i(v,i(u,w)),i(v,w))),y),i(x,z)))).
4017 [hyper,1,51,1049] P(i(i(x,i(i(y,y),z)),i(x,z))).
4108 [hyper,1,4017,2697] P(i(i(x,i(i(y,i(i(z,i(y,u)),i(z,u))),v)),i(x,v))).
4119 [hyper,1,4017,789] P(i(i(x,i(y,z)),i(y,i(x,z)))).
4242 [hyper,1,4108,4108] P(i(i(i(x,i(i(y,i(x,z)),i(y,z))),i(i(u,i(i(v,i(u,w)),i(v,w))),v6)),v6)).
4435 [hyper,1,8,4119] P(i(i(x,y),i(i(z,i(y,u)),i(x,i(z,u))))).
4706 [hyper,1,4242,52] P(i(x,i(i(x,y),y))).
4968 [hyper,1,4017,4435] P(i(i(x,y),i(x,i(i(y,z),z)))).
5360 [hyper,1,4968,4968] P(i(i(x,y),i(i(i(x,i(i(y,z),z)),u),u))).
5909 [hyper,1,4119,5360] P(i(i(i(x,i(i(y,z),z)),u),i(i(x,y),u))).
6342 [hyper,1,5909,4119] P(i(i(x,y),i(i(y,z),i(x,z)))).
6824 [hyper,1,4119,6342] P(i(i(x,y),i(i(z,x),i(z,y)))).
10888 [hyper,1,4017,4706] P(i(x,x)).
11083 [hyper,2,4119,10888,6824] $ANS(all).
```

Although you might be curious about how short a proof I was able to discover, I suspect that you are more interested in how much so-called cheating occurred, inadvertently or otherwise. Of the 31 formulas, Meredith's single axiom for the *BCK* logic and his 30-step proof, just ten are present in the 38-step proof just given. Further, if all variables in the thirty-one and in the thirty-eight are changed to the variable *x*, still only ten of the thirty-one are among the modified thirty-eight. What I am getting at is evidence that the choice of the thirty-one resonators was crucial, I believe, but the attack was not merely and simply based on Meredith's proof for the *BCK* logic. Indeed, even as resonators, only ten of the thirty-one are among the thirty-eight proof steps. As for another comment on the ease of proof, just over 11000 clauses were retained when the proof of the join was completed, which is nothing in the country of OTTER. In other words, rather than building directly on Meredith's study of *BCK*, I suggest that you have evidence that the theorems of one area of logic can be profitably used to study a second area. The following revisiting of the two axioms may aid you in seeing in part why this might be and why the attack was not merely an extension of Meredith's *BCK* study.

```
P(i(i(x,i(y,z)),i(i(i(u,u),i(v,y)),i(v,i(x,z))))).  % M's BCI #1 for BCI
P(i(i(i(x,y),z),i(i(u,i(z,v)),i(y,i(u,v))))).  % M's single axiom for BCK
```

When I turned to refining the 38-step proof, relying, as expected, on methodology closely related to that offered in Section 2, I eventually obtained the following 14-step proof.

A 14-Step Proof in BCI for Meredith's First Single Axiom

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on lemma.mcs.anl.gov,
Sun Jul  2 13:15:06 2006
The command was "otter".  The process ID is 9915.

-----> EMPTY CLAUSE at  54.99 sec ----> 35462 [hyper,2,185,35340,30085] $ANS(all).

Length of proof is 14.  Level of proof is 12.

---------------- PROOF ----------------
```

```
1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] -P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | -P(i(a1,a1)) | -P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(all).
3 [] P(i(i(x,i(y,z)),i(i(i(u,u),i(v,y)),i(v,i(x,z))))).
12 [hyper,1,3,3] P(i(i(i(x,x),i(y,i(i(z,z),i(u,v)))),i(y,i(i(w,i(v,v6)),i(u,i(w,v6)))))).
15 [hyper,1,12,3] P(i(i(i(x,x),i(i(y,y),z)),i(i(u,i(v,w)),i(i(z,v),i(u,w))))).
21 [hyper,1,15,15] P(i(i(x,i(y,z)),i(i(i(i(u,u),v),i(i(v,w),w)),y),i(x,z)))).
36 [hyper,1,21,21] P(i(i(i(i(i(x,x),y),i(i(y,z),z)),i(i(i(u,u),v),i(i(v,w),w)),v6)),i(i(v7,i(v6,v8)),i(v7,v8)))).
89 [hyper,1,36,36] P(i(i(x,i(i(i(i(y,y),i(i(z,z),u)),u),v)),i(x,v))).
160 [hyper,1,89,89] P(i(i(i(i(i(x,x),i(i(y,y),z)),z),i(i(i(u,u),i(i(v,v),w)),w),v6)),v6)).
185 [hyper,1,89,3] P(i(i(x,i(y,z)),i(y,i(x,z)))).
327 [hyper,1,185,185] P(i(x,i(i(y,i(x,z)),i(y,z)))).
621 [hyper,1,3,327] P(i(i(i(x,x),i(y,i(z,i(u,v)))),i(y,i(u,i(z,v))))).
874 [hyper,1,621,3] P(i(i(i(x,x),i(y,z)),i(i(z,u),i(y,u)))).
26237 [hyper,1,874,185] P(i(i(i(i(x,y),y),z),i(x,z))).
29772 [hyper,1,26237,874] P(i(i(x,y),i(i(y,z),i(x,z)))).
30085 [hyper,1,185,29772] P(i(i(x,y),i(i(z,x),i(z,y)))).
35340 [hyper,1,160,160] P(i(x,x)).
35462 [hyper,2,185,35340,30085] $ANS(all).
```

At this point, I depart from what might be expected. In particular, rather than detailing my approach that resulted in the given 14-step proof, I invite you to attempt to find it. Even better, I invite you to find a proof of length strictly less than fourteen (applications of condensed detachment). That may be hard, if you share my view that, when the level and length are close together, further shortening is made quite difficult. If you wish a quite different research topic, you might attempt to explain why such an observation holds.

But how effective is the approach? For example, what would occur if we changed single axioms, replacing the Meredith 1 with one of Ulrich's (sent to me by e-mail), the following?

```
P(i(i(i(i(i(x,x),i(y,z)),z),u),i(i(u,v),i(y,v)))).  % Ulrich BCI #3
%  P(i(i(x,i(y,z)),i(i(i(u,u),i(v,y)),i(v,i(x,z))))).  % M's BCI #1
```

I included, immediately after Ulrich 3, the Meredith single axiom (commented out) for *BCI* to make easier a comparison with the Ulrich candidate. You see that their structure differs sharply. Then, what would occur if we added to the targets in the preceding experiment the negation of the Meredith axiom, the following?

```
-P(i(i(a1,i(a2,a3)),i(i(i(a4,a4),i(a5,a2)),i(a5,i(a1,a3))))) | $ANS(M1).
```

In other words, called for is an experiment designed to prove that the Ulrich candidate formula is in fact a single axiom for the *BCI* logic and, in the same experiment (or set of experiments, if needed) to obtain a derivation of the Meredith 1.

I took a shortcut, beginning by making small modifications to the input file used to study Meredith 1 (as sole hypothesis), given earlier in this section. As you quite likely have surmised, especially from the pair of formulas just presented, I placed in list(sos) the Ulrich candidate, commented out Meredith 1 (to prevent it from participating in any manner), and added to list(passive) the negation of Meredith 1. I made one additional modification, that designed to have OTTER stop if and when five proofs had been completed, one for each of *B*, *C*, and *I*, one for the conjunction of the three, and (finally) one for a derivation of Meredith 1. In that precisely five proofs are sought, the following command replaced its correspondent in the new and slightly modified input file.

```
assign(max_proofs,5).
```

Except for the cited changes and additions, the preceding input file suffices.

Can you guess what would occur, how many experiments would be required to discover the sought-after five proofs? How would the four questions, similar to the four asked for the preceding study (of Meredith 1) be answered?

The gold was quickly mined—all five proofs completed—in approximately the same time as required for the preceding study, roughly 5 CPU seconds. As with the Meredith single axiom, *C* was proved first, then *B*, then *I*, then the join of the three, almost immediately; a few CPU seconds later, Meredith 1 was proved. In order, the proof lengths are 5, 9, 7, 15, and 21. The respective levels are 4, 5, 7, 7, and 10. And now I repeat myself more or less. Because the distance between proof length and proof level is so large, in the last two cases, you know I would expect that far shorter proofs exist, at least for the join of the three and for Meredith 1. But, before turning to that bit of research, I give two proofs (obtained in the just-described experiment) of possible interest, followed by some commentary about their respective nature.

A 15-Step Proof of the Join in BCI from Ulrich 3

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on jaguar.mcs.anl.gov,
Sun Jul  9 20:32:59 2006
The command was "otter". The process ID is 19934.
-----> EMPTY CLAUSE at   0.22 sec ----> 1855 [hyper,2,385,1788,526] $ANS(all).

Length of proof is 15. Level of proof is 7.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] -P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | -P(i(a1,a1)) | -P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(all).
3 [] P(i(i(i(i(i(x,x),i(y,z)),z),u),i(i(u,v),i(y,v)))).
9 [hyper,1,3,3] P(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u))).
10 [hyper,1,9,9] P(i(x,i(y,i(i(x,i(z,u)),i(i(y,z),u))))).
11 [hyper,1,3,9] P(i(i(i(x,y),z),i(i(x,y),z))).
12 [hyper,1,9,3] P(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v)))).
20 [hyper,1,3,11] P(i(i(i(i(i(x,x),i(y,z)),z),u),i(y,u))).
21 [hyper,1,12,12] P(i(i(i(i(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v))),i(i(w,w),i(v6,v7))),v7),v8),i(v6,v8))).
24 [hyper,1,3,12] P(i(i(i(i(i(i(i(i(i(x,x),i(y,z)),z),i(i(u,u),i(v,w))),w),v6),i(v,v6)),v7),i(y,v7))).
26 [hyper,1,12,10] P(i(i(i(i(i(x,i(y,i(i(x,i(z,u)),i(i(y,z),u)))),i(i(v,v),i(w,v6))),v6),v7),i(w,v7))).
50 [hyper,1,20,20] P(i(x,i(i(x,y),y))).
99 [hyper,1,50,50] P(i(i(i(x,i(i(x,y),y)),z),z)).
385 [hyper,1,24,21] P(i(i(x,i(y,z)),i(y,i(x,z)))).
447 [hyper,1,26,3] P(i(i(x,y),i(i(y,z),i(x,z)))).
526 [hyper,1,385,447] P(i(i(x,y),i(i(z,x),i(z,y)))).
1743 [hyper,1,20,99] P(i(i(i(x,x),y),y)).
1788 [hyper,1,1743,1743] P(i(x,x)).
1855 [hyper,2,385,1788,526] $ANS(all).
```

A 21-Step Proof in BCI of Meredith 1 From Ulrich 3

```
----> UNIT CONFLICT at   4.64 sec ----> 13159 [binary,13158.1,4.1] $ANS(M1).

Length of proof is 21. Level of proof is 10.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
3 [] P(i(i(i(i(i(x,x),i(y,z)),z),u),i(i(u,v),i(y,v)))).
4 [] -P(i(i(a1,i(a2,a3)),i(i(i(a4,a4),i(a5,a2)),i(a5,i(a1,a3))))) | $ANS(M1).
```

```
9 [hyper,1,3,3] P(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u))).
10 [hyper,1,9,9] P(i(x,i(y,i(i(x,i(z,u)),i(i(y,z),u))))).
11 [hyper,1,3,9] P(i(i(i(x,y),z),i(i(x,y),z))).
12 [hyper,1,9,3] P(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v)))).
20 [hyper,1,3,11] P(i(i(i(i(i(x,x),i(y,z)),z),u),i(y,u))).
21 [hyper,1,12,12] P(i(i(i(i(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v))),i(i(w,w),i(v6,v7))),v7),v8),i(v6,v8))).
24 [hyper,1,3,12] P(i(i(i(i(i(i(i(i(i(x,x),i(y,z)),z),i(i(u,u),i(v,w))),w),v6),i(v,v6)),v7),i(y,v7))).
26 [hyper,1,12,10] P(i(i(i(i(i(x,i(y,i(i(x,i(z,u)),i(i(y,z),u)))),i(i(v,v),i(w,v6))),v6),v7),i(w,v7))).
50 [hyper,1,20,20] P(i(x,i(i(x,y),y))).
56 [hyper,1,3,20] P(i(i(i(i(x,y),y),z),i(x,z))).
385 [hyper,1,24,21] P(i(i(x,i(y,z)),i(y,i(x,z)))).
441 [hyper,1,26,56] P(i(i(x,i(y,z)),i(x,i(i(i(u,u),y),z)))).
447 [hyper,1,26,3] P(i(i(x,y),i(i(y,z),i(x,z)))).
524 [hyper,1,447,447] P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))).
526 [hyper,1,385,447] P(i(i(x,y),i(i(z,x),i(z,y)))).
928 [hyper,1,526,50] P(i(i(x,y),i(x,i(i(y,z),z)))).
1351 [hyper,1,447,928] P(i(i(i(x,i(i(y,z),z)),u),i(i(x,y),u))).
10190 [hyper,1,524,524] P(i(i(x,i(y,z)),i(i(u,y),i(x,i(u,z))))).
10507 [hyper,1,1351,10190] P(i(i(x,y),i(i(z,i(y,u)),i(x,i(z,u))))).
11651 [hyper,1,385,10507] P(i(i(x,i(y,z)),i(i(u,y),i(u,i(x,z))))).
13158 [hyper,1,441,11651] P(i(i(x,i(y,z)),i(i(i(u,u),i(v,y)),i(v,i(x,z))))).
```

As for reliance on the thirty-one formulas Meredith used for showing his single axiom capable of deriving each of *B*, *C*, and *K*, but three of the thirty-one are present in the fifteen steps proving the join from Ulrich 3. The three are, respectively, *C*, *B*', and *B*; the second is closely related to the third. Seven of the thirty-one are present in the 21-step proof deriving, from Ulrich 3, Meredith 1. I conjecture that more substantial evidence has just been given of the power of using formulas from one study to successfully direct a program's reasoning in finding a so-called first proof in a different, though related, area of logic. At least when proving the join of *B*, *C*, and *I*, a natural conjecture asserts that the thirty-one should play a big role, or could; after all, two of the targets in the two studies—that with Meredith 1 as hypothesis and that with Ulrich 3 as hypothesis—are shared, *B* and *C*. However, so the data suggests, the two shared targets do not cause the program to heavily rely on much of the set of thirty-one formulas.

Next in order here is commentary on proof shortening in the context of Ulrich 3 and the given 15-step and 21-step proofs. An effective move to make when seeking shorter proofs when one has found so-called first proof(s) is to invoke the command ancestor_subsume. The inclusion of that command causes OTTER to compare proof lengths when two paths lead to the same deduced conclusion, preferring the strictly shorter. When I included that command, without making any additional changes to the input file in use, the program found a 13-step level-6 proof of the join and a 12-step level-7 proof of Meredith 1. I was somewhat pleased at the reduction in proof length from 15 to 13, and very pleased at the reduction from 21 to 12.

I next tried a level-saturation, breadth-first, search, motivated by the fact that the levels of the two found proofs are not too high. Here is the input file I used; note that I replaced the 31 resonators corresponding to the Meredith proof for his axiom in *BCK* with resonators corresponding to the 13-step and 12-step cited proofs.

An Input File Illustrating Level Saturation

```
set(hyper_res).
assign(max_weight,36).
assign(change_limit_after,800).
assign(new_max_weight,22).
assign(max_proofs,-1).
```

```
clear(print_kept).
set(ancestor_subsume).
set(back_sub).
% clear(for_sub).
clear(print_back_sub).
clear(print_kept).
clear(print_new_demod).
clear(print_back_demod).
clear(print_back_sub).
assign(max_distinct_vars,12).
%  assign(pick_given_ratio,4).
assign(max_mem,7500).
assign(report,54).
set(order_history).
set(input_sos_first).
set(sos_queue).
assign(bsub_hint_wt,1).
set(keep_hint_subsumers).

weight_list(pick_and_purge).
%  Following 13 prove from Ulrich 3 the join of B C I, temp.ulrich.bci.out2b.
weight(P(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u))),-2).
weight(P(i(i(i(x,y),z),i(i(x,y),z))),-2).
weight(P(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v)))),-2).
weight(P(i(i(i(i(i(x,x),i(y,z)),z),u),i(y,u))),-2).
weight(P(i(i(i(i(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v))),i(i(w,w),i(v6,v7))),v7),v8),i(v6,v8))),-2).
weight(P(i(i(i(i(i(i(i(i(i(x,x),i(y,z)),z),i(i(u,u),i(v,w))),w),v6),i(v,v6)),v7),i(y,v7))),-2).
weight(P(i(i(i(i(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u)),i(i(v,v),i(w,v6))),v6),v7),i(w,v7))),-2).
weight(P(i(x,i(i(i(y,y),i(x,z)),z))),-2).
weight(P(i(i(i(x,x),i(i(y,i(i(i(z,z),i(y,u)),u)),v)),v)),-2).
weight(P(i(i(x,i(y,z)),i(y,i(x,z)))),-2).
weight(P(i(i(i(x,i(i(x,y),z)),u),i(i(y,z),u))),-2).
weight(P(i(i(x,y),i(i(z,x),i(z,y)))),-2).
weight(P(i(x,x)),-2).
%  Following 12 prove Meredith1 from Ulrich 3, temp.ulrich.bci.out2b
weight(P(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u))),0).
weight(P(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v)))),0).
weight(P(i(i(i(i(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v))),i(i(w,w),i(v6,v7))),v7),v8),i(v6,v8))),0).
weight(P(i(i(i(i(i(i(i(i(i(x,x),i(y,z)),z),i(i(u,u),i(v,w))),w),v6),i(v,v6)),v7),i(y,v7))),0).
weight(P(i(i(i(i(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u)),i(i(v,v),i(w,v6))),v6),v7),i(w,v7))),0).
weight(P(i(i(x,i(y,z)),i(y,i(x,z)))),0).
weight(P(i(i(x,y),i(i(y,z),i(x,z)))),0).
weight(P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))),0).
weight(P(i(i(i(i(i(i(x,y),i(i(y,z),i(x,z))),i(i(u,u),i(v,w))),w),v6),i(v,v6))),0).
weight(P(i(i(i(x,i(y,z)),u),i(i(y,i(x,z)),u))),0).
weight(P(i(i(x,y),i(i(i(z,z),i(u,x)),i(u,y)))),0).
weight(P(i(i(x,i(y,z)),i(i(i(u,u),i(v,y)),i(v,i(x,z))))),0).
%  %  Following 31 include Meredith's single for BCK, and his 30-step proof.
%  weight(P(i(i(i(u,v),w),i(i(x,i(w,y)),i(v,i(x,y))))),2).  %  Meredith's single axiom for BCK
%  weight(P(i(i(u,i(i(i(v,i(w,x)),i(y,i(v,x))),z)),i(w,i(u,z)))),2).
%  weight(P(i(u,i(i(i(v,w),x),i(w,i(i(y,i(u,z)),i(y,z)))))),2).
%  weight(P(i(u,i(v,i(i(u,w),i(i(x,i(v,y)),i(x,y)))))),2).
%  weight(P(i(u,i(i(i(i(v,w),x),i(i(y,i(x,z)),i(w,i(y,z)))),v6),i(i(v7,i(u,v8)),i(v7,v8))))),2).
% weight(P(i(i(u,i(v,w)),i(x,i(i(y,i(x,z)),i(y,z))))),2).
% weight(P(i(u,i(i(v,i(u,w)),i(v,w)))),2).
% weight(P(i(i(u,i(i(v,i(i(w,i(v,x)),i(w,x))),y)),i(u,y))),2).
% weight(P(i(i(u,i(v,w)),i(v,i(u,w)))),2). % C
% weight(P(i(i(i(u,i(i(v,i(u,w)),i(v,w))),i(i(x,i(i(y,i(x,z)),i(y,z))),v6)),v6)),2).
% weight(P(i(i(u,i(v,w)),i(i(i(x,i(i(y,i(x,z)),i(y,z))),v),i(u,w)))),2).
% weight(P(i(i(u,i(i(i(i(v,i(i(w,i(v,x)),i(w,x))),y),i(z,v6)),v7)),i(i(y,v6),i(u,v7)))),2).
% weight(P(i(i(u,v),i(w,i(i(i(x,i(i(y,i(x,z)),i(y,z))),u),v)))),2).
% weight(P(i(i(u,v),i(i(i(w,i(i(x,i(w,y)),i(x,y))),u),v))),2).
% weight(P(i(i(i(u,i(i(v,i(u,w)),i(v,w))),i(x,y)),i(i(i(z,i(i(v6,i(z,v7)),i(v6,v7))),x),y))),2).
% weight(P(i(i(u,i(i(i(i(v,i(i(w,i(v,x)),i(w,x))),y),z),v6)),i(i(y,z),i(u,v6)))),2).
% weight(P(i(i(u,i(i(v,i(w,x)),y)),i(i(v,x),i(u,y)))),2).
% weight(P(i(i(u,v),i(w,i(u,v)))),2).
% weight(P(i(i(u,i(i(v,i(w,x)),y)),i(x,i(u,y)))),2).
% weight(P(i(u,i(v,i(w,u)))),2). % K
% weight(P(i(u,i(v,u))),2). % K
% weight(P(i(u,i(i(i(v,w),x),i(i(y,i(x,z)),i(w,i(y,z)))))),2).
% weight(P(i(i(u,v),i(w,i(i(x,i(v,y)),i(u,i(x,y)))))),2).
% weight(P(i(i(u,v),i(i(w,i(v,x)),i(u,i(w,x))))),2).
% weight(P(i(i(u,i(v,w)),i(i(x,v),i(x,i(u,w))))),2).
% weight(P(i(i(u,v),i(u,v))),2).
% weight(P(i(i(u,v),i(u,i(i(v,w),w)))),2).
% weight(P(i(i(u,v),i(i(i(u,i(i(v,w),w)),x),x))),2).
% weight(P(i(i(i(u,i(i(v,w),w)),x),i(i(u,v),x))),2).
% weight(P(i(i(u,v),i(i(v,w),i(u,w)))),2). % B'
% weight(P(i(i(u,v),i(i(w,u),i(w,v)))),2). % B
end_of_list.

list(usable).
-P(i(x,y)) | -P(x) | P(y).
-P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | -P(i(a1,a1)) | -P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(all). % BCI
end_of_list.

list(sos).
P(i(i(i(i(i(x,x),i(y,z)),z),u),i(i(u,v),i(y,v)))). % Ulrich BCI #3
% P(i(i(x,i(y,z)),i(i(i(u,u),i(v,y)),i(v,i(x,z))))). % M's BCI #1
end_of_list.

list(passive).
% Following is neg of Meredith 1, single axiom for BCI.
-P(i(i(a1,i(a2,a3)),i(i(i(a4,a4),i(a5,a2)),i(a5,i(a1,a3))))) | $ANS(M1).
% Following is neg of I, of BCI.
-P(i(a,a)) | $ANS(I).
% Following prove B,C,K from Meredith single.
-P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | $ANS(C). % C
-P(i(a1,i(b,a1))). % K
% -P(i(i(a1,b),i(i(b,a2),i(a1,a2)))) | $ANS(BP). % B'
-P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(B). % B
end_of_list.
```

Before I discuss the not-particularly-satisfying results, with one exception, a few words about the just-given file are in order. Please consider the following three commands.

```
assign(max_weight,36).
assign(change_limit_after,800).
assign(new_max_weight,22).
```

The first of the three commands places a limit (of 36) on the complexity of retained information, ordinarily measured in terms of symbol count. Of course, the inclusion of one or more weight templates will override, where it applies, symbol count. You may have noticed that I chose the assigned value to be 36, rather than 48. I chose this value to make it easier and nicer for the program to go through the various levels. The next two commands inform OTTER to change the max_weight, in this case to 22, after so many clauses (800) have been "given". A "given clause" is a clause that is chosen to initiate application of inference rules. I was quite sure that the max_weight had better be decreased in order to go through enough levels to get to good information. The following command was included to further aid the program in coping with a possible combinatoric explosion.

```
assign(max_distinct_vars,12).
```

That command causes OTTER to discard any deduced clause if it relies on more than, in this case, twelve distinct variables.

As is obvious, I hoped for further progress in proof-length reduction. None occurred. But I did extract from the output a proof to be used in cramming in the next experiment. Here are the salient lines, clauses added to the initial set of support.

```
% Following 6 prove I from Ulrich 3, temp.ulrich.bci.out2b1.
P(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u))).
P(i(i(i(x,y),z),i(i(x,y),z))).
P(i(i(i(i(i(x,x),i(y,z)),z),u),i(y,u))).
P(i(x,i(i(i(y,y),i(x,z)),z))).
P(i(i(i(x,x),i(i(y,i(i(i(z,z),i(y,u)),u)),v)),v)).
P(i(x,x)).
```

Because cramming is not known to the majority of researchers, a few words might be appropriate.

The basic idea is that of cramming, forcing, if possible, the program to rely on some chosen set of formulas in completing one or more proofs. The formulas or equations, if all goes well, will be used in more than one proof, say, in the case of proving a conjunction. By using the added clauses in the initial set of support, the goal is to require fewer new, or additional, clauses to be used to complete whichever proofs are to be completed. Yes, the resulting so-called new subproofs will, in many cases, be in themselves longer than their correspondents in earlier experiments. However, by having the chosen formulas or equations, to be used in cramming, play double duty, triple duty, or more, the proof of a conjunction is often shorter than that already in hand.

The cramming strategy appeared to serve its purpose well. In particular, the run yielded six clauses that, when considered with the six used for the cramming, suggested that a 12-step proof of the join of *B*, *C*, and *I* was within reach. The actions I took are captured in the next input file, a file that indeed does yield a 12-step proof of the join.

An Input File Yielding a 12-Step Proof in BCI of the Join

```
set(hyper_res).
assign(max_weight,48).
% assign(change_limit_after,800).
% assign(new_max_weight,22).
assign(max_proofs,-1).
clear(print_kept).
set(ancestor_subsume).
set(back_sub).
% clear(for_sub).
clear(print_back_sub).
clear(print_kept).
clear(print_new_demod).
clear(print_back_demod).
clear(print_back_sub).
assign(max_distinct_vars,12).
assign(pick_given_ratio,4).
assign(max_mem,7500).
assign(report,54).
set(order_history).
set(input_sos_first).
% set(sos_queue).
assign(bsub_hint_wt,1).
set(keep_hint_subsumers).

weight_list(pick_and_purge).
% Following 6 prove I from Ulrich 3, temp.ulrich.bci.out2b1.
weight(P(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u))),-8).
weight(P(i(i(i(x,y),z),i(i(x,y),z))),-8).
weight(P(i(i(i(i(i(x,x),i(y,z)),z),u),i(y,u))),-8).
weight(P(i(x,i(i(i(y,y),i(x,z)),z))),-8).
weight(P(i(i(i(x,x),i(i(y,i(i(i(z,z),i(y,u)),u)),v)),v)),-8).
weight(P(i(x,x)),-8).
% Following 6 prove B, temp.ulrich.bci.out2d1, from Ulrich 3, apparently contain a proof of C.
weight(P(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v)))),-6).
weight(P(i(i(i(i(i(i(i(i(i(x,x),i(y,z)),z),i(i(u,u),i(v,w))),w),v6),i(v,v6)),v7),i(y,v7))),-6).
weight(P(i(i(i(i(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u)),i(i(v,v),i(w,v6))),v6),v7),i(w,v7))),-6).
weight(P(i(i(x,i(y,z)),i(y,i(x,z)))),-6).
weight(P(i(i(i(x,i(i(x,y),z)),u),i(i(y,z),u))),-6).
weight(P(i(i(x,y),i(i(z,x),i(z,y)))),-6).
% % Following 13 prove from Ulrich 3 the join of B C I, temp.ulrich.bci.out2b.
% weight(P(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u))),-2).
% weight(P(i(i(i(x,y),z),i(i(x,y),z))),-2).
% weight(P(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v)))),-2).
% weight(P(i(i(i(i(i(x,x),i(y,z)),z),u),i(y,u))),-2).
% weight(P(i(i(i(i(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v))),i(i(w,w),i(v6,v7))),v7),v8),i(v6,v8))),-2).
% weight(P(i(i(i(i(i(i(i(i(i(x,x),i(y,z)),z),i(i(u,u),i(v,w))),w),v6),i(v,v6)),v7),i(y,v7))),-2).
% weight(P(i(i(i(i(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u)),i(i(v,v),i(w,v6))),v6),v7),i(w,v7))),-2).
% weight(P(i(x,i(i(i(y,y),i(x,z)),z))),-2).
% weight(P(i(i(i(x,x),i(i(y,i(i(i(z,z),i(y,u)),u)),v)),v)),-2).
% weight(P(i(i(x,i(y,z)),i(y,i(x,z)))),-2).
% weight(P(i(i(i(x,i(i(x,y),z)),u),i(i(y,z),u))),-2).
% weight(P(i(i(x,y),i(i(z,x),i(z,y)))),-2).
% weight(P(i(x,x)),-2).
end_of_list.

list(usable).
-P(i(x,y)) | -P(x) | P(y).
-P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | -P(i(a1,a1)) | -P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(all).  % BCI
end_of_list.

list(sos).
P(i(i(i(i(i(x,x),i(y,z)),z),u),i(i(u,v),i(y,v)))).  % Ulrich BCI #3
% P(i(i(x,i(y,z)),i(i(i(u,u),i(v,y)),i(v,i(x,z))))).  % M's BCI #1
```

```
end_of_list.

list(passive).
% Following is neg of Meredith 1, single axiom for BCI.
-P(i(i(a1,i(a2,a3)),i(i(i(a4,a4),i(a5,a2)),i(a5,i(a1,a3))))) | $ANS(M1).
% Following is neg of I, of BCI.
-P(i(a,a)) | $ANS(I).
% Following prove B,C,K from Meredith single.
-P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | $ANS(C). % C
-P(i(a1,i(b,a1))). % K
% -P(i(i(a1,b),i(i(b,a2),i(a1,a2)))) | $ANS(BP). % B'
-P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(B). % B
end_of_list.
```

The proof of the join, yielded by relying on the just-given file, is the following.

A 12-Step Proof of the Join of B, C, and I, from Ulrich 3

```
-----> EMPTY CLAUSE at   0.01 sec ----> 212 [hyper,2,68,40,138] $ANS(all).

Length of proof is 12.  Level of proof is 6.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] -P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | -P(i(a1,a1)) | -P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(all).
3 [] P(i(i(i(i(i(x,x),i(y,z)),z),u),i(i(u,v),i(y,v)))).
9 [hyper,1,3,3] P(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u))).
11 [hyper,1,3,9] P(i(i(i(x,y),z),i(i(x,y),z))).
12 [hyper,1,9,3] P(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v)))).
14 [hyper,1,3,11] P(i(i(i(i(i(x,x),i(y,z)),z),u),i(y,u))).
18 [hyper,1,14,11] P(i(x,i(i(i(y,y),i(x,z)),z))).
20 [hyper,1,18,18] P(i(i(i(x,x),i(i(y,i(i(i(z,z),i(y,u)),u)),v)),v)).
40 [hyper,1,20,18] P(i(x,x)).
51 [hyper,1,3,12] P(i(i(i(i(i(i(i(i(i(x,x),i(y,z)),z),i(i(u,u),i(v,w))),w),v6),i(v,v6)),v7),i(y,v7))).
57 [hyper,1,12,9] P(i(i(i(i(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u)),i(i(v,v),i(w,v6))),v6),v7),i(w,v7))).
68 [hyper,1,51,14] P(i(i(x,i(y,z)),i(y,i(x,z)))).
99 [hyper,1,3,57] P(i(i(i(x,i(i(x,y),z)),u),i(i(y,z),u))).
138 [hyper,1,99,68] P(i(i(x,y),i(i(z,x),i(z,y)))).
212 [hyper,2,68,40,138] $ANS(all).
```

Again, in the spirit of providing more data that may in turn lead to the type of automation envisioned by Overbeek, a review is in order. Before the 12-step proof was found for the join, I had in hand a 5-step proof of *C*, a 6-step proof of *I*, and a different 6-step proof of *B*. An analysis of those proofs shows that the ensemble is contained in a 13-step proof of the join. You ask, perhaps eagerly, what happened to these short proofs? Well, they became a bit longer. In other words, shorter subproofs did not a total shorter proof make. Indeed, as the following shows, a trade was made to enable the 12-step proof to be completed. For example, 6-step proofs were traded for 9-step proofs. But—and here is the point of cramming—more commonality was present between longer subproofs than with the shorter subproofs of the earlier experiment. Put another way, various formulas were made to play double or triple duty; the intersections between pairs of subproofs was made greater, in number of formulas. So that you can see how this worked and also see how making progress in finding shorter proofs of members of a conjunction can lead to longer proofs of the entire conjunction, I include the following.

Data from the Run Yielding a 12-Step Proof in BCI of the Join

```
----> UNIT CONFLICT at   .0 sec ----> 41 [binary,40.1,5.1] $ANS(I).
Length of proof is 6.  Level of proof is 6.
----> UNIT CONFLICT at   .0 sec ----> 69 [binary,68.1,6.1] $ANS(C).
Length of proof is 6.  Level of proof is 4.
----> UNIT CONFLICT at   .1 sec ----> 139 [binary,138.1,8.1] $ANS(B).
Length of proof is 9.  Level of proof is 5.
-----> EMPTY CLAUSE at   .1 sec ----> 212 [hyper,2,68,40,138] $ANS(all).
Length of proof is 12.  Level of proof is 6.
----> UNIT CONFLICT at   2.95 sec ----> 1133 [binary,1132.1,6.1] $ANS(C).
Length of proof is 5.  Level of proof is 4.
-----> EMPTY CLAUSE at   3.5 sec ----> 1239 [hyper,2,10132,40,138] $ANS(all).
Length of proof is 14.  Level of proof is 6.
----> UNIT CONFLICT at   3.5 sec ----> 1248 [binary,10247.1,8.1] $ANS(B).
Length of proof is 8.  Level of proof is 5.
-----> EMPTY CLAUSE at   3.9 sec ----> 1268 [hyper,2,10132,40,10247] $ANS(all).
Length of proof is 13.  Level of proof is 6.
----> UNIT CONFLICT at  78. sec ----> 49065 [binary,49064.1,4.1] $ANS(M1).
Length of proof is 15.  Level of proof is 7.
----> UNIT CONFLICT at 313.88 sec ----> 9812 [binary,98119.1,4.1] $ANS(M1).
Length of proof is 14.  Level of proof is 8.
----> UNIT CONFLICT at 61.39 sec ----> 135695 [binary,135694.1,8.1] $ANS(B).
Length of proof is 6.  Level of proof is 6.
-----> EMPTY CLAUSE at 65.46 sec ----> 135696 [hyper,2,1132,40,135694] $ANS(all).
Length of proof is 14.  Level of proof is 6.
----> UNIT CONFLICT at 61.36 sec ----> 136115 [binary,136114.1,4.1] $ANS(M1).
Length of proof is 12.  Level of proof is 7.
```

You see that progress was occurring in the context of finding shorter proofs of individual members, but that progress was paid for in the context of the length of the proof of the conjunction of the three targets.

Further attempts at finding a proof of length strictly less than 12 of the join failed—at least, for now. As for a proof of Meredith 1, here is a 12-step proof that, for a little while, I could not improve upon.

A 12-Step Proof in BCI from Ulrich 3 of Meredith 1

```
----> UNIT CONFLICT at 610.36 sec ----> 136115 [binary,136114.1,4.1] $ANS(M1).

Length of proof is 12.  Level of proof is 7.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
3 [] P(i(i(i(i(i(x,x),i(y,z)),z),u),i(i(u,v),i(y,v)))).
4 [] -P(i(i(a1,i(a2,a3)),i(i(i(a4,a4),i(a5,a2)),i(a5,i(a1,a3))))) | $ANS(M1).
9 [hyper,1,3,3] P(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u))).
12 [hyper,1,9,3] P(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v)))).
48 [hyper,1,12,12] P(i(i(i(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v))),i(i(w,w),i(v6,v7))),v7),v8),i(v6,v8))).
51 [hyper,1,3,12] P(i(i(i(i(i(i(i(i(x,x),i(y,z)),z),i(i(u,u),i(v,w))),w),v6),i(v,v6)),v7),i(y,v7))).
57 [hyper,1,12,9] P(i(i(i(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u)),i(i(v,v),i(w,v6))),v6),v7),i(w,v7))).
108 [hyper,1,57,3] P(i(i(x,y),i(i(y,z),i(x,z)))).
190 [hyper,1,108,108] P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))).
198 [hyper,1,12,108] P(i(i(i(i(i(x,y),i(i(y,z),i(x,z))),i(i(u,u),i(v,w))),w),v6),i(v,v6))).
10132 [hyper,1,51,48] P(i(i(x,i(y,z)),i(y,i(x,z)))).
```

```
10245 [hyper,1,108,10132] P(i(i(i(x,i(y,z)),u),i(i(y,i(x,z)),u))).
135638 [hyper,1,198,190] P(i(i(x,y),i(i(i(z,z),i(u,x)),i(u,y)))).
136114 [hyper,1,10245,135638] P(i(i(x,i(y,z)),i(i(i(u,u),i(v,y)),i(v,i(x,z))))).
```

5. Mysteries Encountered, Mysteries Solved

In this section, you will read of mysteries that did indeed puzzle me for a while, causing me, eventually, to resume my study of the *BCI* logic. Here you will also learn of collaboration, of how the results of the preceding section led to Ulrich's quite delightful discovery. Finally, you will read of the mining of additional gold and how mysteries were solved.

Shortly after the two cited 12-step proofs were found and subsequent attempts to obtain further refinements failed, I e-mailed to my colleague Ted (Ulrich) both proofs. Although not in the strictest view about how science should be approached, I hoped for a somewhat big reaction to the shortness of the two proofs. I was certain that, if that was Ulrich's reaction, it would be genuine; he would not, even out of politeness, evince surprise at the brevity of the proofs.

He and I were indeed rewarded: I in that he was surprised, he in that the proofs provoked a study of them on his part. Consistent with a conclusion you may have already drawn from the foregoing, Ulrich is quite a logician, very familiar with both the *BCK* and *BCI* logics, as well as many other areas of logic. He has a program of his own, one that is largely interactive, that he uses to seek single axioms (of which Ulrich 3 and the one I cited for the *BCK* logic are examples). The studies reported here resulted directly from one of his marvelous e-mail messages, one that answered a question asked of me by M. Beeson concerning an example showing that first-order proofs can teach things not evident in an induction proof of the theorem under consideration. Ulrich and I had closely collaborated before; the time was rather early 2002; the topic was a study of the formula *XCB* in equivalential calculus; the result was a proof that that formula is in fact a single axiom, the last of its type.

Shortly after receiving the two 12-step proofs, Ulrich, to my substantial surprise, sent to me by e-mail an 11-step proof (given very soon) that derives from Ulrich 3, the 3-basis for the *BCI* logic that was featured in the preceding section. Almost equally impressive, his proof contains five formulas not present in the 12-step proof I had sent him.

Ulrich's 11-Step Proof Deriving the Join of B, C, and I, from Ulrich 3

```
-----> EMPTY CLAUSE at   0.03 sec ----> 138 [hyper,2,70,90,130] $ANS(all).

Length of proof is 11.  Level of proof is 6.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] -P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | -P(i(a1,a1)) | -P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(all).
3 [] P(i(i(i(i(i(x,x),i(y,z)),z),u),i(i(u,v),i(y,v)))).
10 [hyper,1,3,3] P(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u))).
13 [hyper,1,10,3] P(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v)))).
14 [hyper,1,13,13] P(i(i(i(i(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v))),i(i(w,w),i(v6,v7))),v7),v8),i(v6,v8))).
17 [hyper,1,13,10] P(i(i(i(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u)),i(i(v,v),i(w,v6))),v6),v7),i(w,v7))).
21 [hyper,1,3,14] P(i(i(i(i(i(i(x,x),i(i(y,y),i(z,u))),u),v),v),w),i(z,w))).
23 [hyper,1,14,10] P(i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(x,i(z,v)))).
33 [hyper,1,17,3] P(i(i(x,y),i(i(y,z),i(x,z)))).
53 [hyper,1,21,17] P(i(i(i(x,x),i(y,z)),i(y,z))).
70 [hyper,1,23,10] P(i(i(x,i(y,z)),i(y,i(x,z)))).
90 [hyper,1,53,53] P(i(x,x)).
130 [hyper,1,70,33] P(i(i(x,y),i(i(z,x),i(z,y)))).
```

138 [hyper,2,70,90,130] $ANS(all).

Now how in the world had he found this proof? After all, without success, I had tried quite hard, by using powerful methods, to find a proof shorter than length 12. I knew, from his comments, that he had relied on a breadth-first search, but so had I explored this approach. My conclusion: a phone call was in order.

Ulrich informed me that, after some consideration of my 12-step proof, he and his program studied the theorem, each (or both together) aiming at first finding proofs of *B*' and *C*, the following formulas (in clause notation).

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

(Incidentally, for the curious, he and I each prefer the formula *B*' to its closely related formula *B*.) Next, recall that Meredith's proof in the *BCK* logic stopped when *B*' was derived, omitting the last step, *B*. Then Ulrich, with his program, proceeded to derive *I* and *B*, resulting in the given 11-step proof.

Using as resonators weight templates that are, respectively, correspondents of Ulrich's eleven steps, I asked OTTER to reproduce his proof. The program immediately did so. And an intriguing mystery was encountered. In particular, why hadn't OTTER found the 11-step proof, even after a variety of attempts?

My reaction to this mystery was perhaps a type of panic. For what seemed the longest time—I expect to have ideas very quickly—I had no explanation, not even a theory. But then the following theory occurred to me, a theory that data already in hand might support. The theory is consistent with comments made throughout this essay: Perhaps OTTER found a shorter proof of some step on the way to completing a proof of the join such that the shorter proof blocked the completion of an 11-step proof. If good fortune was present, perhaps the so-called blocking subproof was for one of the three targets, *B*, *C*, or *I*.

I ran an experiment with Ulrich's eleven formulas, as weight templates (resonators), to have in hand the three subproofs and compared them with the three subproofs occurring in the run that completed the 12-step proof sent to my colleague. And, indeed, in the 11-step proof, you find a 7-step proof of *I*; but in the 12-step proof, you find a 6-step proof. I was content with not making any further examination or analysis, except for noting that the 7-step proof contains five formulas not in the 12-step proof. You thus have a fine example of how an individual can bring to a study useful and powerful knowledge. I can say with some certainty that OTTER would not have found Ulrich's 11-step proof for a long time—if ever. To borrow in a weak way from Shakespeare, the fault, dear program, lies not with thee; rather, the fault lies with the need for additional powerful strategies, ones that would enable an automated reasoning program to find proofs of the type found by Ulrich but *without* knowledge of an expert!

Have you conjectured about my next move? Please note that I imitate what has worked, even if I am the person to be imitated. Therefore, as you may have surmised, I decided to cram again, even though various crammings had not enabled me and OTTER to find a proof shorter than length 12. Just to be clear, I was not going to repeat an earlier experiment. Rather, I chose to cram on an 8-step proof of *B* within the Ulrich 11-step proof, a proof that contained three formulas not in the 12-step proof. Mathematics and logic are marvelous. Indeed, the experiment under discussion asserted that, with three additional formulas, a proof of the join of the three could be completed.

The following three formulas, OTTER informed me, would yield an 11-step proof.

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

With a program called "subtract", written by McCune (the author of OTTER) that applies a set-theoretic subtraction, I found that the second of the three formulas is not in the 12-step proof. Far more interesting to me was the fact that, with the subtract program, I learned that the first two of the three are not in the 11-step proof. Therefore, although I was certain that the inclusion of resonators corresponding to the eight formulas relied upon for cramming coupled with the three just given would yield a different 11-step proof, I conducted the corresponding experiment. After all, a different 11-step proof might be of interest and, if shown to Ulrich, might someday lead somewhere.

The experiment led to another mystery. In particular, OTTER did not find a different 11-step proof. No, the program found a—brace yourself—10-step proof. How could this situation have occurred? From where did this new treasure arise? Why did not OTTER merely produce the expected 11-step proof? Such mysteries, at least for me and some of my colleagues, are part of the delight of automated reasoning. I suppose such a reaction is a cousin to the programmer's feeling of excitement at the prospect of attempting to find a bug that resists finding.

The mystery can be solved—and was—by a study of the output file produced in the experiment. (If you are not fascinated by details, you might find the next few sentences a bit trying, but instructive.) First, I note that the following two clauses are found in the output and in the given order.

```
given clause #5: (wt=-4) 16 [hyper,1,3,13] P(i(i(i(i(i(i(i(i(x,x),i(y,z)),z),i(i(u,u),i(v,w))),w),v6),i(v,v6)),v7),i(y,v7))).
given clause #8: (wt=-4) 22 [hyper,1,14,10] P(i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(x,i(z,v)))).
```

Second, I note, or remind you, that a given formula can often be deduced from more than one set of parents. Third, I found that, in the 10-step proof as well as Ulrich's 11-step proof, there exists an 8-step subproof of *B*. However, the two proofs, viewed merely as collections of formulas, differ by one formula. In the 10-step proof, the first of the two just-given formulas is present, but not in the 11-step proof; the situation is reversed in that the second proof contains the second of the two formulas and not the first. In the 10-step proof, the first of the two given formulas couples (condense detaches) with, say, a formula *E* to yield, say, *F*; in the 11-step proof, the second of the two given formulas couples with a formula, say, *G* to yield *F*. Just a bit more and the mystery will be solved. Fourth, OTTER will not (in effect) retain the second 8-step proof, that in the 11-step proof, when (so-to-speak) completed because its length is not strictly less than the 8-step proof of *B* found within the 10-step proof. Fifth—and fortune was indeed present—with the newer 8-step proof, but one formula different from the old 8-step proof (found in the Ulrich 11-step proof), only two additional formulas sufficed to complete a proof of *I*, which completed a 10-step proof of the join in that a proof of *C* was present as a subproof of the newer 8-step proof of *B*. Just to remove any doubt, the 8-step proof of *B* in the 11-step proof also contains as a subproof a proof of *C*. Finally, an examination of the 11-step proof reveals that three steps were needed to complete a proof of *I*, and hence to prove the join, in contrast to needing but two additional formulas in the 10-step proof. Summarizing, because the first of the two formulas was deduced before the second, OTTER discovered a slightly different 8-step proof of *B* with the fortunate property that but two more formulas led to the gold, a new and shorter 10-step proof. To further clarify, the formula that begins with a set of nine nested *i*'s is used to complete anew 8-step proof of *B* and is common to a new 8-step proof of *I*.

Again, cramming proved to be a powerful strategy to rely upon, in the case under discussion indirectly leading to an even shorter proof than that in hand. Here is the 10-step proof.

A 10-Step Proof from Ulrich 3 of the Join of B, C, and I

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on elephant.mcs.anl.gov,
Sat Jul 15 09:21:07 2006
The command was "otter". The process ID is 7879.

----> EMPTY CLAUSE at   0.02 sec ----> 137 [hyper,2,28,88,108] $ANS(all).

Length of proof is 10. Level of proof is 5.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] -P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | -P(i(a1,a1)) | -P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(all).
3 [] P(i(i(i(i(i(x,x),i(y,z)),z),u),i(i(u,v),i(y,v)))).
10 [hyper,1,3,3] P(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u))).
```

```
13 [hyper,1,10,3] P(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v)))).
14 [hyper,1,13,13] P(i(i(i(i(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v))),i(i(w,w),i(v6,v7))),v7),v8),i(v6,v8))).
16 [hyper,1,3,13] P(i(i(i(i(i(i(i(i(i(x,x),i(y,z)),z),i(i(u,u),i(v,w))),w),v6),i(v,v6)),v7),i(y,v7))).
17 [hyper,1,13,10] P(i(i(i(i(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u)),i(i(v,v),i(w,v6))),v6),v7),i(w,v7))).
28 [hyper,1,16,14] P(i(i(x,i(y,z)),i(y,i(x,z)))).
45 [hyper,1,17,16] P(i(i(i(x,x),i(y,i(i(z,z),u))),i(y,u))).
49 [hyper,1,17,3] P(i(i(x,y),i(i(y,z),i(x,z)))).
88 [hyper,1,45,28] P(i(x,x)).
108 [hyper,1,28,49] P(i(i(x,y),i(i(z,x),i(z,y)))).
137 [hyper,2,28,88,108] $ANS(all).
```

Here are, in order, the two input files, the first used to produce Ulrich's 11-step proof, and the second used to produce the 10-step proof.

An Input File That Yields an 11-Step Proof in BCI from Ulrich 3

```
set(hyper_res).
assign(max_weight,48).
% assign(change_limit_after,800).
% assign(new_max_weight,22).
assign(max_proofs,-1).
clear(print_kept).
set(ancestor_subsume).
set(back_sub).
% clear(for_sub).
clear(print_back_sub).
clear(print_kept).
clear(print_new_demod).
clear(print_back_demod).
clear(print_back_sub).
assign(max_distinct_vars,12).
assign(pick_given_ratio,4).
assign(max_mem,750000).
assign(report,5400).
set(order_history).
set(input_sos_first).
%  set(sos_queue).
assign(bsub_hint_wt,1).
set(keep_hint_subsumers).

weight_list(pick_and_purge).
%  Following 12, including Ulrich 3, prove in 11 steps BCI, from Ulrich, rcvd 071306.
weight(P(i(i(i(i(i(x,x),i(y,z)),z),u),i(i(u,v),i(y,v)))),2).  % The axiom "U 3"
weight(P(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u))),2).
weight(P(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v)))),2).
weight(P(i(i(i(i(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u)),i(i(v,v),i(w,v6))),v6),v7),i(w,v7))),2).
weight(P(i(i(i(i(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v))),i(i(w,w),i(v6,v7))),v7),v8),i(v6,v8))),2).
weight(P(i(i(x,y),i(i(y,z),i(x,z)))),2).   % B'-yep, we get it FIRST!
weight(P(i(i(i(i(i(i(i(x,x),i(i(y,y),i(z,u))),u),v),v),w),i(z,w))),2).
weight(P(i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(x,i(z,v)))),2).
weight(P(i(i(i(x,x),i(y,z)),i(y,z))),2).
weight(P(i(i(x,i(y,z)),i(y,i(x,z)))),2).  % C
weight(P(i(x,x)),2).  % I
weight(P(i(i(x,y),i(i(z,x),i(z,y)))),2).  % B
%  %  Following 6 prove I from Ulrich 3, temp.Ulrich.bci.out2b1.
%  weight(P(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u))),-8).
%  weight(P(i(i(i(x,y),z),i(i(x,y),z))),-8).
%  weight(P(i(i(i(i(i(x,x),i(y,z)),z),u),i(y,u))),-8).
%  weight(P(i(x,i(i(i(y,y),i(x,z)),z))),-8).
%  weight(P(i(i(i(x,x),i(i(y,i(i(i(z,z),i(y,u)),u)),v)),v)),-8).
%  weight(P(i(x,x)),-8).
%  %  Following 6 prove B, temp.ulrich.bci.out2d1, from Ulrich 3, apparently contain a proof of C.
%  weight(P(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v)))),-6).
%  weight(P(i(i(i(i(i(i(i(i(i(x,x),i(y,z)),z),i(i(u,u),i(v,w))),w),v6),i(v,v6)),v7),i(y,v7))),-6).
%  weight(P(i(i(i(i(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u)),i(i(v,v),i(w,v6))),v6),v7),i(w,v7))),-6).
%  weight(P(i(i(x,i(y,z)),i(y,i(x,z)))),-6).
%  weight(P(i(i(i(x,i(i(x,y),z)),u),i(i(y,z),u))),-6).
%  weight(P(i(i(x,y),i(i(z,x),i(z,y)))),-6).
%  %  %  Following 13 prove from Ulrich 3 the join of B C I, temp.ulrich.bci.out2b.
%  %  weight(P(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u))),-2).
%  %  weight(P(i(i(i(x,y),z),i(i(x,y),z))),-2).
%  %  weight(P(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v)))),-2).
%  %  weight(P(i(i(i(i(i(x,x),i(y,z)),z),u),i(y,u))),-2).
%  %  weight(P(i(i(i(i(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v))),i(i(w,w),i(v6,v7))),v7),v8),i(v6,v8))),-2).
%  %  weight(P(i(i(i(i(i(i(i(i(i(x,x),i(y,z)),z),i(i(u,u),i(v,w))),w),v6),i(v,v6)),v7),i(y,v7))),-2).
%  %  weight(P(i(i(i(i(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u)),i(i(v,v),i(w,v6))),v6),v7),i(w,v7))),-2).
%  %  weight(P(i(x,i(i(i(y,y),i(x,z)),z))),-2).
%  %  weight(P(i(i(i(x,x),i(i(y,i(i(i(z,z),i(y,u)),u)),v)),v)),-2).
%  %  weight(P(i(i(x,i(y,z)),i(y,i(x,z)))),-2).
%  %  weight(P(i(i(i(x,i(i(x,y),z)),u),i(i(y,z),u))),-2).
%  %  weight(P(i(i(x,y),i(i(z,x),i(z,y)))),-2).
%  %  weight(P(i(x,x)),-2).
%  %  %  Following 12 prove Meredith1 from Ulrich 3, temp.ulrich.bci.out2b
%  %  weight(P(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u))),0).
%  %  weight(P(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v)))),0).
%  %  weight(P(i(i(i(i(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v))),i(i(w,w),i(v6,v7))),v7),v8),i(v6,v8))),0).
%  %  weight(P(i(i(i(i(i(i(i(i(i(x,x),i(y,z)),z),i(i(u,u),i(v,w))),w),v6),i(v,v6)),v7),i(y,v7))),0).
%  %  weight(P(i(i(i(i(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u)),i(i(v,v),i(w,v6))),v6),v7),i(w,v7))),0).
%  %  weight(P(i(i(x,i(y,z)),i(y,i(x,z)))),0).
%  %  weight(P(i(i(x,y),i(i(y,z),i(x,z)))),0).
%  %  weight(P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))),0).
%  %  weight(P(i(i(i(i(i(i(x,y),i(i(y,z),i(x,z))),i(i(u,u),i(v,w))),w),v6),i(v,v6))),0).
%  %  weight(P(i(i(i(x,i(y,z)),u),i(i(y,i(x,z)),u))),0).
%  %  weight(P(i(i(x,y),i(i(i(z,z),i(u,x)),i(u,y)))),0).
%  %  weight(P(i(i(x,i(y,z)),i(i(i(u,u),i(v,y)),i(v,i(x,z))))),0).
%  %  %  %  Following 31 include Meredith's single for BCK, and his 30-step proof.
%  %  %  weight(P(i(i(i(u,v),w),i(i(x,i(w,y)),i(v,i(x,y))))),2).  % Meredith's single axiom for BCK
%  %  %  weight(P(i(i(u,i(i(i(v,i(w,x)),i(y,i(v,x))),z)),i(w,i(u,z)))),2).
%  %  %  weight(P(i(u,i(i(i(v,w),x),i(w,i(i(y,i(u,z)),i(y,z)))))),2).
%  %  %  weight(P(i(u,i(v,i(i(u,w),i(i(x,i(v,y)),i(x,y)))))),2).
%  %  %  weight(P(i(u,i(i(i(i(i(v,w),x),i(i(y,i(x,z)),i(w,i(y,z)))),v6),i(i(v7,i(u,v8)),i(v7,v8))))),2).
%  %  %  weight(P(i(i(u,i(v,w)),i(x,i(i(y,i(x,z)),i(y,z))))),2).
%  %  %  weight(P(i(u,i(i(v,i(u,w)),i(v,w)))),2).
%  %  %  weight(P(i(i(u,i(i(v,i(i(w,i(v,x)),i(w,x))),y)),i(u,y))),2).
%  %  %  weight(P(i(i(u,i(v,w)),i(v,i(u,w)))),2).  %  C
%  %  %  weight(P(i(i(i(u,i(i(v,i(u,w)),i(v,w))),i(i(x,i(i(y,i(x,z)),i(y,z))),v6)),v6)),2).
% % %  weight(P(i(i(u,i(v,w)),i(i(i(x,i(i(y,i(x,z)),i(y,z))),v),i(u,w)))),2).
% % %  weight(P(i(i(u,i(i(i(v,i(i(w,i(v,x)),i(w,x))),y),i(z,v6)),v7)),i(i(y,v6),i(u,v7)))),2).
% % %  weight(P(i(i(u,v),i(w,i(i(i(x,i(i(y,i(x,z)),i(y,z))),u),v)))),2).
% % %  weight(P(i(i(u,v),i(i(i(w,i(i(x,i(w,y)),i(x,y))),u),v))),2).
% % %  weight(P(i(i(i(u,i(i(v,i(u,w)),i(v,w))),i(x,y)),i(i(z,i(i(v6,i(z,v7)),i(v6,v7))),x),y))),2).
% % %  weight(P(i(i(u,i(i(i(v,i(i(w,i(v,x)),i(w,x))),y),z),v6)),i(i(y,z),i(u,v6)))),2).
% % %  weight(P(i(i(u,i(i(v,i(w,x)),y)),i(i(v,x),i(u,y)))),2).
% % %  weight(P(i(i(u,v),i(w,i(u,v)))),2).
% % %  weight(P(i(i(u,i(i(v,i(w,x)),y)),i(x,i(u,y)))),2).
% % %  weight(P(i(u,i(v,i(w,u)))),2).  %  K
% % %  weight(P(i(u,i(v,u))),2). % K
% % %  weight(P(i(u,i(i(i(v,w),x),i(i(y,i(x,z)),i(w,i(y,z)))))),2).
% % %  weight(P(i(i(u,v),i(w,i(i(x,i(v,y)),i(u,i(x,y)))))),2).
% % %  weight(P(i(i(u,v),i(i(w,i(v,x)),i(u,i(w,x))))),2).
% % %  weight(P(i(i(u,i(v,w)),i(i(x,v),i(x,i(u,w))))),2).
% % %  weight(P(i(i(u,v),i(u,v))),2).
% % %  weight(P(i(i(u,v),i(u,i(i(v,w),w)))),2).
% % %  weight(P(i(i(u,v),i(i(i(u,i(i(v,w),w)),x),x))),2).
% % %  weight(P(i(i(i(u,i(i(v,w),w)),x),i(i(u,v),x))),2).
% % %  weight(P(i(i(u,v),i(i(v,w),i(u,w)))),2).  %  B'
% % %  weight(P(i(i(u,v),i(i(w,u),i(w,v)))),2).  %  B
end_of_list.

list(usable).
-P(i(x,y)) | -P(x) | P(y).
-P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | -P(i(a1,a1)) | -P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(all).  % BCI
end_of_list.

list(sos).
P(i(i(i(i(i(x,x),i(y,z)),z),u),i(i(u,v),i(y,v)))).  % Ulrich BCI #3
%  P(i(i(x,i(y,z)),i(i(i(u,u),i(v,y)),i(v,i(x,z))))).  % M's BCI #1
end_of_list.

list(passive).
%  Following is neg of Meredith 1, single axiom for BCI.
-P(i(i(a1,i(a2,a3)),i(i(i(a4,a4),i(a5,a2)),i(a5,i(a1,a3))))) | $ANS(M1).
%  Following is neg of I, of BCI.
-P(i(a,a)) | $ANS(I).
%  Following prove B,C,K from Meredith single.
-P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | $ANS(C).  %  C
-P(i(a1,i(b,a1))). % K
-P(i(i(a1,b),i(i(b,a2),i(a1,a2)))) | $ANS(BP).  %  B'
-P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(B).  % B
end_of_list.
```

An Input File That Yields a 10-Step Proof in BCI from Ulrich 3

```
set(hyper_res).
assign(max_weight,48).
% assign(change_limit_after,800).
% assign(new_max_weight,22).
assign(max_proofs,-1).
```

```
clear(print_kept).
set(ancestor_subsume).
set(back_sub).
% clear(for_sub).
clear(print_back_sub).
clear(print_kept).
clear(print_new_demod).
clear(print_back_demod).
clear(print_back_sub).
assign(max_distinct_vars,10).
assign(pick_given_ratio,4).
assign(max_mem,750000).
assign(report,5400).
set(order_history).
set(input_sos_first).
%  set(sos_queue).
assign(bsub_hint_wt,1).
set(keep_hint_subsumers).

weight_list(pick_and_purge).
%  Following 8 prove B, from Ulrich 11-step proof, temp.ulrich.bci.out2j.
weight(P(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u))),-4).
weight(P(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v)))),-4).
weight(P(i(i(i(i(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v))),i(i(w,w),i(v6,v7))),v7),v8),i(v6,v8))),-4).
weight(P(i(i(i(i(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u)),i(i(v,v),i(w,v6))),v6),v7),i(w,v7))),-4).
weight(P(i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(x,i(z,v)))),-4).
weight(P(i(i(x,y),i(i(y,z),i(x,z)))),-4).
weight(P(i(i(x,i(y,z)),i(y,i(x,z)))),-4).
weight(P(i(i(x,y),i(i(z,x),i(z,y)))),-4).
%  Following 3 yield I, with 8 proving B, getting BCI, queue, temp.ulrich.bci.out2j1.
weight(P(i(i(i(i(i(i(i(i(i(x,x),i(y,z)),z),i(i(u,u),i(v,w))),w),v6),i(v,v6)),v7),i(y,v7))),-4).
weight(P(i(i(i(x,x),i(y,i(i(z,z),u))),i(y,u))),-4).
weight(P(i(x,x)),-4).
% %  Following 12, including Ulrich 3, prove in 11 steps BCI, from Ulrich, rcvd 071306.
%  weight(P(i(i(i(i(i(x,x),i(y,z)),z),u),i(i(u,v),i(y,v)))),2).  % The axiom "U 3"
%  weight(P(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u))),2).
%  weight(P(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v)))),2).
%  weight(P(i(i(i(i(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u)),i(i(v,v),i(w,v6))),v6),v7),i(w,v7))),2).
%  weight(P(i(i(i(i(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v))),i(i(w,w),i(v6,v7))),v7),v8),i(v6,v8))),2).
%  weight(P(i(i(x,y),i(i(y,z),i(x,z)))),2).   % B'-yep, we get it FIRST!
%  weight(P(i(i(i(i(i(i(i(x,x),i(i(y,y),i(z,u))),u),v),v),w),i(z,w))),2).
%  weight(P(i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(x,i(z,v)))),2).
%  weight(P(i(i(i(x,x),i(y,z)),i(y,z))),2).
%  weight(P(i(i(x,i(y,z)),i(y,i(x,z)))),2).  % C
%  weight(P(i(x,x)),2).  % I
%  weight(P(i(i(x,y),i(i(z,x),i(z,y)))),2).  % B
% %  Following 6 prove I from Ulrich 3, temp.ulrich.bci.out2b1.
%  weight(P(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u))),-8).
%  weight(P(i(i(i(x,y),z),i(i(x,y),z))),-8).
%  weight(P(i(i(i(i(i(x,x),i(y,z)),z),u),i(y,u))),-8).
%  weight(P(i(x,i(i(i(y,y),i(x,z)),z))),-8).
%  weight(P(i(i(i(x,x),i(y,i(i(i(z,z),i(y,u)),u)),v)),v)),-8).
%  weight(P(i(x,x)),-8).
% % Following 6 prove B, temp.ulrich.bci.out2d1, from Ulrich 3, apparently contain a proof of C.
% weight(P(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v)))),-6).
% weight(P(i(i(i(i(i(i(i(i(i(x,x),i(y,z)),z),i(i(u,u),i(v,w))),w),v6),i(v,v6)),v7),i(y,v7))),-6).
% weight(P(i(i(i(i(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u)),i(i(v,v),i(w,v6))),v6),v7),i(w,v7))),-6).
% weight(P(i(i(x,i(y,z)),i(y,i(x,z)))),-6).
% weight(P(i(i(i(x,i(i(x,y),z)),u),i(i(y,z),u))),-6).
% weight(P(i(i(x,y),i(i(z,x),i(z,y)))),-6).
% % % Following 13 prove from Ulrich 3 the join of B C I, temp.ulrich.bci.out2b.
% % weight(P(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u))),-2).
% % weight(P(i(i(i(x,y),z),i(i(x,y),z))),-2).
% % weight(P(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v)))),-2).
% % weight(P(i(i(i(i(i(x,x),i(y,z)),z),u),i(y,u))),-2).
% % weight(P(i(i(i(i(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v))),i(i(w,w),i(v6,v7))),v7),v8),i(v6,v8))),-2).
% % weight(P(i(i(i(i(i(i(i(i(i(x,x),i(y,z)),z),i(i(u,u),i(v,w))),w),v6),i(v,v6)),v7),i(y,v7))),-2).
% % weight(P(i(i(i(i(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u)),i(i(v,v),i(w,v6))),v6),v7),i(w,v7))),-2).
% % weight(P(i(x,i(i(i(y,y),i(x,z)),z))),-2).
% % weight(P(i(i(i(x,x),i(i(y,i(i(i(z,z),i(y,u)),u)),v)),v)),-2).
% % weight(P(i(i(x,i(y,z)),i(y,i(x,z)))),-2).
% % weight(P(i(i(i(x,i(i(x,y),z)),u),i(i(y,z),u))),-2).
% % weight(P(i(i(x,y),i(i(z,x),i(z,y)))),-2).
% % weight(P(i(x,x)),-2).
% % % Following 12 prove Meredith1 from Ulrich 3, temp.ulrich.bci.out2b
% % weight(P(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u))),0).
% % weight(P(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v)))),0).
% % weight(P(i(i(i(i(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v))),i(i(w,w),i(v6,v7))),v7),v8),i(v6,v8))),0).
% % weight(P(i(i(i(i(i(i(i(i(i(x,x),i(y,z)),z),i(i(u,u),i(v,w))),w),v6),i(v,v6)),v7),i(y,v7))),0).
% % weight(P(i(i(i(i(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u)),i(i(v,v),i(w,v6))),v6),v7),i(w,v7))),0).
% % weight(P(i(i(x,i(y,z)),i(y,i(x,z)))),0).
% % weight(P(i(i(x,y),i(i(y,z),i(x,z)))),0).
% % weight(P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))),0).
% % weight(P(i(i(i(i(i(i(x,y),i(i(y,z),i(x,z))),i(i(u,u),i(v,w))),w),v6),i(v,v6))),0).
% % weight(P(i(i(i(x,i(y,z)),u),i(i(y,i(x,z)),u))),0).
% % weight(P(i(i(x,y),i(i(i(z,z),i(u,x)),i(u,y)))),0).
% % weight(P(i(i(x,i(y,z)),i(i(i(u,u),i(v,y)),i(v,i(x,z))))),0).
% % % % Following 31 include Meredith's single for BCK, and his 30-step proof.
% % % weight(P(i(i(i(u,v),w),i(i(x,i(w,y)),i(v,i(x,y))))),2). % Meredith's single axiom for BCK
% % % weight(P(i(i(u,i(i(i(v,i(w,x)),i(y,i(v,x))),z)),i(w,i(u,z)))),2).
% % % weight(P(i(u,i(i(i(v,w),x),i(w,i(i(y,i(u,z)),i(y,z)))))),2).
% % % weight(P(i(u,i(v,i(i(u,w),i(i(x,i(v,y)),i(x,y)))))),2).
% % % weight(P(i(u,i(i(i(i(i(v,w),x),i(i(y,i(x,z)),i(w,i(y,z)))),v6),i(i(v7,i(u,v8)),i(v7,v8))))),2).
% % % weight(P(i(i(u,i(v,w)),i(x,i(i(y,i(x,z)),i(y,z))))),2).
% % % weight(P(i(u,i(i(v,i(u,w)),i(v,w)))),2).
% % % weight(P(i(i(u,i(i(v,i(i(w,i(v,x)),i(w,x))),y)),i(u,y))),2).
% % % weight(P(i(i(u,i(v,w)),i(v,i(u,w)))),2). % C
% % % weight(P(i(i(i(u,i(i(v,i(u,w)),i(v,w))),i(i(x,i(i(y,i(x,z)),i(y,z))),v6)),v6)),2).
% % % weight(P(i(i(u,i(v,w)),i(i(i(x,i(i(y,i(x,z)),i(y,z))),v),i(u,w)))),2).
% % % weight(P(i(i(u,i(i(i(v,i(i(w,i(v,x)),i(w,x))),y),i(z,v6)),v7)),i(i(y,v6),i(u,v7)))),2).
% % % weight(P(i(i(u,v),i(w,i(i(i(x,i(i(y,i(x,z)),i(y,z))),u),v)))),2).
% % % weight(P(i(i(u,v),i(i(i(w,i(i(x,i(w,y)),i(x,y))),u),v))),2).
% % % weight(P(i(i(i(i(u,i(i(v,i(u,w)),i(v,w))),i(x,y)),i(i(z,i(i(v6,i(z,v7)),i(v6,v7))),x),y))),2).
% % % weight(P(i(i(u,i(i(i(i(v,i(i(w,i(v,x)),i(w,x))),y),z),v6)),i(i(y,z),i(u,v6)))),2).
% % % weight(P(i(i(u,i(i(v,i(w,x)),y)),i(i(v,x),i(u,y)))),2).
% % % weight(P(i(i(u,v),i(w,i(u,v)))),2).
% % % weight(P(i(i(u,i(i(v,i(w,x)),y)),i(x,i(u,y)))),2).
% % % weight(P(i(u,i(v,i(w,u)))),2). % K
% % % weight(P(i(u,i(v,u))),2). % K
% % % weight(P(i(u,i(i(i(v,w),x),i(i(y,i(x,z)),i(w,i(y,z)))))),2).
% % % weight(P(i(i(u,v),i(w,i(i(x,i(v,y)),i(u,i(x,y)))))),2).
% % % weight(P(i(i(u,v),i(i(w,i(v,x)),i(u,i(w,x))))),2).
% % % weight(P(i(i(u,i(v,w)),i(i(x,v),i(x,i(u,w))))),2).
% % % weight(P(i(i(u,v),i(u,v))),2).
% % % weight(P(i(i(u,v),i(u,i(i(v,w),w)))),2).
% % % weight(P(i(i(u,v),i(i(i(u,i(i(v,w),w)),x),x))),2).
% % % weight(P(i(i(i(u,i(i(v,w),w)),x),i(i(u,v),x))),2).
% % % weight(P(i(i(u,v),i(i(v,w),i(u,w)))),2). % B'
% % % weight(P(i(i(u,v),i(i(w,u),i(w,v)))),2). % B
end_of_list.

list(usable).
-P(i(x,y)) | -P(x) | P(y).
-P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | -P(i(a1,a1)) | -P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(all). % BCI
end_of_list.

list(sos).
P(i(i(i(i(i(x,x),i(y,z)),z),u),i(i(u,v),i(y,v)))). % Ulrich BCI #3
end_of_list.

list(passive).
% Following is neg of Meredith 1, single axiom for BCI.
-P(i(i(a1,i(a2,a3)),i(i(i(a4,a4),i(a5,a2)),i(a5,i(a1,a3))))) | $ANS(M1).
% Following is neg of I, of BCI.
-P(i(a,a)) | $ANS(I).
% Following prove B,C,K from Meredith single.
-P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | $ANS(C). % C
-P(i(a1,i(b,a1))). % K
-P(i(i(a1,b),i(i(b,a2),i(a1,a2)))) | $ANS(BP). % B'
-P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(B). % B
end_of_list.
```

If you experiment with the second input file, you will observe OTTER completing a succession of proofs of the join of the three targets of respective lengths 10, 11, 13, and 12. You just might find it instructive to study those four proofs and the intermediate proofs of the individual targets as they are completed. As for a proof from Ulrich 3 of Meredith 1 that is shorter than the 12-step proof I e-mailed to Ulrich, cramming again came through.

I chose for the cramming a proof of the next-to-the-last step of a 12-step proof. That particular subproof has length 7. And you see, from these comments, that you need not choose for cramming a proof of a member of a target conjunction.

An 11-Step Proof in BCI from Ulrich 3 of Meredith 1

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on jaguar.mcs.anl.gov,
Sat Jul 15 15:31:06 2006
The command was "otter". The process ID is 29987.

----> UNIT CONFLICT at   0.04 sec ----> 246 [binary,245.1,15.1] $ANS(M1).
```

```
Length of proof is 11.  Level of proof is 9.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
3 [] P(i(i(i(i(i(x,x),i(y,z)),z),u),i(i(u,v),i(y,v)))).
15 [] -P(i(i(a1,i(a2,a3)),i(i(i(a4,a4),i(a5,a2)),i(a5,i(a1,a3))))) | $ANS(M1).
21 [hyper,1,3,3] P(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u))).
25 [hyper,1,21,3] P(i(x,i(i(i(i(x,i(i(y,y),i(z,u))),u),v),i(z,v)))).
30 [hyper,1,3,25] P(i(i(i(i(i(i(i(i(i(x,x),i(y,z)),z),i(i(u,u),i(v,w))),w),v6),i(v,v6)),v7),i(y,v7))).
32 [hyper,1,25,21] P(i(i(i(i(i(i(i(i(x,y),i(i(z,x),y)),u),i(z,u)),i(i(v,v),i(w,v6))),v6),v7),i(w,v7))).
46 [hyper,1,32,3] P(i(i(x,y),i(i(y,z),i(x,z)))).
58 [hyper,1,46,46] P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))).
62 [hyper,1,25,46] P(i(i(i(i(i(i(x,y),i(i(y,z),i(x,z))),i(i(u,u),i(v,w))),w),v6),i(v,v6))).
95 [hyper,1,62,58] P(i(i(x,y),i(i(i(z,z),i(u,x)),i(u,y)))).
105 [hyper,1,95,95] P(i(i(i(x,x),i(y,i(z,u))),i(y,i(i(i(v,v),i(w,z)),i(w,u))))).
178 [hyper,1,105,3] P(i(i(i(i(i(x,x),i(y,z)),z),u),i(i(i(v,v),i(w,y)),i(w,u)))).
245 [hyper,1,30,178] P(i(i(x,i(y,z)),i(i(i(u,u),i(v,y)),i(v,i(x,z))))).
```

A Global Mystery

If you would enjoy attempting to solve a global mystery, you might seek to formulate a general strategy whose use would enable OTTER to find, from Ulrich 3, the 10-step proof given for the join of the three targets, without, of course, informing the program of its existence in any crucial manner.

6. #6

In one sense, if all goes according to some grand plan, your reading of an essay or of a researcher's notebook, found on this website, will offer you implicit and perhaps many explicit challenges. In this section, I offer some explicit challenges, leaving to you the formulation of implicit challenges. I shall, therefore, touch on other results related to those discussed in earlier sections.

Two ideas come to mind, each of which might admit automation to some degree, in the context of seeking shorter proofs. The first idea is spawned, I suspect, from my interest in cramming, and I think it could be useful for proof finding as well as proof refining. In cramming, the notion is to find proofs, say, of members of a conjunction, whose intersections pairwise are maximal. In other words, you prefer to find proofs that require few if any extra steps to complete the remaining proofs to complete a proof of the entire (join) set of members. You begin by accruing a number of proofs of, say, the first member of the conjunction, then of the second, then of the third; for example, you might focus on the *BCK* logic. You select one proof from the first set, one from the second, and one from the third and place the three selected items in a file and do a sort -u; that is, you produce from the three selected proofs their union (with no duplicates). You do this for all combinations, each combination involving one member from each of the three sets of proofs. You are looking for the smallest union, that with the fewest members. That union will, if all goes well, amount to the shortest proof (of the conjunction) available from the study just described. The just-given approach is aimed at maximal intersections of the proofs considered pairwise. Automation appears to be possible.

The second idea comes from my esteemed colleague Ulrich. First, you have the automated reasoning program in use find a number of proofs for each member of the conjunction under study. Then you cram on each of the various proofs. The idea can be extended to finding proofs of various steps of a proof in hand, rather than focusing on the members of the conjunction. Ulrich found the cited 11-step proof for the *BCI* logic by cramming on a 7-step proof of *I*. I had found a 12-step proof by cramming on a 6-step subproof, which shows that cramming on a shorter subproof does not necessarily lead to progress in the context of a shorter proof of the entire conjunction. You see, if you cram on a longer subproof, you give the program more possible combinations of ancestors to try to minimize the number of additional steps that are needed to complete the sought-after entire proof. Again, automation seems possible. I can offer no guidelines that suggest which proof will turn out to be the one that, with cramming, produces the shortest proof of the conjunction.

As indicated in the earlier material, to cram, you (usually) place the steps of the subproof that provides the basis for cramming in the initial set of support, together with the axioms in use. You then have the program conduct a breadth-first search, equivalently, level-saturation search. The two sentences correctly imply that a level-saturation search can be replaced with one based on complexity of formulas or equations. However, a breadth-first approach, so it appears, is more likely to force (cram) the steps of the subproof into other subproofs, thus increasing the likelihood of finding a shorter total proof than that in hand.

For a sharply different type of challenge, as is the case with so many puzzles, you might attempt to find a better solution than one given, might attempt to find proofs that supplant those given in this notebook. Are you asking yourself how this fits into the global scheme of things? Who in the history of mathematics, for example, would care about simpler proofs? Well, apparently Hilbert cared. His twenty-fourth problem, discovered only in the last decade, is concerned with finding "simpler proofs". One can only imagine, one of my colleagues recently observed, how extremely important these studies of proof refinement would have been if Hilbert had only had the time to include his twenty-fourth problem in his 1900 Paris lecture. That mathematician commented in his notebooks that he simply did not have the time to make the problem sufficiently precise to include it in the Paris lecture. (Thanks goes to R. Thiele for the unearthing of this Hilbert problem.)

Different from the challenge of automating some approach is the analysis that explains why something like the following happened. I was studying the *BCI* logic still, but now in the context of a different single axiom, one supplied by Ulrich, his eighteenth.

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

Among my objectives was the continued testing of the hypothesis that the use of resonators from one successful study would enable OTTER to quickly prove a theorem other than that from which the resonators were extracted. Indeed, I decided to take, from the proof of the join of *B*, *C*, and *I* based on Ulrich's third axiom, the steps of the best proof I had found and use them as resonators in the study of his eighteenth. I was intent upon finding short proofs, from his eighteenth, of the join of the usual three targets and of Meredith 1. I now again supply the negation of *M*1 because of the relevance to the challenge I am about to offer.

```
4 [] -P(i(i(a1,i(a2,a3)),i(i(i(a4,a4),i(a5,a2)),i(a5,i(a1,a3))))) | $ANS(M1).
```

To conduct my study, I simply took an input file used to study Ulrich's third single axiom for the *BCI* logic, replaced in list(sos) the third by the eighteenth, and included as resonators, rather than the ten corresponding to the 10-step proof OTTER had discovered, the eight plus three that had been used to find that proof. After all, that set of eleven had, surprisingly, led to a most satisfying result.

As the following data shows, OTTER completed a proof of the join quite quickly.

A 9-Step Proof from Ulrich 18 of the Join of B, C, and I

```
-----> EMPTY CLAUSE at   7.61 sec ----> 20368 [hyper,2,5687,20355,12] $ANS(all).

Length of proof is 9.  Level of proof is 6.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] -P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | -P(i(a1,a1)) | -P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(all).
3 [] P(i(i(i(x,x),i(y,z)),i(i(i(i(u,v),v),y),i(u,z)))).
```

```
10 [hyper,1,3,3] P(i(i(i(i(x,y),y),i(i(i(z,u),u),v)),i(x,i(z,v)))).
11 [hyper,1,3,10] P(i(i(i(i(x,y),y),i(z,u)),i(x,i(z,u)))).
12 [hyper,1,10,3] P(i(i(x,y),i(i(z,x),i(z,y)))).
30 [hyper,1,11,12] P(i(x,i(i(y,i(x,z)),i(y,z)))).
45 [hyper,1,30,30] P(i(i(x,i(i(y,i(i(z,i(y,u)),i(z,u))),v)),i(x,v))).
50 [hyper,1,3,30] P(i(i(i(i(x,y),y),i(z,i(i(u,u),v))),i(x,i(z,v)))).
5687 [hyper,1,45,30] P(i(i(x,i(y,z)),i(y,i(x,z)))).
6540 [hyper,1,50,50] P(i(i(x,i(i(y,y),i(i(z,z),u))),i(x,u))).
20355 [hyper,1,6540,30] P(i(x,x)).
20368 [hyper,2,5687,20355,12] $ANS(all).
```

For the curious, the given proof is actually the third proof of the join, the first two being, respectively, of lengths 16 and 12, both found in less than 2 CPU seconds. The challenge, rather than focusing on a proof of the join, focuses on two proofs of Meredith 1, both obtained in the same experiment, but much later. Here are the two for your close examination.

A 13-Step Proof in BCI of Meredith 1 from Ulrich 18

```
----- Otter 3.3g-work, Jan 2005 -----
----> UNIT CONFLICT at 142.12 sec ----> 72563 [binary,72562.1,4.1] $ANS(M1).

Length of proof is 13.  Level of proof is 8.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
3 [] P(i(i(i(x,x),i(y,z)),i(i(i(u,v),v),y),i(u,z)))).
4 [] -P(i(i(a1,i(a2,a3)),i(i(i(a4,a4),i(a5,a2)),i(a5,i(a1,a3))))) | $ANS(M1).
10 [hyper,1,3,3] P(i(i(i(i(x,y),y),i(i(i(z,u),u),v)),i(x,i(z,v)))).
11 [hyper,1,3,10] P(i(i(i(i(x,y),y),i(z,u)),i(x,i(z,u)))).
12 [hyper,1,10,3] P(i(i(x,y),i(i(z,x),i(z,y)))).
14 [hyper,1,12,12] P(i(i(x,i(y,z)),i(x,i(i(u,y),i(u,z))))).
30 [hyper,1,11,12] P(i(x,i(i(y,i(x,z)),i(y,z)))).
45 [hyper,1,30,30] P(i(i(x,i(i(y,i(i(z,i(y,u)),i(z,u))),v)),i(x,v))).
50 [hyper,1,3,30] P(i(i(i(i(x,y),y),i(z,i(i(u,u),v))),i(x,i(z,v)))).
5687 [hyper,1,45,30] P(i(i(x,i(y,z)),i(y,i(x,z)))).
5766 [hyper,1,14,5687] P(i(i(x,i(y,z)),i(i(u,y),i(u,i(x,z))))).
6609 [hyper,1,50,10] P(i(i(x,i(i(i(i(y,y),z),z),u)),i(x,u))).
24042 [hyper,1,6609,12] P(i(i(x,y),i(i(i(z,z),x),y))).
24119 [hyper,1,12,24042] P(i(i(x,i(y,z)),i(x,i(i(i(u,u),y),z)))).
72562 [hyper,1,24119,5766] P(i(i(x,i(y,z)),i(i(i(u,u),i(v,y)),i(v,i(x,z))))).
```

A 12-Step Proof in BCI of Meredith 1 from Ulrich 18

```
----- Otter 3.3g-work, Jan 2005 -----

----> UNIT CONFLICT at 30375.63 sec ----> 935390 [binary,935389.1,4.1] $ANS(M1).

Length of proof is 12.  Level of proof is 8.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
```

```
3 [] P(i(i(i(x,x),i(y,z)),i(i(i(i(u,v),v),y),i(u,z)))).
4 [] -P(i(i(a1,i(a2,a3)),i(i(i(a4,a4),i(a5,a2)),i(a5,i(a1,a3))))) | $ANS(M1).
10 [hyper,1,3,3] P(i(i(i(i(x,y),y),i(i(i(z,u),u),v)),i(x,i(z,v)))).
11 [hyper,1,3,10] P(i(i(i(i(x,y),y),i(z,u)),i(x,i(z,u)))).
12 [hyper,1,10,3] P(i(i(x,y),i(i(z,x),i(z,y)))).
14 [hyper,1,12,12] P(i(i(x,i(y,z)),i(x,i(i(u,y),i(u,z))))).
30 [hyper,1,11,12] P(i(x,i(i(y,i(x,z)),i(y,z)))).
45 [hyper,1,30,30] P(i(i(x,i(i(y,i(i(z,i(y,u)),i(z,u))),v)),i(x,v))).
50 [hyper,1,3,30] P(i(i(i(i(x,y),y),i(z,i(i(u,u),v))),i(x,i(z,v)))).
5687 [hyper,1,45,30] P(i(i(x,i(y,z)),i(y,i(x,z)))).
5766 [hyper,1,14,5687] P(i(i(x,i(y,z)),i(i(u,y),i(u,i(x,z))))).
6609 [hyper,1,50,10] P(i(i(x,i(i(i(i(y,y),z),z),u)),i(x,u))).
15110 [hyper,1,14,5766] P(i(i(x,i(y,z)),i(i(u,i(v,y)),i(u,i(v,i(x,z)))))).
935389 [hyper,1,6609,15110] P(i(i(x,i(y,z)),i(i(i(u,u),i(v,y)),i(v,i(x,z))))).
```

In addition to the huge gap in CPU time required to complete the two given proofs, an examination of the two proofs reveals a rather startling bit of data. (The robustness of OTTER, so I have been told, is most impressive, computing for days in real time, sometimes yielding one diamond after another; see, for example, the two proofs.) I will assume you have perused the two proofs and found some key differences and similarities; therefore, I can make some comments before issuing the precise challenge. The 12-step proof does not include the eleventh step of the 13-step proof. Thus you see in part why a shorter proof was completed. The next-to-the-last step in the 12-step proof differs from the next-to-the-last step of the 13-step proof. Finally, perhaps the key difference, in the 12-step proof, Meredith 1 (the last step) is deduced by applying condensed detachment to the tenth and eleventh step. In contrast, the last step, Meredith 1, of the 13-step proof is deduced from the ninth and twelfth steps. In other words, the sought-after goal, Meredith 1, has a different set of parents in the 12-step proof from the set in the 13-step proof.

Now, to set the stage for the challenge, I note that the number of the minor premiss (the third number in the history) of the deduction of the last step of the 12-step proof is strictly less than the number of the major premiss in the history of the last step in the 13-step proof. In the attempt to clarify this last comment, OTTER already had in hand the parents that yield, when condensed detachment is applied, the sought-after conclusion *before* completing the 13-step proof. And yet, the 12-step proof was not offered before the 13-step proof. OTTER simply did not consider, for application of condensed detachment, the pair consisting of the parents of the last step of the 12-step proof until the program completed the 13-step proof. A mystery to solve: How can that be?

Especially for those who do not know, the numbers given (by OTTER) in the history of a deduction in this notebook are, in order, nucleus (for hyperresolution), major premiss, and minor premiss. The numbers (of clauses) indicate where the corresponding clause resides, its position among the retained clauses. Therefore, although the program could have completed the 12-step proof before the 13-step proof, in that the needed formulas were already in hand, it did not. Your challenge: Try to explain what occurred.

If you wish to solve this mystery before reading the solution, you had best pause now. Indeed, almost immediately, I shall give an explanation, a solution to the puzzle. In particular, I shall explain why OTTER apparently ignored for consideration with condensed detachment the pair (6609,15110) until the given 13-step proof was completed.

I shall take it slowly to permit you to stop at any point in an attempt to complete the solution. I note immediately that OTTER prefers for initiating inference rule applications, ordinarily, the next available formula or equation based on smallest complexity. (The field owes great thanks to Overbeek for the concept of *weighting*, measuring complexity in terms of symbol count unless dictated otherwise. This direction strategy, when offered by an automated reasoning program, gives the program much power and, if the user has good insight or knowledge, allows that user to offer valuable guidelines.) First of all, the parents of the 13-step proof each have weight 16, rely on sixteen symbols (including the predicate symbol). The weight or complexity of a clause is determined by its symbol count, unless one of the included resonators or hints dictates otherwise, which, in the two cases under discussion, did not occur. By way of expansion, in the experiment in focus, none of the eleven included resonators matched (ignoring the specific variables) any of the four parents that are relevant to the deduction of Meredith 1 in, respectively, the 12-step and 13-step proofs. To continue—pause if you wish to think it through from here—three of the four parents have weight (complexity) 16, and the fourth (that numbered 15110) has weight 20. Before OTTER focuses for inference rule application on the key parent, (15110), with weight 20, the program focuses on 414 clauses with weight 16. Further, clause (15110) is numbered 1405 among the clauses chosen for conclusion drawing. In contrast, clause (24119) is numbered 379, meaning its position among the items chosen for inference rule initiation is much earlier than for clause (15110), which explains why so much CPU time elapsed before the 12-step proof was completed. Thus, the solution to the mystery of why the 12-step proof was not offered and found before the 13-step proof rests with the complexity, 20, of one of the parents of the last step in the 12-step proof. Such intricacies, rather than causing disappointment, are—if you accept this—not much different from the unaided researcher bypassing what turns out later to be a key lemma or key formula or equation. In particular, research is dotted with occurrences of some person noting that the key steps were in hand, if only they had been recognized when first encountered.

In answer to a question that you might ask at this moment, just two of the eleven resonators are present as formulas in the 12-step proof and two in the 13-step proof. Further, if viewed at the resonator level, ignoring all variables and treating them simply as a variable (as indistinguishable), just three of the eleven are present in the 12-step proof and three in the 13-step proof. For another sample of what is occurring, but three of the eleven formulas, used as resonators, are present in the first proof of the join of the three targets (in a proof of length 16). At the resonator level, four of the eleven are present in the 16-step proof. My conclusion: I believe that the resonators still played an important role throughout the experiment under discussion, allowing the program to bridge various gaps, so-to-speak.

Before leaving this area, it occurred to me that almost required was an experiment focusing on the use of *no* resonators, to see how crucial their use actually is. Therefore, without any guidance in the form of resonators, hints, or other weight templates, I asked OTTER to find, from Ulrich 18, proofs of the join of *B*, *C*, and *I* and of Meredith 1. I expected—perhaps, more accurately, preferred—that far more CPU time would be required when compared with the use of the already-cited eleven resonators. OTTER found the proofs assigned to it, and, in the beginning, in less CPU time. However, the completion of the final 12-step proof, that of Meredith 1, took about 104 more CPU-seconds when no resonators were used, compared with the use of the eleven resonators. The two 12-step proofs, that with and that without the use of resonators, are identical. To be a bit more precise, the last step in the 12-step proof based on the use of the eleven resonators is numbered 935389, whereas the last step when no resonators are used is 942053. Conclusion: The use of resonators did not play a crucial role, although their inclusion served some small purpose.

Ulrich suggested one additional set of experiments, those focusing on the use of his ninth single axiom for the *BCI* logic, the following.

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

Because I shall detail an interesting set of experiments for deriving the join of *B*, *C*, and *I* from Ulrich 9, I shall first simply give the best proof OTTER found (in the first attempt) that derives Meredith 1 from Ulrich 9.

A 9-Step Proof in BCI of Meredith 1 from Ulrich 9

```
----> UNIT CONFLICT at 2283.06 sec ----> 252699 [binary,252698.1,4.1] $ANS(M1).

Length of proof is 9.  Level of proof is 6.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
3 [] P(i(i(i(x,x),i(y,z)),i(i(i(i(z,u),u),v),i(y,v)))).
4 [] -P(i(i(a1,i(a2,a3)),i(i(i(a4,a4),i(a5,a2)),i(a5,i(a1,a3))))) | $ANS(M1).
```

```
10 [hyper,1,3,3] P(i(i(i(i(i(x,y),z),z),u),i(i(i(i(x,v),v),y),u))).
12 [hyper,1,10,3] P(i(i(i(i(x,y),y),z),i(i(i(i(z,u),u),v),i(x,v)))).
13 [hyper,1,12,12] P(i(i(i(i(i(i(i(i(x,y),y),z),i(u,z)),v),v),w),i(i(u,x),w))).
15 [hyper,1,12,10] P(i(i(i(i(i(i(i(i(x,y),y),z),u),v),v),w),i(i(i(x,z),u),w))).
30 [hyper,1,13,13] P(i(i(x,i(y,z)),i(i(u,y),i(x,i(u,z))))).
44 [hyper,1,3,30] P(i(i(i(i(i(i(x,y),i(z,y)),u),u),v),i(i(z,x),v))).
211 [hyper,1,13,15] P(i(i(x,i(y,z)),i(i(i(y,z),u),i(x,u)))).
266 [hyper,1,211,3] P(i(i(i(i(i(i(x,y),y),z),i(u,z)),v),i(i(i(w,w),i(u,x)),v))).
252698 [hyper,1,44,266] P(i(i(x,i(y,z)),i(i(i(u,u),i(v,y)),i(v,i(x,z))))).
```

Additional attempts at finding a shorter proof produced nothing of interest.

The preceding result was obtained by simply taking an input file used to study Ulrich 18 and replacing that axiom with Ulrich 9 (in the initial set of support list). In the output file that contained the given 9-step proof of Meredith 1, I found the following proof, a proof that illustrates in various ways the power OTTER offers.

A 16-Step Proof of the Join of B, C, and I from Ulrich 9

```
-----> EMPTY CLAUSE at 65197.30 sec ----> 1324675 [hyper,2,1321737,9972,1322447] $ANS(all).

Length of proof is 16.  Level of proof is 10.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] -P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | -P(i(a1,a1)) | -P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(all).
3 [] P(i(i(i(x,x),i(y,z)),i(i(i(z,u),u),v),i(y,v)))).
10 [hyper,1,3,3] P(i(i(i(i(i(x,y),z),z),u),i(i(i(i(x,v),v),y),u))).
12 [hyper,1,10,3] P(i(i(i(i(x,y),y),z),i(i(i(i(z,u),u),v),i(x,v)))).
13 [hyper,1,12,12] P(i(i(i(i(i(i(i(i(x,y),y),z),i(u,z)),v),v),w),i(i(u,x),w))).
14 [hyper,1,3,12] P(i(i(i(i(i(x,y),z),z),u),i(i(i(i(i(i(x,v),v),w),w),y),u))).
15 [hyper,1,12,10] P(i(i(i(i(i(i(i(i(x,y),y),z),u),v),v),w),i(i(i(x,z),u),w))).
107 [hyper,1,14,12] P(i(i(i(i(i(i(x,y),y),z),z),u),i(i(i(i(u,v),v),w),i(x,w)))).
209 [hyper,1,15,15] P(i(i(i(i(x,y),z),u),i(i(i(x,y),z),u))).
238 [hyper,1,10,209] P(i(i(i(i(x,y),y),z),i(i(i(x,z),u),u))).
393 [hyper,1,238,238] P(i(i(i(i(x,y),i(i(i(x,y),z),z)),u),u)).
747 [hyper,1,393,3] P(i(i(i(i(x,y),y),z),i(i(i(u,u),x),z))).
2307 [hyper,1,747,747] P(i(i(i(x,x),i(y,z)),i(i(i(u,u),y),z))).
9685 [hyper,1,2307,2307] P(i(i(i(x,x),i(i(y,y),z)),z)).
9972 [hyper,1,9685,9685] P(i(x,x)).
47624 [hyper,1,107,10] P(i(i(i(i(i(i(i(i(x,y),y),z),z),u),u),v),i(x,v))).
1321737 [hyper,1,13,47624] P(i(i(x,i(y,z)),i(y,i(x,z)))).
1322447 [hyper,1,47624,13] P(i(i(x,y),i(i(z,x),i(z,y)))).
1324675 [hyper,2,1321737,9972,1322447] $ANS(all).
```

Do you share with me the pleasure of seeing large numbers? In particular, the CPU time is impressive, more than 65,000 CPU-seconds. Perhaps more impressive are the numbers of some of the clauses in the proof, more than 1,000,000, meaning that more than one million new formulas were retained before the 16-step proof was completed. Of course, as you most likely have deduced, the 16-step proof was not the first proof found of the join of the three targets; rather, it was the sixth. The first four were completed in less than 3 CPU-seconds, and the fifth was completed in approximately 3454 CPU-seconds. The lengths in order of the first five proofs are 20, 19, 21, 19, and 21. Were I lecturing at this time, I would say something like the following.

OTTER decided to see how strong the desire might be for a better proof than the first five, better in terms of length. To test your zeal, the program searched for a long time, retaining more than one million new formulas. Because you evinced eagerness to see something wonderful, the program finally, after more than 65,000 CPU-seconds (more than 18 CPU-hours), offered you the given 16-step proof. I do not recall precisely, but a few days in real time elapsed before the gold was mined.

Perhaps stimulated by the effort put forth by OTTER, perhaps by the large numbers (in time and in clause numbers), or perhaps because I so often seek better (in terms of length, at least) proofs, I decided to seek a proof shorter than length 16. Because the output file contained a 7-step proof of *B* and a 7-step proof of *C*, I believe within the 16-step proof, I looked to the proof of *I* within the 16-step proof and found one of length 10 and level 10. (For the individual who enjoys fine points, a proof whose length and level are the same means that (so-to-speak) a straight line was drawn from the single axiom, in this case, to the conclusion; each derived step has level one greater than its predecessor.) Now you know what I had in mind almost immediately: cramming on the 10-step proof of *I*, with the hope that not many additional steps would be needed to complete a shorter proof of the join of the three targets. Therefore, I placed in list(sos) ten formulas, a proof of *I*, commented out the assign(pick_given_ratio) parameter, and commented back in set(sos_queue), meaning that I instructed OTTER to conduct a breadth-first (level-saturation) search.

The program very quickly found a 3-step proof of *B* and a 3-step proof of *C*. In that 3+3+10 is 16, on the surface, no progress occurred; after all, the goal was to find a proof of length strictly less than 16. However, with sort -u—equivalently, you could simply read both cited 3-step proofs—delight was the result. In particular, the two 3-step proofs share the first two steps, which means that but four additional steps, together with the ten, would enable OTTER to return a 14-step proof (by using the fourteen as resonators). And it did produce the 14-step proof, the following, which provides you with yet another triumph for cramming.

A 14-Step Proof of the Join of B, C, and I from Ulrich 9

```
-----> EMPTY CLAUSE at   0.02 sec ----> 181 [hyper,2,129,100,138] $ANS(all).

Length of proof is 14.  Level of proof is 10.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] -P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | -P(i(a1,a1)) | -P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(all).
3 [] P(i(i(i(x,x),i(y,z)),i(i(i(z,u),u),v),i(y,v)))).
10 [hyper,1,3,3] P(i(i(i(i(i(x,y),z),z),u),i(i(i(i(x,v),v),y),u))).
12 [hyper,1,10,3] P(i(i(i(i(x,y),y),z),i(i(i(i(z,u),u),v),i(x,v)))).
13 [hyper,1,12,12] P(i(i(i(i(i(i(i(i(x,y),y),z),i(u,z)),v),v),w),i(i(u,x),w))).
15 [hyper,1,12,10] P(i(i(i(i(i(i(i(i(x,y),y),z),u),v),v),w),i(i(i(x,z),u),w))).
16 [hyper,1,15,15] P(i(i(i(i(x,y),z),u),i(i(i(x,y),z),u))).
23 [hyper,1,10,16] P(i(i(i(i(x,y),y),z),i(i(i(x,z),u),u))).
33 [hyper,1,23,23] P(i(i(i(i(x,y),i(i(i(x,y),z),z)),u),u)).
42 [hyper,1,33,12] P(i(i(i(i(i(i(i(i(x,y),y),z),z),u),u),v),i(x,v))).
44 [hyper,1,33,3] P(i(i(i(i(x,y),y),z),i(i(i(u,u),x),z))).
45 [hyper,1,44,44] P(i(i(i(x,x),i(y,z)),i(i(i(u,u),y),z))).
56 [hyper,1,45,45] P(i(i(i(x,x),i(i(y,y),z)),z)).
100 [hyper,1,56,56] P(i(x,x)).
129 [hyper,1,13,42] P(i(i(x,i(y,z)),i(y,i(x,z)))).
138 [hyper,1,42,13] P(i(i(x,y),i(i(z,x),i(z,y)))).
181 [hyper,2,129,100,138] $ANS(all).
```

Within the 14-step proof, you find a 10-step proof of *I*, a 9-step proof of *C*, and a 9-step proof of *B*, in that order. The proofs of both *C* and *B*, within the 16-step proof that prompted the cramming experiment, each have length 7. So, again you have examples of trading short subproofs for longer subproofs when the length of the entire proof (of the join) is shortened. I do enjoy oddities like this one, which is, reasonably speaking, counterintuitive in that one might easily think that progress is occurring as the subproofs get shorter and shorter.

Now, what may seem like a radical departure, but I believe may prove quite interesting, I turn to some experiments that were running as I was writing the material of this section. Especially if you enjoy programming—but even if you do not—I suspect you may find what comes next stimulating and thought-provoking.

Of course, by this point in your reading, you are familiar with my fascination with numbers, those that signify how much CPU time was needed as well as those that denote how many conclusions were drawn at various points in the search. Also, you know I am delighted—I cannot find a better word—with large numbers. So the following will enable you to share, or witness, my reactions.

I again visited the *BCK* logic with the goal of finding a proof of length strictly less than 23. I decided to take a proof of the 19th step of a 23-step proof, a proof itself of length 12, and cram on it to see if I could find proofs of what I needed that led to a proof of length less than 23 of the join. The following proof brought me great excitement, as noted, because of the number of CPU-seconds and the numbers of the clauses involved. The proof, as you see, derives *B* in eight steps; *B* is in this logic the most difficult to prove of the three.

An Amazing Subproof

```
----> UNIT CONFLICT at 1044126.01 sec ----> 1326945 [binary,1326944.1,41.1] $ANS(B).

Length of proof is 8.  Level of proof is 6.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
3 [] P(i(i(i(u,v),w),i(i(x,i(w,y)),i(v,i(x,y))))).
8 [] P(i(x,i(i(y,i(x,z)),i(y,z)))).
11 [] P(i(i(i(x,y),i(z,i(u,v))),i(y,i(u,i(z,v))))).
12 [] P(i(i(x,i(y,z)),i(i(i(u,i(i(v,i(u,w)),i(v,w))),y),i(x,z)))).
14 [] P(i(i(x,y),i(i(i(z,i(i(u,i(z,v)),i(u,v))),x),y))).
15 [] P(i(i(x,i(i(i(y,z),i(i(i(u,i(i(v,i(u,w)),i(v,w))),y),z)),v6)),i(x,v6))).
41 [] -P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(B).
111 [hyper,1,12,14] P(i(i(i(x,i(i(y,i(x,z)),i(y,z))),i(i(u,i(i(v,i(u,w)),i(v,w))),v6)),i(i(v6,v7),v7))).
133 [hyper,1,15,3] P(i(i(i(x,y),i(i(z,i(i(u,i(z,v)),i(u,v))),w)),i(y,i(i(w,v6),v6)))).
1663 [hyper,1,8,133] P(i(i(x,i(i(i(i(y,z),i(i(u,i(i(v,i(u,w)),i(v,w))),v6)),i(z,i(i(v6,v7),v7))),v8)),i(x,v8))).
1682 [hyper,1,133,111] P(i(i(i(x,i(i(y,i(x,z)),i(y,z))),u),i(i(i(v,i(u,w)),i(v,w)),v6),v6))).
16245 [hyper,1,1663,1682] P(i(i(i(x,i(i(y,i(x,z)),i(y,z))),u),i(i(u,v),i(i(v,w),w)))).
101496 [hyper,1,11,16245] P(i(x,i(i(y,z),i(i(x,y),z)))).
495922 [hyper,1,12,101496] P(i(i(i(x,i(i(y,i(x,z)),i(y,z))),i(u,v)),i(w,i(i(w,u),v)))).
1326944 [hyper,1,11,495922] P(i(i(x,y),i(i(z,x),i(z,y)))).
```

Of course I immediately used this 8-step proof as resonators, with others, to see what type of proof, as well as how long, I would get with OTTER for the join of *B*, *C*, and *I* with Meredith's single axiom as sole hypothesis. I failed, and I succeeded. I did not get a proof shorter than 23. I did get a proof in which *B*' is not present, which indeed surprised me in that that formula occurs in a crucial manner in the Meredith 30-step proof and in both of the 23-step proofs I had in hand by this time. For your curiosity, and perhaps to further stimulate you to finding a proof of length strictly less than 23, I now include the proof OTTER discovered.

An Unusual 24-Step Proof in the BCK Logic

```
-----> EMPTY CLAUSE at   0.07 sec ----> 634 [hyper,2,43,233,543] $ANS(all).

Length of proof is 24.  Level of proof is 17.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] -P(i(i(a1,i(b,a2)),i(b,i(a1,a2)))) | -P(i(a1,i(b,a1))) | -P(i(i(a1,b),i(i(a2,a1),i(a2,b)))) | $ANS(all).
3 [] P(i(i(i(u,v),w),i(i(x,i(w,y)),i(v,i(x,y))))).
8 [hyper,1,3,3] P(i(i(x,i(i(i(y,i(z,u)),i(v,i(y,u))),w)),i(z,i(x,w)))).
9 [hyper,1,8,8] P(i(x,i(i(y,i(i(i(z,i(i(i(u,i(x,v)),i(w,i(u,v))),v6)),i(v7,i(z,v6))),v8)),i(y,v8)))).
12 [hyper,1,8,9] P(i(x,i(y,i(i(z,i(x,u)),i(z,u))))).
14 [hyper,1,8,12] P(i(x,i(y,i(i(z,i(y,u)),i(z,u))))).
18 [hyper,1,14,14] P(i(x,i(i(y,i(x,z)),i(y,z)))).
23 [hyper,1,18,18] P(i(i(x,i(i(y,i(i(z,i(y,u)),i(z,u))),v)),i(x,v))).
37 [hyper,1,23,23] P(i(i(i(x,i(i(y,i(x,z)),i(y,z))),i(i(u,i(i(v,i(u,w)),i(v,w))),v6)),v6)).
43 [hyper,1,23,18] P(i(i(x,i(y,z)),i(y,i(x,z)))).
48 [hyper,1,23,3] P(i(i(i(x,y),i(z,i(u,v))),i(y,i(u,i(z,v))))).
50 [hyper,1,3,37] P(i(i(x,i(y,z)),i(i(i(u,i(i(v,i(u,w)),i(v,w))),y),i(x,z)))).
82 [hyper,1,48,50] P(i(i(x,y),i(z,i(i(i(u,i(i(v,i(u,w)),i(v,w))),x),y)))).
98 [hyper,1,23,82] P(i(i(x,y),i(i(i(z,i(i(u,i(z,v)),i(u,v))),x),y))).
129 [hyper,1,50,98] P(i(i(i(x,i(i(y,i(x,z)),i(y,z))),i(i(u,i(i(v,i(u,w)),i(v,w))),v6)),i(i(v6,v7),v7))).
133 [hyper,1,18,98] P(i(i(x,i(i(i(y,z),i(i(i(u,i(i(v,i(u,w)),i(v,w))),y),z)),v6)),i(x,v6))).
138 [hyper,1,3,98] P(i(i(x,i(i(i(i(y,i(i(z,i(y,u)),i(z,u))),v),w),v6)),i(w,i(x,v6)))).
188 [hyper,1,133,9] P(i(x,i(i(i(y,i(z,i(i(u,i(x,v)),i(u,v)))),w),w))).
189 [hyper,1,133,3] P(i(i(i(x,y),i(i(z,i(i(u,i(z,v)),i(u,v))),w)),i(y,i(i(w,v6),v6)))).
233 [hyper,1,138,188] P(i(x,i(y,x))).
266 [hyper,1,18,189] P(i(i(x,i(i(i(i(y,z),i(i(u,i(i(v,i(u,w)),i(v,w))),v6)),i(z,i(i(v6,v7),v7))),v8)),i(x,v8))).
275 [hyper,1,189,129] P(i(i(i(x,i(i(y,i(x,z)),i(y,z))),u),i(i(i(i(v,i(u,w)),i(v,w)),v6),v6))).
327 [hyper,1,266,275] P(i(i(i(x,i(i(y,i(x,z)),i(y,z))),u),i(i(u,v),i(i(v,w),w)))).
418 [hyper,1,48,327] P(i(x,i(i(y,z),i(i(x,y),z)))).
461 [hyper,1,50,418] P(i(i(i(x,i(i(y,i(x,z)),i(y,z))),i(u,v)),i(w,i(i(w,u),v)))).
543 [hyper,1,48,461] P(i(i(x,y),i(i(z,x),i(z,y)))).
634 [hyper,2,43,233,543] $ANS(all).
```

Now for a radical change of pace and a new challenge. Specifically, I offer the following strategy (which I call *Dynamic resonator adjunction*), one that, if available in an automated reasoning program, I conjecture would add materially to its power.

First, the evidence for the value of using resonators and/or Veroff's hints to guide a program's reasoning is quite substantial, whether the goal is refining a proof or finding a first proof. The idea is this. Place in list(passive) a set of target lemmas. They are not necessarily provable in the theory under study. If and when any are proved, have the program pause, take the proof steps of the proved lemma(s), and adjoin them as resonators or as hints or as both and each with a small assigned value. I suggest a value smaller than being used for any included resonators or hints in the original input file. Then, reassign to all retained conclusions the same value to those formulas or equations that match, in the resonator sense (all variables treated as indistinguishable), any of the newly adjoined items. The idea is that new proof steps indicate progress and, therefore, they, or matching items, merit emphasis in guiding a program's reasoning.

You ask why this has not yet been added. The answer is simple: Other areas were under investigation. You ask for some evidence that the new strategy deserves attention; here is a bit of such.

Earlier, this notebook focused on two proofs of respective lengths 13 and 12, where the second was obtained much, much later than the first. The sole hypothesis, axiom, was Ulrich's eighteenth for this area of logic, and the target was Meredith 1. Not long before I was writing this very material, I tried the following experiment. I took the input file that led to the discovery of the two proofs and adjoined to it thirteen resonators, each with an assigned value (the same) smaller than any other present. I wondered whether the 12-step proof would be found sooner than in its birth. I had some doubt in that the 13-step and 12-step proof have so much in common. The area, as you recall, is the *BCI* logic.

To my delight, and even astonishment, the 12-step proof—the same found earlier—was discovered in less than 200 CPU-seconds. Quite soon after this success, I discussed with three of my colleagues my conclusion about the relevance of this bit of data to the value of dynamic resonator adjunction. Each (not out of politeness or friendship, I hope) thought the evidence relevant. Therefore, my challenge to that researcher who enjoys programming is to add, most likely to OTTER, this new strategy. To be totally clear: the result appears to show the strategy to be useful for finding first proofs, as well as finding proof refinements in the context of proof length.

7. Intermission and Overview

Perhaps your plate now runneth over. You might indeed enjoy at this time a break, an intermission. As for what will follow, yes, you are correct, another essay or notebook. Regarding what will be featured, I cannot say as yet; a number of choices are in view.

You have been offered, if you enjoy programming, various dishes. If, instead or in addition, you enjoy the solving of puzzles, you have been invited to find proofs shorter than the ones offered here. The two logics that are featured are the *BCK* logic and the *BCI* logic, the latter weaker than the former. However— I cannot stress this point enough—you need not know anything about either logic to attempt to solve a puzzle presented by it, if the puzzle is of the type featured in this notebook. More generally, in so many cases, if the goal is that of finding a proof shorter than known when your interest arises, I can assure you that you have a good chance of succeeding if the only obstacle is a lack of knowledge regarding the field of investigation.

I have not here given much insight into the means for choosing which commands to rely upon and which values to assign to the various parameters. In other notebooks, I do give comments about both aspects. For the curious, there do exist automated reasoning programs that are faster than is OTTER when, for example, measured in terms of the number of conclusions drawn per CPU-second. However, from what I know here in the latter part of 2007, no program offers the power that OTTER does when measured in terms of possible assignment completion. The key reason for this, so it seems, is the wide array of strategies McCune's marvelous program offers, for example, some to restrict the reasoning, some to direct it. For the record, in the area of designing and implementing automated reasoning, McCune has set a standard that might never be matched or exceeded. Tangential, but somewhat related, if you have a purported theorem that resists proof, Veroff is the researcher to contact when a reasoning program can play a key role.

Time asks for me to also partake of the intermission. I look forward to our next meeting in the next notebook. But you need not wait until then: I welcome e-mail communication. I would indeed appreciate, for example, theorems accompanied by a proof where the goal is to find a shorter proof. Ideally, I prefer that the activity be phrased in terms of the type of clause found here. Together we can experience the excitement and joy of using an automated reasoning program in some endeavor.