subformula.strategy
created 12-18-2007
revised last 05-18-2008

The Subformula Strategy: Coping with Complex Expressions*

Larry Wos

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

1. The Birth of a New Strategy

If an expression, formula or equation, is so powerful that it alone provides a complete axiomatization for some theory of logic or mathematics, what power is offered by its subexpressions? This notebook gives one answer to the question and, in doing so, introduces (in Section 2) a new strategy, to be called the *subformula strategy*, for directing an automated reasoning program's search for assignment completion. Of course, the question as posed was not the source of the research reported here. Instead, the source rests with the following general problem and also with a specific incarnation of it.

In general, if the target of a proof when completed is a formula or equation of substantial complexity, you can expect to encounter great difficulty. For example, the following four axioms, viewed as formulas, give you an axiom system for the implicational fragment *RI* of the relevance logic *R*. (The function *i* denotes relevant implication).

```
P(i(i(u,v),i(i(v,w),i(u,w)))).  % B'
P(i(i(u,i(v,w)),i(v,i(u,w)))).  % C
P(i(u,u)).  % I
P(i(i(u,i(u,v)),i(u,v))).  % W
```

I was intent, eventually, upon proving from those four axioms the following equivalent single axiom, constructed using the methods of the Romanian logician A. Rezus and supplied to me by D. Ulrich.

```
% Following is a Rezus-style single axiom for RI.
P(i(i(i(x,i(i(i(y,y),i(i(z,z),i(i(u,u),i(i(v,v),i(x,w))))),w)),i(i(i(i(i(v6,v7),i(i(v7,v8),i(v6,v8))),
  i(i(i(i(i(v9,i(v9,v10)),i(v9,v10)),i(i(i(v11,v12),i(i(v12,v13),i(v11,v13))),v14)),v14),v15)),v15),
  (i(v16,i(i(i(v17,v17),i(i(v18,v18),i(i(v19,v19),i(i(v20,v20),i(v16,v21))))),v21)),v22))),v22)).
  % Rezus-style single axiom for RI
```

If successful, I would eventually have a proof that relied for hypotheses on *B'*, *C*, *I*, and *W* and completed with the deduction of the Rezus formula (or a generalization of it). (I am delighted to note that Ulrich has himself found a much shorter single axiom for *RI*, the following.)

```
P(i(i(i(i(i(u,v),i(w,u)),i(i(u,v),i(w,v))),i(i(x,x),i(y,i(z,v6)))),i(i(v7,y),i(z,i(v7,v6))))).  % Ulrich single axiom
```

This axiom is presented and discussed on Ulrich's own website at http://web.ics.purdue.edu/~dulrich/C-pure-R-page.htm. Ulrich's result is indeed most impressive.

The program I chose to rely upon was William McCune's automated reasoning program OTTER. (Although the discussion throughout is often in terms of the mechanisms offered by OTTER—as is the case in the various notebooks of mine you may read—the ideas that are offered do not depend on the use of this fine program.) The inference rule I chose was hyperresolution, chosen because the proof was

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

to employ *condensed detachment* (expressed with the following clause, where '' | '' denotes logical **or**, ''-'' denotes logical **not**), and the predicate *P* denotes ''provability''.

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

Rather than the study of the given Rezus-style formula, it was actually my work on deriving Ulrich's axiom from *B'*, *C*, *I*, and *W* that initiated my development of the new subformula strategy, a strategy that is iterative in nature. Here, however, to explain this strategy and illustrate its effectiveness, I shall use instead the goal of proving the far longer Rezus formula from the cited four-axiom system.

With one exception, to direct its search for assignment completion, OTTER prefers to focus on syntactically short or simple clauses. The exception concerns the use of Ross Overbeek's *weighting strategy*, whose use (shown in an input file to be given) enables the user to override the computation of clause complexity in terms of symbol count. For example, you can include a long formula coupled with the (in effect) appropriate instruction to cause OTTER to treat it as quite short. To illustrate how weighting works, consider the following *weight template*.

weight(P(i(i(i(i(u,v),i(w,u)),i(i(u,v),i(w,v))),i(i(x,x),i(y,i(z,v6)))),i(i(v7,y),i(z,i(v7,v6))))),1).

The inclusion of the item just given has an automated reasoning program, that offers weighting, treat the formula that appears as consisting of a single symbol because of the assigned weight of 1. (You probably have noticed that the included formula is Ulrich's 35-symbol single axiom.) Since all variables in such a template are treated by OTTER as indistinguishable (from each other), any expression that matches the given template, at the functional level, is also treated as consisting of a single symbol. In other words, if you replaced the given template with a similar one, but with all variables being *x*, the program would behave as it did with the given template. If your goal was to proof-check a given proof, and if that proof contained very long expressions, OTTER could easily assist you provided you employ Overbeek's weighting strategy. You would merely include each of the proof steps and inform the program, by assigning small weights to them, to treat all given proof steps as short or simple expressions. In such an event, most likely, you would be the recipient of an appropriate proof.

However, if you had little or no idea about how a sought-after proof might proceed—which was the case when I studied the Rezus formula—then you would not know which formulas or equations to assign small values (weights) to. If your goal, as was mine, was to prove the Rezus formula from the first four of the five given formulas, a program of OTTER's type might never succeed. Indeed, any automated reasoning program would almost certainly wander among clauses far, far shorter than that for Rezus, never getting close to it. To generalize the given incarnation, if the target was a most complex formula or equation, how in the world could you help your program? As an obvious aside, people on their own usually do not find it an easy task to aim at a very complex target.

Before turning to the actual new strategy, one additional approach merits discussion, that of *level saturation*. Over the years, many, many times, somebody has suggested the following attack on proof finding. Let the level of an input clause be 0, and the level of a deduced clause be one greater than the maximum of the levels of its parents. In level saturation, you tell the program (or person, if you wish to be amusing) to deduce all clauses of level 1, then of level 2, then of level 3, and so on until th sought-after proof has been found. Of course, you impose one or more constraints, such as a bound on the complexity of retained information. In principle, if the set of hypotheses is sufficient to lead to a deduction of the target, and (of course) if the constraints are not too severe, level saturation will return a proof. However, to the surprise and even dismay of many, level saturation is in a large fraction of cases impractical; the size of the levels grows much faster than might be expected. If the target offers substantial complexity, as is the case with Rezus, then—because the complexity bound must be assigned a correspondingly large number—the size grows very rapidly, sometimes exponentially.

Summarizing, at least for the automated reasoning programs with which I am familiar, the task of finding a proof in which quite complex expressions are expected to be present is most formidable. When the complexity is more than 70 (measured in symbol count), the sought-after proof is most likely out of range unless some direction strategy is brought into play.

2. The New Strategy

The following strategy, as well as variants of it, has in preliminary experiments led to finding the sought-after proofs. It has been used when the target is a most complex formula, in particular, the Rezus formula. It has been used at the other end of the spectrum, namely, when the complex formula is the hypothesis. Briefly, the basic idea is to take all of the nontrivial subexpressions of the complex element under study and include, for each, in the input file a weight template with a small assigned value. Weight templates are used to direct the program's reasoning toward more profitable paths of inquiry, if all goes well. The smaller the assigned value, the greater the priority given to the formula that contains a relevant expression (to be explained almost immediately). So far, I have not experimented with the inclusion of subexpressions that are themselves variables, nor have I yet tried the strategy in areas where equality is present. Variants will be discussed, but first the promised details and an example is in order with an accompanying input file that is relevant to the variants.

An important difference exists between the following two templates.

```
weight(P(i(x,x)),1).
weight(i(x,x),1).
```

As for the first of the two, which is called a *resonator* (a concept to be discussed shortly), its inclusion instructs a program to assign a value of 1 to any clause that is of the same form, even such as p(i(x,y). The triviality of the first might be deceptive. Indeed, if the first of the two is replaced by a far, far more complex formula, then matching formulas, although long (complex) in symbol count, are treated as having length 1. In contrast, the second of the two given templates, being free of a predicate symbol, is relevant to proper subexpressions of formulas. Each subexpression of a formula of the form *i* of variable, variable is given weight 1. Therefore, a very long (complex formula will, in the presence of the second of the two templates, have its assigned weight (in effect, length) be a number quite a bit smaller than the value based purely on symbol count. Yes, rather than the second, or in addition to the second, you can include a number of different with templates, each of which is free of a predicate symbol and each used to compute an assigned value to formulas.

Now for the promised discussion of a resonator and the *resonance strategy*. The objective of the resonance strategy is to enable the researcher to include (for OTTER, as weight templates) equations or formulas, called *resonators*, that will be used to direct a program's reasoning. A resonator is a symbol pattern (all of whose variables are considered indistinguishable) conjectured to merit special attention because it, and any matching expression, has particular appeal. To each resonator one assigns a value reflecting its relative (conjectured) significance; the smaller the value, the greater the significance. You could, for example, have in mind a set of steps that are promising in the context of finding a new proof or a first proof. In such an event, you might include each of those steps, as a resonator, to direct the program's reasoning accordingly.

Finally, the promised example is in focus.

Classical two-valued propositional calculus admits a number of axiom systems including the following single axiom due to Lukasiewicz.

```
% Following is Lukasiewicz's 23-letter single axiom.
P(i(i(i(x,y),i(i(i(n(z),n(u)),v),z)),i(w,i(i(z,x),i(u,x))))).
```

The function *i* denotes implication, and the function *n* denotes negation. When Lukasiewicz offered this formula as a single axiom in the 1930s, he (to the surprise of many) did not include a proof that the formula is in fact a single axiom, nor did he include information about the nature of such a proof. One way to prove the corresponding theorem is to deduce some known axiom system, say, the following also due to Lukasiewicz.

```
% Luka 1 2 3.
P(i(i(x,y),i(i(y,z),i(x,z)))).
P(i(i(n(x),x),x)).
P(i(x,i(n(x),y))).
```

At this point in the narrative, the focus is on the use of the new subformula strategy when the complexity is evidenced in the hypothesis, the Lukasiewicz 23-letter formula. Although the given

Lukasiewicz 3-axiom system is the main target, other axiom systems found in the input file shortly included would also suffice.

First, you need for the subformula strategy the eight nontrivial subformulas of the Lukasiewicz axiom, those that involve at least one occurrence of the function *i*. The following list was sent to me by e-mail by my colleague Ross Overbeek, produced with a program he wrote for that purpose.

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

Of course, some of the given eight formulas are not theorems of the classical propositional calculus. Being a theorem is not the key; rather, each of these eight formulas will be used in the subformula strategy to direct OTTER's reasoning in the attempt to find a proof that deduces a known axiom system. Elements used by the strategy do not, therefore, take on a **true** or **false** value; none are assumed, for example, to be lemmas, nor are they used as such. Further, as noted, OTTER treats all variables in a weight template as indistinguishable, just noting that they are variables.

The following input file contains far more than is needed for a discussion of the subformula strategy in its simplest form, but it will serve well for later discussions.

An Input File for the Subformula Strategy

```
% Trying for an automated proof from the Lukasiewicz 23-letter single axiom.
set(hyper_res).
assign(max_weight,24).
% assign(change_limit_after,2000).
% assign(new_max_weight,24).
assign(max_proofs,-1).
clear(print_kept).
% set(process_input).
clear(print_back_sub).
assign(max_distinct_vars,6).
assign(pick_given_ratio,6).
assign(max_mem,880000).
assign(report,3600).
set(order_history).
set(input_sos_first).
set(ancestor_subsume).
set(back_sub).
assign(heat,0).
% assign(dynamic_heat_weight,0).

weight_list(pick_and_purge).
% Following 8 are subformulas of Luka-23 letter and the formula itself.
weight(i(n(z),n(u)),2).
weight(i(i(z,x),i(u,x)),2).
weight(i(i(n(z),n(u)),v),2).
weight(i(w,i(i(z,x),i(u,x))),2).
weight(i(i(i(n(z),n(u)),v),z),2).
weight(i(i(x,y),i(i(i(n(z),n(u)),v),z)),2).
weight(i(i(i(x,y),i(i(i(n(z),n(u)),v),z)),i(w,i(i(z,x),i(u,x)))),2).
weight(P(i(i(i(x,y),i(i(i(n(z),n(u)),v),z)),i(w,i(i(z,x),i(u,x))))),2).
```

```
% % Following are members of known axiom systems for two-valued.
% weight(P(i(i(x,y),i(i(y,r),i(x,r)))),1).
% weight(P(i(i(n(x),x),x)),1).
% weight(P(i(x,i(n(x),y))),1).
% weight(P(i(y,i(x,y))),1).
% weight(P(i(i(i(x,y),z),i(y,z))),1).
% weight(P(i(i(x,i(y,z)),i(y,i(x,z)))),1).
% weight(P(i(i(y,z),i(i(x,y),i(x,z)))),1).
% weight(P(i(i(x,i(x,y)),i(x,y))),1).
% weight(P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))),1).
% weight(P(i(i(i(x,y),z),i(n(x),z))),1).
% weight(P(i(n(n(x)),x)),1).
% weight(P(i(x,n(n(x)))),1).
% weight(P(i(i(x,y),i(n(y),n(x)))),1).
% weight(P(i(i(n(x),n(y)),i(y,x))),1).
% weight(P(i(i(x,y),i(i(n(x),y),y))),1).
% weight(P(i(i(n(x),z),i(i(y,z),i(i(x,y),z)))),1).
% weight(P(i(i(u,i(n(x),z)),i(u,i(i(y,z),i(i(x,y),z))))),1).
% % Following are theses 4 through 71.
% weight(P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))),2).
% weight(P(i(i(x,i(y,z)),i(i(u,y),i(x,i(u,z))))),2).
% weight(P(i(i(x,y),i(i(i(x,z),u),i(i(y,z),u)))),2).
% weight(P(i(i(x,i(i(y,z),u)),i(i(y,v),i(x,i(i(v,z),u))))),2).
% weight(P(i(i(x,y),i(i(z,x),i(i(y,u),i(z,u))))),2).
% weight(P(i(i(i(n(x),y),z),i(x,z))),2).
% weight(P(i(x,i(i(i(n(x),x),x),i(i(y,x),x)))),2).
% weight(P(i(i(x,i(i(n(y),y),y)),i(i(n(y),y),y))),2).
% weight(P(i(x,i(i(n(y),y),y))),2).
% weight(P(i(i(n(x),y),i(z,i(i(y,x),x)))),2).
% weight(P(i(i(i(i(x,i(i(y,z),z)),u),i(i(n(z),y),u))),2).
% weight(P(i(i(n(x),y),i(i(y,x),x))),2).
% weight(P(i(x,x)),2).
% weight(P(i(x,i(i(y,x),x))),2).
% weight(P(i(x,i(y,x))),2).
% weight(P(i(i(i(x,y),z),i(y,z))),2).
% weight(P(i(x,i(i(x,y),y))),2).
% weight(P(i(i(x,i(y,z)),i(y,i(x,z)))),2).
% weight(P(i(i(x,y),i(i(z,x),i(z,y)))),2).
% weight(P(i(i(i(i(x,i(y,z)),u),i(i(y,i(x,z)),u))),2).
% weight(P(i(i(i(x,y),x),x)),2).
% weight(P(i(i(i(x,y),z),i(i(x,u),i(i(u,y),z)))),2).
% weight(P(i(i(i(x,y),z),i(i(z,x),x))),2).
% weight(P(i(i(i(x,y),y),i(i(y,x),x))),2).
% weight(P(i(i(i(i(x,y),y),z),i(i(i(y,u),x),z))),2).
% weight(P(i(i(i(x,y),z),i(i(x,z),z))),2).
% weight(P(i(i(x,i(x,y)),i(x,y))),2).
% weight(P(i(i(x,y),i(i(i(x,z),u),i(i(y,u),u)))),2).
% weight(P(i(i(i(x,y),z),i(i(x,u),i(i(u,z),z)))),2).
% weight(P(i(i(x,y),i(i(y,i(z,i(x,u))),i(z,i(x,u))))),2).
% weight(P(i(i(x,i(y,i(z,u))),i(i(z,x),i(y,i(z,u))))),2).
% weight(P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))),2).
% weight(P(i(n(x),i(x,y))),2).
% weight(P(i(i(i(x,y),z),i(n(x),z))),2).
% weight(P(i(i(x,n(x)),n(x))),2).
% weight(P(i(n(n(x)),x)),2).
```

```
%  weight(P(i(x,n(n(x)))),2).
%  weight(P(i(i(x,y),i(n(n(x)),y))),2).
%  weight(P(i(i(i(n(n(x)),y),z),i(i(x,y),z))),2).
%  weight(P(i(i(x,y),i(i(y,n(x)),n(x)))),2).
%  weight(P(i(i(x,i(y,n(z))),i(i(z,y),i(x,n(z))))),2).
%  weight(P(i(i(x,i(y,z)),i(i(n(z),y),i(x,z)))),2).
%  weight(P(i(i(x,y),i(n(y),n(x)))),2).
%  weight(P(i(i(x,n(y)),i(y,n(x)))),2).
%  weight(P(i(i(n(x),y),i(n(y),x))),2).
%  weight(P(i(i(n(x),n(y)),i(y,x))),2).
%  weight(P(i(i(i(n(x),y),z),i(i(n(y),x),z))),2).
%  weight(P(i(i(x,i(y,z)),i(x,i(n(z),n(y))))),2).
%  weight(P(i(i(x,i(y,n(z))),i(x,i(z,n(y))))),2).
%  weight(P(i(i(n(x),y),i(i(x,y),y))),2).
%  weight(P(i(i(x,y),i(i(n(x),y),y))),2).
%  weight(P(i(i(x,y),i(i(x,n(y)),n(x)))),2).
%  weight(P(i(i(i(i(x,y),y),z),i(i(n(x),y),z))),2).
%  weight(P(i(i(n(x),y),i(i(x,z),i(i(z,y),y)))),2).
%  weight(P(i(i(i(i(x,y),i(i(y,z),z)),u),i(i(n(x),z),u))),2).
%  weight(P(i(i(n(x),y),i(i(z,y),i(i(x,z),y)))),2).
%  weight(P(i(i(x,i(n(y),z)),i(x,i(i(u,z),i(i(y,u),z))))),2).
%  weight(P(i(i(x,y),i(i(z,y),i(i(n(x),z),y)))),2).
%  weight(P(i(i(n(n(x)),y),i(x,y))),2).
%  weight(P(i(x,i(y,y))),2).
%  weight(P(i(n(i(x,x)),y)),2).
%  weight(P(i(i(n(x),n(i(y,y))),x)),2).
%  weight(P(i(n(i(x,y)),x)),2).
%  weight(P(i(n(i(x,y)),n(y))),2).
%  weight(P(i(n(i(x,n(y))),y)),2).
%  weight(P(i(x,i(n(y),n(i(x,y))))),2).
%  weight(P(i(x,i(y,n(i(x,n(y)))))),2).
%  weight(P(n(i(i(x,x),n(i(y,y))))),2).
%  %  Following is Lukasiewicz's 23-letter single axiom.
%  weight(P(i(i(i(x,y),i(i(i(n(z),n(u)),v),z)),i(w,i(i(z,x),i(u,x))))),1).
end_of_list.

list(usable).
% condensed detachment
-P(i(x,y)) | -P(x) | P(y).
%  The following disjunctions are known axiom systems.
-P(i(q,i(p,q))) | -P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) | -P(i(n(n(p)),p)) | -P(i(p,n(n(p)))) |
  -P(i(i(p,q),i(n(q),n(p)))) | -P(i(i(p,i(q,r)),i(q,i(p,r)))) |
  $ANS(step_allFrege_18_35_39_40_46_21). % 21 is dependent.
-P(i(q,i(p,q))) | -P(i(i(p,i(q,r)),i(q,i(p,r)))) | -P(i(i(q,r),i(i(p,q),i(p,r)))) | -P(i(p,i(n(p),q))) |
  -P(i(i(p,q),i(i(n(p),q),q))) | -P(i(i(p,i(p,q)),i(p,q))) | $ANS(step_allHilbert_18_21_22_3_54_30).
  % 30 is dependent.
-P(i(q,i(p,q))) | -P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) | -P(i(i(n(p),n(q)),i(q,p))) |
   $ANS(step_allBEH_Church_FL_18_35_49).
-P(i(i(i(p,q),r),i(q,r))) | -P(i(i(i(p,q),r),i(n(p),r))) | -P(i(i(n(p),r),i(i(q,r),i(i(p,q),r)))) |
   $ANS(step_allLuka_x_19_37_59).
-P(i(i(i(p,q),r),i(q,r))) | -P(i(i(i(p,q),r),i(n(p),r))) | -P(i(i(s,i(n(p),r)),i(s,i(i(q,r),i(i(p,q),r))))) |
   $ANS(step_allWos_x_19_37_60).
-P(i(i(p,q),i(i(q,r),i(p,r)))) | -P(i(i(n(p),p),p)) | -P(i(p,i(n(p),q))) | $ANS(step_allLuka_1_2_3).
end_of_list.
```

```
list(sos).
%  Following is Lukasiewicz's 23-letter single axiom.
P(i(i(i(x,y),i(i(i(n(z),n(u)),v),z)),i(w,i(i(z,x),i(u,x))))).
end_of_list.

list(passive).
%  Following are members of known axiom systems for two-valued.
-P(i(i(p,q),i(i(q,r),i(p,r)))) | $ANS(step_L1).
-P(i(i(n(p),p),p)) | $ANS(step_L2).
-P(i(p,i(n(p),q))) | $ANS(step_L3).
-P(i(q,i(p,q))) | $ANS(step_18).
-P(i(i(i(p,q),r),i(q,r))) | $ANS(step_19).
-P(i(i(p,i(q,r)),i(q,i(p,r)))) | $ANS(step_21).
-P(i(i(q,r),i(i(p,q),i(p,r)))) | $ANS(step_22).
-P(i(i(p,i(p,q)),i(p,q))) | $ANS(step_30).
-P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) | $ANS(step_35).
-P(i(i(i(p,q),r),i(n(p),r))) | $ANS(step_37).
-P(i(n(n(p)),p)) | $ANS(step_39).
-P(i(p,n(n(p)))) | $ANS(step_40).
-P(i(i(p,q),i(n(q),n(p)))) | $ANS(step_46).
-P(i(i(n(p),n(q)),i(q,p))) | $ANS(step_49).
-P(i(i(p,q),i(i(n(p),q),q))) | $ANS(step_54).
-P(i(i(n(p),r),i(i(q,r),i(i(p,q),r)))) | $ANS(step_59).
-P(i(i(s,i(n(p),r)),i(s,i(i(q,r),i(i(p,q),r))))) | $ANS(step_60).
%  %  Following are negations of theses 4-71.
% -P(i(i(i(i(q,r),i(p,r)),s),i(i(p,q),s))) | $ANS(neg_th_04).
% -P(i(i(p,i(q,r)),i(i(s,q),i(p,i(s,r))))) | $ANS(neg_th_05).
% -P(i(i(p,q),i(i(i(p,r),s),i(i(q,r),s)))) | $ANS(neg_th_06).
% -P(i(i(t,i(i(p,r),s)),i(i(p,q),i(t,i(i(q,r),s))))) | $ANS(neg_th_07).
% -P(i(i(q,r),i(i(p,q),i(i(r,s),i(p,s))))) | $ANS(neg_th_08).
% -P(i(i(i(n(p),q),r),i(p,r))) | $ANS(neg_th_09).
% -P(i(p,i(i(i(n(p),p),p),i(i(q,p),p)))) | $ANS(neg_th_10).
% -P(i(i(q,i(i(n(p),p),p)),i(i(n(p),p),p))) | $ANS(neg_th_11).
% -P(i(t,i(i(n(p),p),p))) | $ANS(neg_th_12).
% -P(i(i(n(p),q),i(t,i(i(q,p),p)))) | $ANS(neg_th_13).
% -P(i(i(i(t,i(i(q,p),p)),r),i(i(n(p),q),r))) | $ANS(neg_th_14).
% -P(i(i(n(p),q),i(i(q,p),p))) | $ANS(neg_th_15).
% -P(i(p,p)) | $ANS(neg_th_16).
% -P(i(p,i(i(q,p),p))) | $ANS(neg_th_17).
% -P(i(q,i(p,q))) | $ANS(neg_th_18).
% -P(i(i(i(p,q),r),i(q,r))) | $ANS(neg_th_19).
% -P(i(p,i(i(p,q),q))) | $ANS(neg_th_20).
% -P(i(i(p,i(q,r)),i(q,i(p,r)))) | $ANS(neg_th_21).
% -P(i(i(q,r),i(i(p,q),i(p,r)))) | $ANS(neg_th_22).
% -P(i(i(i(q,i(p,r)),s),i(i(p,i(q,r)),s))) | $ANS(neg_th_23).
% -P(i(i(i(p,q),p),p)) | $ANS(neg_th_24).
% -P(i(i(i(p,r),s),i(i(p,q),i(i(q,r),s)))) | $ANS(neg_th_25).
% -P(i(i(i(p,q),r),i(i(r,p),p))) | $ANS(neg_th_26).
% -P(i(i(i(p,q),q),i(i(q,p),p))) | $ANS(neg_th_27).
% -P(i(i(i(i(r,p),p),s),i(i(i(p,q),r),s))) | $ANS(neg_th_28).
% -P(i(i(i(p,q),r),i(i(p,r),r))) | $ANS(neg_th_29).
% -P(i(i(p,i(p,q)),i(p,q))) | $ANS(neg_th_30).
% -P(i(i(p,s),i(i(i(p,q),r),i(i(s,r),r)))) | $ANS(neg_th_31).
% -P(i(i(i(p,q),r),i(i(p,s),i(i(s,r),r)))) | $ANS(neg_th_32).
% -P(i(i(p,s),i(i(s,i(q,i(p,r))),i(q,i(p,r))))) | $ANS(neg_th_33).
```

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

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

To set the stage for a discussion of the set of experiments that succeeded and that show that the subformula strategy is promising, some comments on the given input file are in order. Some of the comments explain in part why I made certain decisions. If you find that you wish to read about the

strategy immediately, I recommend skipping to the section entitled The Strategy in Action. The set of support strategy is in use with the only element in the (initial) set of support being the Lukasiewicz 23-letter formula. No other element of the input file is permitted to initiate an application of an inference rule. The list(usable) contains the clause (used together with the inference rule of hyperresolution) for condensed detachment as well as clauses corresponding to the negations of known axiom systems. I placed in list(passive) the negations of members of known axiom systems to monitor progress. (When a per cent sign occurs on a line, from that point forward on the line, OTTER treats the material as a comment.) The elements in list(demodulators) are used in a nonstandard manner, namely, to purge unwanted information upon generation. Here, for example, I informed OTTER to purge any deduced formula that contained three immediately nested occurrences of the function *n* (negation). Finally, in the context of lists, for the first experiment, the only templates that guided the program's attempt to complete a proof are those corresponding to the eight subformulas of the Lukasiewicz axiom, including the axiom itself. (The additional weight templates are given to hint at variants of the subformula strategy.)

As for the ''assign'' statements, I instructed the program to discard any new conclusion whose weight strictly exceeds 24. The weight will be measured in symbol count unless one of the included (active) weight templates modifies the weight of a clause. The assignment of -1 to max_proofs has the program find as many proofs as it can within time and memory limits. The assignment of 6 to max_distinct_vars informs the program that newly deduced conclusions that rely on seven or more distinct variables are to be discarded. By assigning the value 6 to the pick_given_ratio, OTTER is told to select for inference-rule application six clauses by complexity, one by first come first serve, then six, then one, and so on. (William McCune formulated this most useful strategy some years ago.) For now, the other assign statements are of less importance.

As for the ''set'' statements, set(hyper_res) has the program rely on hyperresolution as the inference rule, chosen because condensed detachment is under study. The inclusion of the set(order_history) command causes the program to list parents (of a deduced conclusion) in the order nucleus (for hyperresolution), major premiss, minor premiss. (Again I remark that the spelling of the word ''premiss'' is in keeping with the very strong recommendation of A. Church, a spelling that I have adhered to throughout my writings.) The set(input_sos_first) command instructs the program to focus on all members of the initial set of support, for initiating inference-rule application, before focusing on any deduced clause. In the case under discussion, it has no effect because the initial set of support contains a single element. For the set(ancestor_subsume), again McCune's contributions come into play. That command has the program, when two copies of a conclusion are deduced, (in effect) keep and prefer that which is reached by the shorter path. When that command is included, you had best always include set(back_sub) to rely on back subsumption. (I think it a fair conjecture that McCune formulated and encoded ancestor subsumption for me to use in my quest for ever-shorter proofs. Whether such is the case, as you will see in this notebook as well as others, I employ the procedure almost constantly.)

The Strategy in Action

In one of my earliest experiments, I had the program depend for guidance on the eight given subformulas (including the 23-letter formula itself), each assigned a weight of 2. I also included for guidance weight templates corresponding to the seventeen members of known axiom systems, each assigned a weight of 1, giving them even greater preference. My goal, as you would assert, was to obtain a proof that deduces all members of one of the known axiom systems cited in the given input file. There exist seventeen different members of the various axiom systems. I chose to rely on ancestor subsumption, conjecturing that to do otherwise might produce too many possible weight templates for the next iteration. Yes, I was rather certain that this early attempt would yield only partial success. In fact, the experiment proved ten of the seventeen members, but failed to prove all of the members of any of the target axiom systems.

In no way was I disappointed. After all, the theorem establishing the Lukasiewicz 23-letter formula to be a single axiom is far, far from easy to prove. Indeed, from what I know, until I obtained (most likely on May 25, 1999, using an approach quite different from the one under discussion here)) a proof with OTTER, the literature offered no proof. (I shall give, in Section 3, some comparisons in the

context of the Lukasiewicz single axiom shortly.) I was also not perturbed because I knew I could again call on Overbeek to return to me appropriate subformulas if needed. They were needed.

With a reminder that the proof itself was less important than the testing of this new strategy, a next experiment was in order. For that experiment, I sent to Overbeek eighty formulas, proof steps of the cited ten members of known axiom systems, sorted to remove duplicates. He returned to me a rather large set of subformulas, including the eighty I had sent him. From that set, I selected two subsets: the first consisted of the eighty formulas, each with an assigned weight of 0 to give them preference when and if such was deduced. For my second subset to be made into weight templates (so to speak), I chose 241, those that were proper subformulas and that also contained at least one occurrence of the function i. To each of these 241 I assigned a weight of 2, giving each less preference for guiding the reasoning when compared with the eighty proof steps. I continued to include the eight templates used in the first experiment, still assigned a value of 2. I also included weight templates, each assigned a weight of 1, corresponding to the seventeen members of known axiom systems. In other words, I modified the given input file by removing per cent signs from seventeen elements and inserted 80 plus 241 additional weight templates.

I made two more changes; I commented out the ancestor_subsume command because the use of that procedure can be quite time consuming. You see, at least two of the target axiom systems had the property that all but one of their members had been proven in the preceding experiment. In particular, if the first of the three Lukasiewicz axioms (found in the passive list in negated form) could be proved, the objective would have been reached. Similarly, if the formula that Lukasiewicz calls thesis 35 could be proved (to augment what had already been proved), again the goal would be reached. These two formulas are often quite difficult to prove, of course depending on what is present in the input file, for example, what knowledge is imparted to the program. Therefore, I thought it advisable to (in effect) speed things up by not relying on ancestor subsumption. As for the second change, also prompted by the knowledge of the difficulty to be encountered, I had the program avoid retaining any so-called double-negation-expressions, expressions that contained $n(n(t))$, for some term t, at any point. I had learned many years ago that such a move often promoted assignment completion.

Sometimes one or more Greek gods cheer for you. Indeed, this experiment under discussion succeeded in proving the so-called missing formulas, those needed to complete different axiom systems. Specifically, five different axiom systems were derived, the first in approximately 167 CPU-seconds, and the last in approximately 5542 CPU-seconds. Since computers vary widely in how fast they complete assignments, the following is perhaps more relevant. The first (consisting of the well-known three-axiom system of Lukasiewicz, Lukasiewicz 1, 2, 3) relied on a retained clause numbered 85026. The fifth and last whose proof was completed relied on a retained clause numbered 438349; that system was introduced by me in research quite a few years ago. I was somewhat surprised that but two crucial experiments sufficed, especially in view of the fact that the literature had not for decades offered a proof for the Lukasiewicz 23-letter formula.

A related experiment was in order, and I chose to focus on a single axiom for classical two-valued sentential calculus due to C. A. Meredith, the following.

```
% Following is Meredith.
P(i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x)))).
```

This single axiom is the shortest of its kind so far found, and, further, strong evidence exists that no shorter exists. Not counting the predicate (as was the case with Lukasiewicz), Meredith's axiom is a 21-letter formula. The study of the Meredith formula will be offered after certain comparisons regarding the Lukasiewicz 23-letter formula are made.

3. Comparison of Two Approaches

When I began assembling these notes, my main goal was to offer a new strategy, the subformula strategy, and present some evidence of its potential. As indicated, I was motivated by a serious obstacle that exists in automated reasoning, especially for programs that are reminiscent of McCune's OTTER. No, according to all I can find, the problem does not rest with OTTER. Rather it rests with the huge, huge space of conclusions that can typically be drawn when making a study in, say, mathematics or logic.

The obstacle, as you almost certainly have identified, is that presented when, say, the target is an extremely complex expression. Indeed, when Ulrich constructed for me the following Rezus-style 93-symbol formula that is itself a single axiom for the implicational fragment of *R*, I began wondering about how one could prove this formula to be a theorem of that logic.

```
P(i(i(i(x,i(i(i(y,y),i(i(z,z),i(i(u,u),i(i(v,v),i(x,w))))),w)),i(i(i(i(i(v6,v7),i(i(v7,v8),i(v6,v8))),
  i(i(i(i(i(v9,i(v9,v10)),i(v9,v10)),i(i(i(v11,v12),i(i(v12,v13),i(v11,v13))),v14)),v14),v15)),v15),
  i(i(v16,i(i(i(v17,v17),i(i(v18,v18),i(i(v19,v19),i(i(v20,v20),i(v16,v21))))),v21)),v22))),v22)).
  % Rezus-style single axiom for RI
```

Of course, I would rely on OTTER, my companion in automated reasoning for almost twenty years. But, I asked, how in the world could the program be encouraged to consider such large expressions? I could not simply rely on R. Veroff's hints or on my resonance strategy because I had no idea what to include in the input file in the context of hints or resonators. Level saturation was clearly out of the question. I then recalled a conversation with Overbeek, one focusing on aspects of his weighting strategy. And the grapes were harvested, and the wine-making begun.

I had no intention of comparing the use of the subformula strategy with any other approach and had little or no interest in comparing diverse results. I would have followed my intentions had it not been for certain (to me, and I hope to some others) astounding paths that I pursued because of data obtained during the experimentation reported here. That data must wait for a later section; at this point, another approach merits a brief discussion and, in doing so, a foretaste of a fine dessert is provided. The stage will then be set for results that I was in no way prepared for. Enough of this preamble.

The first proof, with the Lukasiewicz 23-letter formula as sole hypothesis, of a known axiom system obtained by using the subformula strategy has length 111 and level 39, deducing the following system. (Recall that the level of an input clause is 0; the level of a deduced clause is one greater than the maximum of the levels of its parents.)

```
%  Luka 1 2 3.
P(i(i(x,y),i(i(y,z),i(x,z)))).
P(i(i(n(x),x),x)).
P(i(x,i(n(x),y))).
```

That proof avoids the use of double-negation terms, and it relies on formulas whose variable richness is six (six distinct variables occur in some formulas). I shall give you the proof, and you might see how short a proof you can find starting with it. I spent a fair amount of time in the given context with the goal of getting one or more startling new results. As part of history, I note that years earlier (in 1999) I had found a 160-step proof that relied solely for hypothesis on the Lukasiewicz 23-letter single axiom and derived the 3-axiom system of Lukasiewicz. A glance at the two proofs, that of length 111 and that of length 160, provides some incentive in that regard; indeed, the 111-step proof contains 83 formulas not present in the 160-step proof. Perhaps because my studies of proof shortening in the context of the Lukasiewicz single axiom yielded little of note, as you will learn, I made the corresponding studies in the context of another single axiom (Meredith's 21-letter single axiom for the classical propositional calculus), which I shall discuss in a later section. Those studies, thanks to Overbeek, fulfill the implied promise concerning ''startling results''. The proof I found in 1999 (as I noted earlier) has length 160 and level 74; it avoids the use of double negation, and the proof has variable richness six. But how was the ''first'' proof obtained, that showing the Lukasiewicz 23-letter formula to be a single axiom for classical propositional calculus? Perhaps an understanding of the approach will shed some light on the radical differences in length, level, and nature of the two cited proofs.

The approach I used to eventually obtain the cited 160-step proof is called *lemma adjunction*. The following cursory treatment will make clearer the elements of the subformula strategy. In lemma adjunction, you choose a set of targets to prove. They may include formulas or equations that are unprovable because they are not theorems of the field under study. Their negations are placed in list(passive). You might, for example, choose (from Lukasiewicz) theses 4 through 71; see the input file. Like the subformula strategy, the lemma-adjunction approach is iterative in nature. In particular, the lemmas that are proved in the first stage are adjoined in the second stage to list(sos). Here you see an important difference between this approach and that which is central to this notebook; indeed, in lemma adjunction you do adjoin items that take on the value **true**. You include the command

set(input_sos_first) to have the program consider each of the items in the initial set of support for inference-rule application before any deduced item is so chosen.

If the second in the series of moves does not produce what is wanted, you take in the third stage the (new) items proved in the second and adjoin those to list(sos). The 160-step proof was obtained in four consecutive stages without deviating from the plan.

Both for finding a more elegant proof or for finding a first proof, especially when either is offering substantial resistance, a good move to make is to take action that shuffles or perturbs the search dramatically. The avoidance of double negation, placing a limit on variable richness, including demodulators to block the retention of one or more deduced items are but a few ways to have the program spend time in a new part of the universe of deducible conclusions. The sharp difference between the two studies of Lukasiewicz cited here can be explained by the program's having spent time in different parts of the search space. This small taste of means for pursuing radically diverse paths is just that, a small taste. If you consider other approaches and then the combinations that can be chosen, you can see why certain results can elude a powerful automated reasoning program for years. However, perhaps in the not-too-distant future, somebody will automate much of what is found in these notebooks. Perhaps, with a single command, a large number of computers, each taking a different approach, will be set to work on some as-yet unsolved problem. That type of parallelism (suggested by McCune) appeals to me in contrast to the more standard use.

A 111-Step Proof from the Lukasiewicz 23-Letter Axiom

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on jaguar.mcs.anl.gov,
Fri Dec 14 11:44:59 2007
The command was "otter".  The process ID is 29723.

-----> EMPTY CLAUSE at 167.50 sec ----> 85173 [hyper,7,85026,438,209]
    $ANS(step_allLuka_1_2_3).

Length of proof is 111.  Level of proof is 39.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
7 [] -P(i(i(p,q),i(i(q,r),i(p,r)))) | -P(i(i(n(p),p),p)) | -P(i(p,i(n(p),q))) | $ANS(step_allLuka_1_2_3).
8 [] P(i(i(i(x,y),i(i(i(n(z),n(u)),v),z)),i(w,i(i(z,x),i(u,x))))).
32 [hyper,1,8,8] P(i(x,i(i(i(i(y,z),i(u,z)),i(z,v)),i(w,i(z,v))))).
33 [hyper,1,32,32] P(i(i(i(i(x,y),i(z,y)),i(y,u)),i(v,i(y,u)))).
35 [hyper,1,8,33] P(i(x,i(i(i(y,z),i(i(u,y),i(v,y))),i(w,i(i(u,y),i(v,y)))))).
36 [hyper,1,33,8] P(i(x,i(y,i(i(y,z),i(u,z))))).
37 [hyper,1,35,35] P(i(i(i(x,y),i(i(z,x),i(u,x))),i(v,i(i(z,x),i(u,x))))).
38 [hyper,1,36,36] P(i(x,i(i(x,y),i(z,y)))).
42 [hyper,1,38,38] P(i(i(i(x,i(i(x,y),i(z,y))),u),i(v,u))).
43 [hyper,1,37,38] P(i(x,i(i(i(y,z),y),i(u,y)))).
53 [hyper,1,43,43] P(i(i(i(x,y),x),i(z,x))).
58 [hyper,1,33,53] P(i(x,i(y,i(z,y)))).
59 [hyper,1,8,53] P(i(x,i(i(y,i(y,z)),i(u,i(y,z))))).
60 [hyper,1,53,37] P(i(x,i(y,i(i(z,y),i(u,y))))).
70 [hyper,1,58,58] P(i(x,i(y,x))).
74 [hyper,1,59,59] P(i(i(x,i(x,y)),i(z,i(x,y)))).
76 [hyper,1,60,60] P(i(x,i(i(y,x),i(z,x)))).
96 [hyper,1,74,74] P(i(x,i(i(y,i(y,z)),i(y,z)))).
103 [hyper,1,74,8] P(i(x,i(i(i(y,z),i(i(i(n(u),n(v)),w),u)),i(i(u,y),i(v,y))))).
105 [hyper,1,8,76] P(i(x,i(i(i(y,i(z,u)),z),i(v,z)))).
```

134 [hyper,1,96,96] P(i(i(x,i(x,y)),i(x,y))).
139 [hyper,1,103,103] P(i(i(i(x,y),i(i(i(n(z),n(u)),v),z)),i(i(z,x),i(u,x)))).
146 [hyper,1,105,105] P(i(i(i(x,i(y,z)),y),i(u,y))).
154 [hyper,1,134,42] P(i(i(i(x,i(i(x,y),i(z,y))),u),u)).
157 [hyper,1,139,139] P(i(i(i(x,y),i(y,z)),i(u,i(y,z)))).
158 [hyper,1,8,139] P(i(x,i(i(i(y,z),i(z,u)),i(v,i(z,u))))).
160 [hyper,1,139,146] P(i(i(x,i(y,i(x,z))),i(u,i(y,i(x,z))))).
165 [hyper,1,146,8] P(i(x,i(i(n(y),n(z)),i(i(y,u),i(z,u))))).
166 [hyper,1,139,157] P(i(i(i(x,y),i(z,x)),i(u,i(z,x)))).
167 [hyper,1,134,157] P(i(i(i(x,y),i(y,z)),i(y,z))).
169 [hyper,1,74,157] P(i(x,i(i(i(y,z),i(z,u)),i(z,u)))).
173 [hyper,1,139,158] P(i(i(i(x,i(n(y),z)),u),i(y,u))).
176 [hyper,1,165,165] P(i(i(n(x),n(y)),i(i(x,z),i(y,z)))).
180 [hyper,1,134,166] P(i(i(i(x,y),i(z,x)),i(z,x))).
191 [hyper,1,139,169] P(i(i(i(n(x),y),z),i(x,z))).
199 [hyper,1,173,173] P(i(x,i(y,i(n(x),z)))).
209 [hyper,1,173,167] P(i(x,i(n(x),y))).
216 [hyper,1,37,176] P(i(x,i(i(y,n(y)),i(z,n(y))))).
229 [hyper,1,191,38] P(i(x,i(i(i(n(x),y),z),i(u,z)))).
289 [hyper,1,38,209] P(i(i(i(x,i(n(x),y)),z),i(u,z))).
334 [hyper,1,216,216] P(i(i(x,n(x)),i(y,n(x)))).
343 [hyper,1,134,289] P(i(i(i(x,i(n(x),y)),z),z)).
356 [hyper,1,191,334] P(i(x,i(y,n(n(x))))).
357 [hyper,1,139,334] P(i(i(n(x),x),i(y,x))).
368 [hyper,1,167,356] P(i(x,n(n(i(y,x))))).
371 [hyper,1,134,356] P(i(x,n(n(x)))).
438 [hyper,1,134,357] P(i(i(n(x),x),x)).
543 [hyper,1,176,371] P(i(i(x,y),i(n(n(x)),y))).
646 [hyper,1,543,368] P(i(n(n(x)),n(n(i(y,x))))).
679 [hyper,1,543,8] P(i(n(n(i(i(x,y),i(i(i(n(z),n(u)),v),z)))),i(w,i(i(z,x),i(u,x))))).
695 [hyper,1,176,646] P(i(i(n(x),y),i(n(i(z,x)),y))).
709 [hyper,1,180,695] P(i(n(i(x,y)),n(y))).
734 [hyper,1,176,709] P(i(i(i(x,y),z),i(y,z))).
763 [hyper,1,734,356] P(i(x,i(y,n(n(i(z,x)))))).
767 [hyper,1,734,70] P(i(x,i(y,i(z,x)))).
769 [hyper,1,734,8] P(i(i(i(i(n(x),n(y)),z),x),i(u,i(i(x,v),i(y,v))))).
938 [hyper,1,734,769] P(i(x,i(y,i(i(x,z),i(u,z))))).
945 [hyper,1,139,769] P(i(i(i(i(x,y),i(z,y)),i(i(n(x),n(z)),u)),i(v,i(i(n(x),n(z)),u)))).
1026 [hyper,1,945,767] P(i(x,i(i(n(y),n(z)),i(u,i(i(y,v),i(z,v)))))).
1027 [hyper,1,945,763] P(i(x,i(i(n(y),n(z)),n(n(i(u,i(i(y,v),i(z,v))))))).
1030 [hyper,1,945,199] P(i(x,i(i(n(y),n(z)),i(n(i(i(y,u),i(z,u))),v)))).
1036 [hyper,1,945,42] P(i(x,i(i(n(y),n(z)),i(z,i(i(y,u),i(v,u)))))).
1038 [hyper,1,1026,1026] P(i(i(n(x),n(y)),i(z,i(i(x,u),i(y,u))))).
1044 [hyper,1,1027,1027] P(i(i(n(x),n(y)),n(n(i(z,i(i(x,u),i(y,u))))))).
1059 [hyper,1,1030,1030] P(i(i(n(x),n(y)),i(n(i(i(x,z),i(y,z))),u))).
1079 [hyper,1,1036,1036] P(i(i(n(x),n(y)),i(y,i(i(x,z),i(u,z))))).
1092 [hyper,1,139,1038] P(i(i(i(i(x,y),i(z,y)),n(x)),i(u,n(x)))).
1097 [hyper,1,734,1044] P(i(n(x),n(n(i(y,i(i(z,u),i(x,u))))))).
1115 [hyper,1,180,1059] P(i(n(i(i(x,y),i(z,y))),n(x))).
1144 [hyper,1,139,1092] P(i(i(n(x),i(i(x,y),i(z,y))),i(u,i(i(x,y),i(z,y))))).
1149 [hyper,1,1079,1097] P(i(n(i(x,i(i(y,z),i(u,z)))),i(i(u,v),i(w,v)))).
1163 [hyper,1,176,1097] P(i(i(x,y),i(n(i(z,i(i(u,v),i(x,v)))),y))).
1181 [hyper,1,176,1115] P(i(i(i(i(x,y),i(z,y)),u),i(x,u))).
1206 [hyper,1,134,1144] P(i(i(n(x),i(i(x,y),i(z,y))),i(i(x,y),i(z,y)))).
1235 [hyper,1,1163,209] P(i(n(i(x,i(i(y,z),i(u,z)))),i(n(u),v))).

1246 [hyper,1,1181,1181] P(i(i(x,y),i(x,i(z,i(u,y))))).
1268 [hyper,1,1181,134] P(i(x,i(i(x,y),y))).
1284 [hyper,1,1206,938] P(i(i(x,i(y,z)),i(i(n(x),z),i(y,z)))).
1285 [hyper,1,1206,679] P(i(i(n(i(i(x,y),i(i(i(n(z),n(u)),v),z))),i(u,x)),i(i(z,x),i(u,x)))).
1404 [hyper,1,1285,1235] P(i(i(i(x,y),z),i(n(x),z))).
1524 [hyper,1,1404,1181] P(i(n(i(i(x,y),i(z,y))),i(x,u))).
1629 [hyper,1,1163,1268] P(i(n(i(x,i(i(y,z),i(u,z)))),i(i(u,v),v))).
2324 [hyper,1,134,160] P(i(i(x,i(y,i(x,z))),i(y,i(x,z)))).
4994 [hyper,1,1285,1149] P(i(i(i(x,y),i(z,u)),i(i(x,u),i(z,u)))).
5337 [hyper,1,1246,438] P(i(i(n(x),x),i(y,i(z,x)))).
5761 [hyper,1,1284,1268] P(i(i(n(x),y),i(i(x,y),y))).
5776 [hyper,1,1284,1163] P(i(i(n(i(x,y)),y),i(n(i(z,i(i(u,v),i(x,v)))),y))).
6120 [hyper,1,1246,1524] P(i(n(i(i(x,y),i(z,y))),i(u,i(v,i(x,w))))).
6436 [hyper,1,1285,1629] P(i(i(i(x,y),z),i(i(x,z),z))).
7549 [hyper,1,343,4994] P(i(i(x,y),i(n(i(x,z)),y))).
7554 [hyper,1,154,4994] P(i(i(x,i(y,z)),i(i(i(x,u),z),i(y,z)))).
9858 [hyper,1,5761,1524] P(i(i(i(i(x,y),i(z,y)),i(x,u)),i(x,u))).
10280 [hyper,1,1206,6120] P(i(i(i(i(x,y),i(z,y)),i(x,u)),i(v,i(x,u)))).
10307 [hyper,1,139,10280] P(i(i(i(x,y),i(i(x,z),i(u,z))),i(v,i(i(x,z),i(u,z))))).
10339 [hyper,1,134,10307] P(i(i(i(x,y),i(i(x,z),i(u,z))),i(i(x,z),i(u,z)))).
12490 [hyper,1,7549,695] P(i(n(i(i(n(x),y),z)),i(n(i(u,x)),y))).
12510 [hyper,1,7549,229] P(i(n(i(x,y)),i(i(i(n(x),z),u),i(v,u)))).
12661 [hyper,1,7554,5337] P(i(i(i(i(n(x),x),y),i(z,x)),i(u,i(z,x)))).
14722 [hyper,1,1181,9858] P(i(i(x,y),i(x,i(z,y)))).
17999 [hyper,1,2324,12510] P(i(i(i(n(x),y),z),i(n(i(x,u)),z))).
18135 [hyper,1,139,12661] P(i(i(i(x,y),i(i(n(y),y),z)),i(u,i(i(n(y),y),z)))).
23059 [hyper,1,6436,17999] P(i(i(i(i(n(x),y),i(n(i(x,z)),u)),i(n(i(x,z)),u))).
23213 [hyper,1,10339,18135] P(i(i(i(x,y),z),i(i(n(y),y),z))).
28971 [hyper,1,23059,5776] P(i(n(i(i(x,y),i(i(z,u),i(x,u)))),y)).
29107 [hyper,1,23213,769] P(i(i(n(x),x),i(y,i(i(x,z),i(u,z))))).
50785 [hyper,1,134,12490] P(i(n(i(i(n(x),y),x)),y)).
63232 [hyper,1,14722,28971] P(i(n(i(i(x,y),i(i(z,u),i(x,u)))),i(v,y))).
71485 [hyper,1,14722,50785] P(i(n(i(i(n(x),y),x)),i(z,y))).
80918 [hyper,1,5761,71485] P(i(i(i(i(n(x),y),x),i(z,y)),i(z,y))).
81040 [hyper,1,139,80918] P(i(i(x,i(i(n(y),x),y)),i(z,i(i(n(y),x),y)))).
81260 [hyper,1,10339,81040] P(i(i(x,y),i(i(n(y),x),y))).
82040 [hyper,1,81260,29107] P(i(i(n(i(x,i(i(y,z),i(u,z)))),i(n(y),y)),i(x,i(i(y,z),i(u,z))))).
85026 [hyper,1,82040,63232] P(i(i(x,y),i(i(y,z),i(x,z)))).

4. Second Test of the Subformula Strategy

For the next test of the subformula strategy, I selected for study Meredith's 21-letter single axiom for classical two-valued sentential calculus, the following.

% Following is Meredith.
P(i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x)))).

My choice was no doubt influenced by history: many years ago, when I studied the Lukasiewicz 23-letter formula, my goal was to demonstrate that my new lemma-adjunction strategy would work on a theorem for which no proof existed. The formulation of that strategy occurred with my study of the just-cited Meredith 21-letter axiom. As I recall, in the beginning, I was intent on devising an approach that would find a proof (even if known) of some interesting theorem *without* using a known proof. So, why not be perverse; indeed, for the subformula strategy, study Lukasiewicz followed by Meredith, which reverses their role when lemma adjunction came into being.

As my test with the Meredith 21-letter axiom commenced, I had no idea where in the long run I would wander, and I had no idea what wonders I would behold. I intended, at this point, to detail this

journey so that, perhaps, you might share in my amazement, in my successes as well as failures. But, because of what eventually occurred, I simply say that the various paths I followed to again illustrate the power of the subformula strategy led essentially nowhere. Indeed, no matter what I tried, I could not enable OTTER to prove a known axiom system by focusing on the Meredith single axiom as sole hypothesis. In short, I was quite disappointed, and I had given up trying to reach the stated goal. But Overbeek saved the day, and here is what he suggested.

Overbeek entered the study as a direct result of my informing him that all of my attempts to deduce an axiom system, starting with the Meredith formula, had failed. He immediately suggested that the problem was, in effect, that the program was probably being given too much latitude. The reasoning must be restricted, he asserted, and the best way to do this was to place a low bound on the complexity of retained clauses. He, on his own, ran a set of runs with max_weight assigned the value 1, 2, 3, and the like. His experiments were conducted with an input file I sent him, one that had in fact failed to reach the sought-after goal; he merely changed the max_weight assignment. Note that the input file had two sets of resonators, one for the ten subformulas of the Meredith single axiom, and one for the members of known axiom systems. Therefore, if measured purely in terms of symbol count, the clauses retained were not restricted to those of complexity equal to or less than the assigned max_weight.

Not too long after he informed me of his plan, he called to tell me his approach had succeeded. In particular, if the max_weight were assigned the value 15, OTTER would find a proof of the type under consideration. Yes, I was delighted and surprised. Clearly, restricting the space of deduced conclusions made all the difference. Of course, I returned to an input file, the following, to verify his success.

An Input File, Using the Subformula Strategy, for Meredith

```
% Trying for an automated proof the Lukasiewicz 23-letter single axiom.
set(hyper_res).
assign(max_weight,15).
% assign(change_limit_after,2000).
% assign(new_max_weight,24).
assign(max_proofs,-1).
clear(print_kept).
% set(process_input).
% set(ancestor_subsume).
set(back_sub).
clear(print_back_sub).
assign(max_distinct_vars,7).
assign(pick_given_ratio,4).
assign(max_mem,880000).
assign(report,5400).
set(order_history).
set(input_sos_first).
assign(heat,0).
% assign(dynamic_heat_weight,0).

weight_list(pick_and_purge).
% Following 10, from Ross, subformulas of the Meredith single axiom for two-valued.
weight(i(u,x),2).
weight(i(v,x),2).
weight(i(x,y),2).
weight(i(n(z),n(u)),2).
weight(i(i(v,x),i(u,x)),2).
weight(i(i(x,y),i(n(z),n(u))),2).
weight(i(i(i(x,y),i(n(z),n(u))),z),2).
weight(i(i(i(i(x,y),i(n(z),n(u))),z),v),2).
```

```
weight(i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x))),2).
weight(P(i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x)))),2).
%  Following are members of known axiom systems for two-valued.
weight(P(i(i(x,y),i(i(y,r),i(x,r)))),1).
weight(P(i(i(n(x),x),x)),1).
weight(P(i(x,i(n(x),y))),1).
weight(P(i(y,i(x,y))),1).
weight(P(i(i(i(x,y),z),i(y,z))),1).
weight(P(i(i(x,i(y,z)),i(y,i(x,z)))),1).
weight(P(i(i(y,z),i(i(x,y),i(x,z)))),1).
weight(P(i(i(x,i(x,y)),i(x,y))),1).
weight(P(i(i(x,i(y,z)),i(i(x,y),i(x,z)))),1).
weight(P(i(i(i(x,y),z),i(n(x),z))),1).
weight(P(i(n(n(x)),x)),1).
weight(P(i(x,n(n(x)))),1).
weight(P(i(i(x,y),i(n(y),n(x)))),1).
weight(P(i(i(n(x),n(y)),i(y,x))),1).
weight(P(i(i(x,y),i(i(n(x),y),y))),1).
weight(P(i(i(n(x),z),i(i(y,z),i(i(x,y),z)))),1).
weight(P(i(i(u,i(n(x),z)),i(u,i(i(y,z),i(i(x,y),z))))),1).
end_of_list.

list(usable).
% condensed detachment
-P(i(x,y)) | -P(x) | P(y).
%  The following disjunctions are known axiom systems.
-P(i(q,i(p,q))) | -P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) | -P(i(n(n(p)),p)) | -P(i(p,n(n(p)))) | -P(i(i(p,q),i(n(q),n(p)))) |
     -P(i(i(p,i(q,r)),i(q,i(p,r)))) | $ANS(step_allFrege_18_35_39_40_46_21). % 21 is dependent.
-P(i(q,i(p,q))) | -P(i(i(p,i(q,r)),i(q,i(p,r)))) | -P(i(i(q,r),i(i(p,q),i(p,r)))) | -P(i(p,i(n(p),q))) |
     -P(i(i(p,q),i(i(n(p),q),q))) | -P(i(i(p,i(p,q)),i(p,q))) |
     $ANS(step_allHilbert_18_21_22_3_54_30). % 30 is dependent.
-P(i(q,i(p,q))) | -P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) | -P(i(i(n(p),n(q)),i(q,p))) |
     $ANS(step_allBEH_Church_FL_18_35_49).
-P(i(i(i(p,q),r),i(q,r))) | -P(i(i(i(p,q),r),i(n(p),r))) | -P(i(i(n(p),r),i(i(q,r),i(i(p,q),r)))) |
     $ANS(step_allLuka_x_19_37_59).
-P(i(i(i(p,q),r),i(q,r))) | -P(i(i(i(p,q),r),i(n(p),r))) | -P(i(i(s,i(n(p),r)),i(s,i(i(q,r),i(i(p,q),r))))) |
     $ANS(step_allWos_x_19_37_60).
-P(i(i(p,q),i(i(q,r),i(p,r)))) | -P(i(i(n(p),p),p)) | -P(i(p,i(n(p),q))) | $ANS(step_allLuka_1_2_3).
end_of_list.

list(sos).
%  Following is Lukasiewicz's 23-letter single axiom.
P(i(i(i(x,y),i(i(i(n(z),n(u)),v),z)),i(w,i(i(z,x),i(u,x))))).
end_of_list.

list(passive).
%  Following 10 negations of subformulas of Meredith.
-P(i(a4,a1)) | $ANS(SUB).
-P(i(a5,a1)) | $ANS(SUB).
-P(i(a1,a2)) | $ANS(SUB).
-P(i(n(a3),n(a4))) | $ANS(SUB).
-P(i(i(a5,a1),i(a4,a1))) | $ANS(SUB).
-P(i(i(a1,a2),i(n(a3),n(a4)))) | $ANS(SUB).
-P(i(i(i(a1,a2),i(n(a3),n(a4))),a3)) | $ANS(SUB).
-P(i(i(i(i(a1,a2),i(n(a3),n(a4))),a3),a5)) | $ANS(SUB).
```

```
-P(i(i(i(i(i(a1,a2),i(n(a3),n(a4))),a3),a5),i(i(a5,a1),i(a4,a1)))) | $ANS(SUB).
-P(i(i(i(i(i(a1,a2),i(n(a3),n(a4))),a3),a5),i(i(a5,a1),i(a4,a1)))) | $ANS(SUB).
%  Following are members of known axiom systems for two-valued.
-P(i(i(p,q),i(i(q,r),i(p,r)))) | $ANS(step_L1).
-P(i(i(n(p),p),p)) | $ANS(step_L2).
-P(i(p,i(n(p),q))) | $ANS(step_L3).
-P(i(q,i(p,q))) | $ANS(step_18).
-P(i(i(i(p,q),r),i(q,r))) | $ANS(step_19).
-P(i(i(p,i(q,r)),i(q,i(p,r)))) | $ANS(step_21).
-P(i(i(q,r),i(i(p,q),i(p,r)))) | $ANS(step_22).
-P(i(i(p,i(p,q)),i(p,q))) | $ANS(step_30).
-P(i(i(p,i(q,r)),i(i(p,q),i(p,r)))) | $ANS(step_35).
-P(i(i(i(p,q),r),i(n(p),r))) | $ANS(step_37).
-P(i(n(n(p)),p)) | $ANS(step_39).
-P(i(p,n(n(p)))) | $ANS(step_40).
-P(i(i(p,q),i(n(q),n(p)))) | $ANS(step_46).
-P(i(i(n(p),n(q)),i(q,p))) | $ANS(step_49).
-P(i(i(p,q),i(i(n(p),q),q))) | $ANS(step_54).
-P(i(i(n(p),r),i(i(q,r),i(i(p,q),r)))) | $ANS(step_59).
-P(i(i(s,i(n(p),r)),i(s,i(i(q,r),i(i(p,q),r))))) | $ANS(step_60).
end_of_list.

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

list(hot).
-P(i(x,y)) | -P(x) | P(y).
%  Following is Meredith.
P(i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x)))).
end_of_list.
```

And, of course, Overbeek had triumphed, succeeded in showing that, indeed, the subformula strategy was quite powerful. The proof, given shortly, OTTER returned has length 188 and level 43, requires the use of variable richness seven, and includes formulas relying on double negation. In contrast, the proof I obtained years ago (with lemma adjunction) has length 160, level 74, relies on variable richness seven, and avoids double negation (terms of the form $n(n(t))$ for some term t).

A Proof for Meredith Obtained with the Subformula Strategy

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on jaguar.mcs.anl.gov,
Sat Jan  5 15:35:54 2008
The command was "otter".  The process ID is 24003.

-----> EMPTY CLAUSE at 3092.25 sec ----> 149466 [hyper,7,149275,25717,419]
```

$ANS(step_allLuka_1_2_3).

Length of proof is 188. Level of proof is 43.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
7 [] -P(i(i(p,q),i(i(q,r),i(p,r)))) | -P(i(i(n(p),p),p)) | -P(i(p,i(n(p),q))) | $ANS(step_allLuka_1_2_3).
8 [] P(i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x)))).
43 [hyper,1,8,8] P(i(i(i(i(x,y),i(z,y)),i(y,u)),i(v,i(y,u)))).
45 [hyper,1,8,43] P(i(i(i(x,i(n(y),z)),u),i(y,u))).
51 [hyper,1,45,45] P(i(x,i(y,i(n(x),z)))).
52 [hyper,1,8,45] P(i(i(i(x,x),y),i(z,y))).
55 [hyper,1,52,52] P(i(x,i(y,i(z,z)))).
57 [hyper,1,43,52] P(i(x,i(y,i(z,y)))).
58 [hyper,1,8,52] P(i(i(i(x,y),n(y)),i(z,n(y)))).
61 [hyper,1,55,55] P(i(x,i(y,y))).
62 [hyper,1,8,55] P(i(i(i(x,i(y,y)),z),i(u,z))).
63 [hyper,1,61,61] P(i(x,x)).
64 [hyper,1,8,63] P(i(i(i(i(i(x,y),i(n(z),n(u))),z),x),i(u,x))).
68 [hyper,1,8,64] P(i(i(i(x,y),i(y,z)),i(u,i(y,z)))).
69 [hyper,1,57,57] P(i(x,i(y,x))).
80 [hyper,1,69,8] P(i(x,i(i(i(i(i(y,z),i(n(u),n(v))),u),w),i(i(w,y),i(v,y))))).
82 [hyper,1,8,80] P(i(i(i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x))),w),i(v6,w))).
117 [hyper,1,68,51] P(i(x,i(y,i(n(i(z,y)),u)))).
139 [hyper,1,8,62] P(i(i(i(x,y),z),i(y,z))).
147 [hyper,1,8,139] P(i(i(i(i(n(x),n(y)),x),z),i(y,z))).
149 [hyper,1,139,69] P(i(x,i(y,i(z,x)))).
152 [hyper,1,139,8] P(i(x,i(i(x,y),i(z,y)))).
153 [hyper,1,152,152] P(i(i(i(x,i(i(x,y),i(z,y))),u),i(v,u))).
154 [hyper,1,139,152] P(i(x,i(i(i(y,x),z),i(u,z)))).
158 [hyper,1,8,152] P(i(i(i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(w,v)),x),i(u,x))).
159 [hyper,1,152,139] P(i(i(i(i(i(x,y),z),i(y,z)),u),i(v,u))).
207 [hyper,1,152,149] P(i(i(i(x,i(y,i(z,x))),u),i(v,u))).
210 [hyper,1,139,149] P(i(x,i(y,i(z,i(u,x))))).
290 [hyper,1,139,210] P(i(x,i(y,i(z,i(u,i(v,x)))))).
348 [hyper,1,117,117] P(i(x,i(n(i(y,x)),z))).
354 [hyper,1,139,348] P(i(x,i(n(i(y,i(z,x))),u))).
408 [hyper,1,152,147] P(i(i(i(i(i(i(n(x),n(y)),x),z),i(y,z)),u),i(v,u))).
413 [hyper,1,8,147] P(i(i(i(x,y),n(i(n(y),n(z)))),i(z,n(i(n(y),n(z)))))).
414 [hyper,1,147,348] P(i(x,i(n(i(y,i(i(n(z),n(x)),z))),u))).
419 [hyper,1,147,139] P(i(x,i(n(x),y))).
423 [hyper,1,147,58] P(i(x,i(y,n(n(x))))).
437 [hyper,1,45,419] P(i(x,i(n(i(y,i(n(x),z))),u))).
471 [hyper,1,139,423] P(i(x,i(y,n(n(i(z,x)))))).
475 [hyper,1,68,423] P(i(x,i(y,n(n(i(z,y)))))).
586 [hyper,1,152,471] P(i(i(i(x,i(y,n(n(i(z,x))))),u),i(v,u))).
612 [hyper,1,475,475] P(i(x,n(n(i(y,x))))).
618 [hyper,1,152,475] P(i(i(i(x,i(y,n(n(i(z,y))))),u),i(v,u))).
623 [hyper,1,8,475] P(i(i(i(x,n(n(i(y,x)))),z),i(u,z))).
845 [hyper,1,8,154] P(i(i(i(i(i(x,i(i(i(y,z),i(n(u),n(v))),u)),w),i(v6,w)),y),i(v,y))).
865 [hyper,1,139,354] P(i(x,i(n(i(y,i(z,i(u,x)))),v))).
964 [hyper,1,8,158] P(i(i(i(x,y),i(i(i(y,z),i(n(u),n(x))),u)),i(v,i(i(i(y,z),i(n(u),n(x))),u)))).
1123 [hyper,1,964,52] P(i(x,i(i(i(y,z),i(n(y),n(i(u,u)))),y))).
1145 [hyper,1,8,159] P(i(i(i(x,y),i(z,n(y))),i(u,i(z,n(y))))).

1175 [hyper,1,139,437] P(i(x,i(n(i(y,i(n(i(z,x)),u))),v))).
1656 [hyper,1,8,618] P(i(i(i(x,y),z),i(n(i(u,n(y))),z))).
1733 [hyper,1,64,408] P(i(x,i(i(n(y),n(n(y))),y))).
1734 [hyper,1,1733,1733] P(i(i(n(x),n(n(x))),x)).
1745 [hyper,1,139,1734] P(i(n(n(x)),x)).
1782 [hyper,1,154,1745] P(i(i(i(x,i(n(n(y)),y)),z),i(u,z))).
1831 [hyper,1,8,1782] P(i(i(i(x,n(n(y))),z),i(y,z))).
1862 [hyper,1,1831,69] P(i(x,i(y,i(z,n(n(x)))))).
1937 [hyper,1,8,1862] P(i(i(i(x,i(y,n(n(i(i(i(z,u),i(n(v),n(w))),v))))),z),i(w,z))).
2054 [hyper,1,1123,1123] P(i(i(i(x,y),i(n(x),n(i(z,z)))),x)).
2056 [hyper,1,8,2054] P(i(i(i(x,y),x),i(z,x))).
2078 [hyper,1,2056,1831] P(i(x,i(y,n(n(y))))).
2079 [hyper,1,2056,1656] P(i(x,i(n(i(y,n(z))),z))).
2096 [hyper,1,152,2078] P(i(i(i(x,i(y,n(n(y)))),z),i(u,z))).
2103 [hyper,1,8,2078] P(i(i(i(x,n(n(x))),y),i(z,y))).
2414 [hyper,1,152,2079] P(i(i(i(x,i(n(i(y,n(z))),z)),u),i(v,u))).
2927 [hyper,1,8,845] P(i(i(i(x,y),i(z,i(i(i(y,u),i(n(v),n(x))),v))),i(w,i(z,i(i(i(y,u),i(n(v),n(x))),v))))).
2986 [hyper,1,8,1937] P(i(i(i(x,y),z),i(n(i(i(i(y,u),i(n(v),n(x))),v)),z))).
3022 [hyper,1,2986,612] P(i(n(i(i(i(x,y),i(n(z),n(u))),z)),n(n(i(v,i(u,x)))))).
3078 [hyper,1,1734,3022] P(i(i(i(x,y),i(n(i(z,x)),n(z))),i(z,x))).
3115 [hyper,1,139,3078] P(i(i(n(i(x,y)),n(x)),i(x,y))).
3824 [hyper,1,8,2414] P(i(i(i(x,i(y,n(n(z)))),u),i(z,u))).
3827 [hyper,1,8,3824] P(i(i(i(x,y),z),i(n(x),z))).
3952 [hyper,1,3827,413] P(i(n(i(x,y)),i(z,n(i(n(y),n(z)))))).
4487 [hyper,1,2927,152] P(i(x,i(i(i(y,z),u),i(i(i(z,v),i(n(u),n(y))),u)))).
4536 [hyper,1,4487,4487] P(i(i(i(x,y),z),i(i(i(y,u),i(n(z),n(x))),z))).
4572 [hyper,1,4536,8] P(i(i(i(x,y),i(n(i(i(x,z),i(u,z))),n(i(i(i(z,v),i(n(w),n(u))),w)))),i(i(x,z),i(u,z)))).
4743 [hyper,1,8,4572] P(i(i(i(i(i(x,y),z),i(u,z)),x),i(v,x))).
4747 [hyper,1,4572,1175] P(i(i(x,y),i(n(i(z,i(x,u))),y))).
4749 [hyper,1,4572,865] P(i(i(x,i(y,i(x,z))),i(u,i(y,i(x,z))))).
4750 [hyper,1,4572,437] P(i(i(x,y),i(n(i(x,z)),y))).
4751 [hyper,1,4572,414] P(i(i(x,y),i(i(n(y),n(i(x,z))),y))).
4752 [hyper,1,4572,354] P(i(i(x,i(x,y)),i(z,i(x,y)))).
4754 [hyper,1,4752,4752] P(i(x,i(i(y,i(y,z)),i(y,z)))).
4828 [hyper,1,4752,2056] P(i(x,i(i(i(y,z),y),y))).
4911 [hyper,1,4752,82] P(i(x,i(i(i(i(i(i(y,z),i(n(u),n(v))),u),w),i(i(w,y),i(v,y))),v6),v6))).
4913 [hyper,1,4752,68] P(i(x,i(i(i(y,z),i(z,u)),i(z,u)))).
4916 [hyper,1,4752,52] P(i(x,i(i(i(y,y),z),z))).
4964 [hyper,1,4913,4913] P(i(i(i(x,y),i(y,z)),i(y,z))).
4992 [hyper,1,8,4913] P(i(i(i(i(i(x,y),i(y,z)),i(y,z)),u),i(v,u))).
5467 [hyper,1,4964,4828] P(i(i(i(x,y),x),x)).
5484 [hyper,1,8,4828] P(i(i(i(i(i(x,y),x),x),z),i(u,z))).
5517 [hyper,1,5467,4911] P(i(i(i(i(i(i(i(x,y),i(n(z),n(u))),z),v),i(i(v,x),i(u,x))),w),w)).
5533 [hyper,1,5517,4916] P(i(i(i(x,x),y),y)).
5694 [hyper,1,5533,4754] P(i(i(x,i(x,y)),i(x,y))).
5706 [hyper,1,8,4754] P(i(i(i(i(x,i(x,y)),i(x,y)),z),i(u,z))).
5748 [hyper,1,5694,3952] P(i(n(i(x,y)),n(i(n(y),n(n(i(x,y)))))).
5788 [hyper,1,5694,2103] P(i(i(i(x,n(n(x))),y),y)).
5789 [hyper,1,5694,2096] P(i(i(i(x,i(y,n(n(y)))),z),z)).
5843 [hyper,1,5694,1145] P(i(i(i(x,y),i(z,n(y))),i(z,n(y)))).
5875 [hyper,1,5694,623] P(i(i(i(x,n(n(i(y,x)))),z),z)).
5876 [hyper,1,5694,618] P(i(i(i(x,i(y,n(n(i(z,y))))),u),u)).
5877 [hyper,1,5694,586] P(i(i(i(x,i(y,n(n(i(z,x))))),u),u)).
5906 [hyper,1,5694,207] P(i(i(i(x,i(y,i(z,x))),u),u)).
5921 [hyper,1,5694,153] P(i(i(i(x,i(i(x,y),i(z,y))),u),u)).

5935 [hyper,1,5694,62] P(i(i(i(x,i(y,y)),z),z)).
6020 [hyper,1,152,5921] P(i(i(i(i(i(x,i(i(x,y),i(z,y))),u),u),v),i(w,v))).
6052 [hyper,1,152,5935] P(i(i(i(i(i(x,i(y,y)),z),z),u),i(v,u))).
6565 [hyper,1,4750,4747] P(i(n(i(i(x,y),z)),i(n(i(u,i(x,v))),y))).
6610 [hyper,1,4747,5935] P(i(n(i(x,i(i(i(y,i(z,z)),u),v))),u)).
6787 [hyper,1,5694,4992] P(i(i(i(i(i(x,y),i(y,z)),i(y,z)),u),u)).
6810 [hyper,1,8,6787] P(i(i(x,i(y,n(x))),i(z,i(y,n(x))))).
7228 [hyper,1,5921,4751] P(i(i(n(i(i(x,y),i(z,y))),n(i(x,u))),i(i(x,y),i(z,y)))).
7425 [hyper,1,5694,5484] P(i(i(i(i(i(x,y),x),x),z),z)).
7439 [hyper,1,8,7425] P(i(i(x,i(i(n(x),n(y)),z)),i(y,i(i(n(x),n(y)),z)))).
7508 [hyper,1,8,5789] P(i(i(x,y),i(n(n(x)),y))).
9868 [hyper,1,5694,4743] P(i(i(i(i(i(x,y),z),i(u,z)),x),x)).
9894 [hyper,1,9868,408] P(i(i(n(i(x,i(y,z))),n(y)),i(x,i(y,z)))).
10204 [hyper,1,5694,5706] P(i(i(i(i(x,i(x,y)),i(x,y)),z),z)).
10240 [hyper,1,7228,5748] P(i(i(n(i(x,y)),y),i(x,y))).
10605 [hyper,1,8,5876] P(i(i(x,y),i(n(i(z,n(x))),y))).
10808 [hyper,1,10605,4750] P(i(n(i(x,n(i(y,z)))),i(n(i(y,u)),z))).
13172 [hyper,1,5694,10808] P(i(n(i(x,n(i(x,y)))),y)).
15612 [hyper,1,5694,4749] P(i(i(x,i(y,i(x,z))),i(y,i(x,z)))).
16883 [hyper,1,5694,6020] P(i(i(i(i(i(x,i(i(x,y),i(z,y))),u),u),v),v)).
16891 [hyper,1,5694,6052] P(i(i(i(i(i(x,i(y,y)),z),z),u),u)).
16946 [hyper,1,5694,6565] P(i(n(i(i(x,y),i(x,z))),y)).
17207 [hyper,1,154,16946] P(i(i(i(x,i(n(i(i(y,z),i(y,u))),z)),v),i(w,v))).
17208 [hyper,1,152,16946] P(i(i(i(i(n(i(i(x,y),i(x,z))),y),u),i(v,u))).
17416 [hyper,1,5694,17208] P(i(i(i(n(i(i(x,y),i(x,z))),y),u),u)).
17949 [hyper,1,5694,17207] P(i(i(i(x,i(n(i(i(y,z),i(y,u))),z)),v),v)).
17957 [hyper,1,8,17949] P(i(i(i(i(x,n(y)),i(x,z)),u),i(y,u))).
18216 [hyper,1,17957,16891] P(i(x,i(i(i(y,i(z,z)),n(x)),u))).
18352 [hyper,1,5843,18216] P(i(i(i(i(x,i(y,y)),n(i(z,u))),n(u))).
18485 [hyper,1,8,18352] P(i(i(n(x),y),i(n(i(z,x)),y))).
18633 [hyper,1,18485,13172] P(i(n(i(x,i(y,n(i(y,z))))),z)).
18973 [hyper,1,3115,18633] P(i(x,i(y,n(i(y,n(x)))))).
19032 [hyper,1,3827,18973] P(i(n(x),i(y,n(i(y,n(i(x,z))))))).
19700 [hyper,1,5694,19032] P(i(n(x),n(i(n(x),n(i(x,y)))))).
20457 [hyper,1,7228,6610] P(i(i(x,y),i(i(i(z,i(u,u)),n(i(x,v))),y))).
20694 [hyper,1,5694,6810] P(i(i(x,i(y,n(x))),i(y,n(x)))).
21542 [hyper,1,5906,7439] P(i(x,i(i(n(y),n(x)),i(z,y)))).
21547 [hyper,1,5877,7439] P(i(x,i(i(n(y),n(x)),n(n(i(z,y)))))).
21695 [hyper,1,7439,290] P(i(x,i(i(n(y),n(x)),i(z,i(u,i(v,y)))))).
21722 [hyper,1,15612,21542] P(i(i(n(x),n(y)),i(y,x))).
22820 [hyper,1,4750,21547] P(i(n(i(x,y)),i(i(n(z),n(x)),n(n(i(u,z)))))).
24206 [hyper,1,15612,21695] P(i(i(n(x),n(y)),i(y,i(z,i(u,x))))).
25228 [hyper,1,20694,22820] P(i(i(n(x),n(y)),n(n(i(y,x))))).
25259 [hyper,1,7508,25228] P(i(n(n(i(n(x),n(y)))),n(n(i(y,x))))).
25358 [hyper,1,21722,25259] P(i(n(i(x,y)),n(i(n(y),n(x))))).
25386 [hyper,1,18485,25358] P(i(n(i(x,i(y,z))),n(i(n(z),n(y))))).
25671 [hyper,1,18485,25386] P(i(n(i(x,i(y,i(z,u)))),n(i(n(u),n(z))))).
25675 [hyper,1,7228,25386] P(i(i(n(x),x),i(y,x))).
25717 [hyper,1,5694,25675] P(i(i(n(x),x),x)).
26465 [hyper,1,7228,25671] P(i(i(n(x),i(y,x)),i(z,i(y,x)))).
26474 [hyper,1,5694,26465] P(i(i(n(x),i(y,x)),i(y,x))).
37039 [hyper,1,5921,20457] P(i(i(i(x,i(y,y)),n(i(z,u))),i(i(z,v),i(w,v)))).
37052 [hyper,1,8,37039] P(i(i(i(i(x,y),i(z,y)),u),i(n(i(x,v)),u))).
37066 [hyper,1,16883,37052] P(i(n(i(x,y)),i(z,i(i(x,u),i(v,u))))).
37076 [hyper,1,10204,37052] P(i(n(i(x,y)),i(i(x,z),z))).

37095 [hyper,1,5875,37052] P(i(n(i(x,y)),n(n(i(z,i(i(x,u),i(v,u)))))).
37101 [hyper,1,5788,37052] P(i(n(i(x,y)),n(n(i(i(x,z),i(u,z)))))).
37218 [hyper,1,37052,17416] P(i(n(i(n(i(i(x,y),i(x,z))),u)),i(v,y))).
38701 [hyper,1,10240,37066] P(i(x,i(y,i(i(x,z),i(u,z))))).
38826 [hyper,1,7439,38701] P(i(x,i(i(n(y),n(x)),i(i(y,z),i(u,z))))).
38920 [hyper,1,26474,38826] P(i(i(n(x),n(n(i(i(x,y),i(z,y))))),i(i(x,y),i(z,y)))).
39106 [hyper,1,10240,37076] P(i(x,i(i(x,y),y))).
39307 [hyper,1,7508,39106] P(i(n(n(x)),i(i(x,y),y))).
39375 [hyper,1,39106,38701] P(i(i(i(x,i(y,i(i(x,z),i(u,z)))),v),v)).
40409 [hyper,1,9894,37101] P(i(x,i(n(i(i(x,y),i(z,y))),u))).
40771 [hyper,1,20694,40409] P(i(n(i(i(x,y),i(z,y))),n(x))).
40912 [hyper,1,24206,40771] P(i(x,i(y,i(z,i(i(x,u),i(v,u)))))).
42285 [hyper,1,7439,40912] P(i(x,i(i(n(y),n(x)),i(z,i(i(y,u),i(v,u)))))).
42517 [hyper,1,10240,37218] P(i(n(i(i(x,y),i(x,z))),i(u,y))).
43721 [hyper,1,15612,42285] P(i(i(n(x),n(y)),i(y,i(i(x,z),i(u,z))))).
43845 [hyper,1,43721,40771] P(i(x,i(i(i(i(x,y),i(z,y)),u),i(v,u)))).
43938 [hyper,1,43721,19700] P(i(i(n(x),n(i(x,y))),i(i(x,z),i(u,z)))).
44182 [hyper,1,15612,43845] P(i(i(i(i(x,y),i(z,y)),u),i(x,u))).
53223 [hyper,1,38920,37095] P(i(i(i(x,y),i(z,u)),i(i(x,u),i(z,u)))).
53570 [hyper,1,53223,43938] P(i(i(n(x),i(y,z)),i(i(x,z),i(y,z)))).
53576 [hyper,1,53223,39106] P(i(i(x,y),i(i(i(x,z),y),y))).
55077 [hyper,1,53570,42517] P(i(i(i(i(x,y),i(x,z)),y),i(u,y))).
55088 [hyper,1,53570,39307] P(i(i(n(x),y),i(i(x,y),y))).
57246 [hyper,1,5694,55077] P(i(i(i(i(x,y),i(x,z)),y),y)).
60255 [hyper,1,5921,53576] P(i(i(i(x,y),i(i(x,z),i(u,z))),i(i(x,z),i(u,z)))).
60346 [hyper,1,53576,57246] P(i(i(i(i(i(i(x,y),i(x,z)),y),u),y),y)).
79292 [hyper,1,8,60346] P(i(i(x,i(i(y,x),i(y,z))),i(u,i(i(y,x),i(y,z))))).
79295 [hyper,1,60255,79292] P(i(i(x,i(y,z)),i(i(y,x),i(y,z)))).
79688 [hyper,1,39375,79295] P(i(i(x,y),i(x,i(i(y,z),i(u,z))))).
86734 [hyper,1,17416,79688] P(i(n(i(i(x,y),i(x,z))),i(i(y,u),i(v,u)))).
88105 [hyper,1,55088,86734] P(i(i(i(i(x,y),i(x,z)),i(i(y,u),i(v,u))),i(i(y,u),i(v,u)))).
149275 [hyper,1,44182,88105] P(i(i(x,y),i(i(y,z),i(x,z)))).

Because of this occurrence, I immediately returned to the study of the Lukasiewicz 23-letter single axiom. The experiment called for was that of assigning the value of 15 to max_weight to see if OTTER would eventually complete a proof of a known axiom system. Resonators (weight templates) were placed in the input file corresponding to eight subformulas of the Lukasiewicz axiom, along with seventeen corresponding to members of known axiom systems; no additional templates were included. In contrast to the Meredith study, I blocked (with demodulation) the use of double negation. With the retention of clause (762308), a proof deducing the well-known 3-axiom system of Lukasiewicz was completed; more than 25,000 CPU-seconds were required. The proof has length 573 and level 106, and it relies on variable richness seven. If you would find it interesting to run the corresponding experiment, you make three changes to the input file given for the Meredith study. First, you replace the ten weight templates corresponding to the subformulas for Meredith with the following eight for Lukasiewicz.

```
%  Following 8 are subformulas of Luka-23letter and the formula itself.
weight(i(n(z),n(u)),2).
weight(i(i(z,x),i(u,x)),2).
weight(i(i(n(z),n(u)),v),2).
weight(i(w,i(i(z,x),i(u,x))),2).
weight(i(i(i(n(z),n(u)),v),z),2).
weight(i(i(x,y),i(i(i(n(z),n(u)),v),z)),2).
weight(i(i(i(x,y),i(i(i(n(z),n(u)),v),z)),i(w,i(i(z,x),i(u,x)))),2).
weight(P(i(i(i(x,y),i(i(i(n(z),n(u)),v),z)),i(w,i(i(z,x),i(u,x))))),2).
```

Second, you replace the Meredith formula in the list(sos) (and its commented item) by the following.

```
% Following is Lukasiewicz's 23-letter single axiom.
P(i(i(i(x,y),i(i(i(n(z),n(u)),v),z)),i(w,i(i(z,x),i(u,x))))).
```

Third, you replace in list(demodulators) two items regarding negation with the following.

```
% (n(n(n(x))) = junk).
(n(n(x)) = junk).
```

A review shows you that the iterative approach for the Lukasiewicz 23-letter formula given earlier in this notebook can be replaced with a single run, thanks to Overbeek. The experiment just described gives more evidence for the value of the subformula strategy. But the best evidence is reserved for the next section.

5. A Coup de Grace

The focus finally in this notebook is on a fine illustration of the power of the subformula strategy, namely, the following single axiom for the implicational fragment of *R*, an axiom supplied by applying the methods of Rezus.

```
% Following is a Rezus-style single axiom for RI.
P(i(i(i(x,i(i(i(y,y),i(i(z,z),i(i(u,u),i(i(v,v),i(x,w))))),w)),i(i(i(i(i(v6,v7),i(i(v7,v8),i(v6,v8))),
  i(i(i(i(i(v9,i(v9,v10)),i(v9,v10)),i(i(i(v11,v12),i(i(v12,v13),i(v11,v13))),v14)),v14),v15)),v15),
  (i(v16,i(i(i(v17,v17),i(i(v18,v18),i(i(v19,v19),i(i(v20,v20),i(v16,v21))))),v21)),v22))),v22)).
  % Rezus-style single axiom for RI
```

Precisely stated, the goal is to have OTTER rely solely on condensed detachment to eventually find a proof that the given Rezus-style formula is a single axiom for *RI*, the implicational fragment of *R*. The axioms I focused on for *RI* are the following.

```
P(i(i(u,v),i(i(w,u),i(w,v)))). % B
P(i(i(u,i(v,w)),i(v,i(u,w)))). % C
P(i(u,u)). % I
P(i(i(u,i(u,v)),i(u,v))). % W
```

(The formula *B* can be replaced by the formula *B'* to obtain an alternate axiom system for *RI*.) Two studies are warranted. In the first, the object is to rely on the Rezus formula as the only hypothesis and produce a proof deriving the given four axioms. In the second, which is that most pertinent to the discussion of the subformula strategy, the object is to prove that the Rezus formula is in fact a theorem of *RI*, is derivable from *B*, *C*, *I*, and *W*. For added clarity, because the Rezus formula has length in symbol count (with the predicate symbol) of 94, the second task is ordinarily most formidable.

In that the first of the two objectives seems easy to reach (a proof will be included later, at the end of Section 5), the second is that meriting rather intense discussion. Intuition might suggest that some use of resonators, hints, or both might provide a reasonable beginning for guiding the program toward the desired proof. But, so it appears, no obvious choice in either class comes immediately to mind. Further, and here is the crux of the problem, how can one enable the program to consider very complex expressions on the way to proving the Rezus formula a theorem? One way to do this, of course, is to use weight templates to instruct OTTER to treat various complex expressions, measured in symbol count, as being simple, having small weight. If the appropriate templates could be discovered, then, perhaps, the program could so-to-speak work its way up to focusing on long, long formulas, eventually deducing the Rezus formula. Indeed, perhaps the interaction of the templates could consider the Rezus formula as having (in effect) 25 or fewer symbols. If all goes as planned (in this world of fantasy), the max_weight could be assigned the value 25, enabling the program to avoid getting lost among tall trees (complex formulas) and, more important, to follow a path terminating with the given Rezus formula. The obstacle, great as it is, of coping with long, long formulas would be overcome.

To attempt to reach the second and more interesting objective, the choice was, therefore, to focus on nontrivial subformulas of the Rezus formula, subformulas in which the function *i* occurs. However, conjecturing that additional guidance would be necessary, the choice was to so-to-speak parallel the inclusion of those earlier-used targets, namely members of known axiom systems. In the context of the first class, twenty subformulas (supplied to me by my colleague M. Beeson) were included, each

assigned a weight of 1. As for the second class, I extracted from an e-mail from Ulrich so-called building blocks, four of them counting the Rezus formula, and included weight templates, each assigned a weight of 1. Because I anticipated the possible need of using lemma adjunction, in the passive list I included negations of the twenty subformulas, negations of the building blocks, negations of a few related items, and the negation of the Rezus formula. The idea is that, if and when any element of the first two classes is proved, I could, and probably would, use the proof steps (if necessary) to complete a proof of Rezus. Of course, some or most of the subformulas, if viewed as possible theorems, might in fact not be theorems of *RI*. The following input file is the one I used first.

An Input File for Seeking a Proof of the Rezus Formula

```
set(hyper_res).
assign(max_mem,880000).
%  assign(max_seconds,2).
%  set(sos_queue).
assign(max_weight,24).
%  assign(change_limit_after,1500).
%  assign(new_max_weight,32).
assign(max_proofs,-1).
assign(max_distinct_vars,24).
assign(pick_given_ratio,2).
assign(bsub_hint_wt,1).
clear(keep_hint_subsumers).
set(keep_hint_equivalents).
%  set(ancestor_subsume).
set(back_sub).
set(order_history).
%  set(process_input).
clear(print_kept).

weight_list(pick_and_purge).
%  Following 20 are subformulas of the given Rezus formula for RI, from Beeson.
weight(i(i(v6,v7),i(i(v7,v8),i(v6,v8))),1).
weight(i(i(z,z),i(i(u,u),i(i(v,v),i(x,w)))),1).
weight(i(i(v11,v12),i(i(v12,v13),i(v11,v13))),1).
weight(i(i(v19,v19),i(i(v20,v20),i(v16,v21))),1).
weight(i(i(i(v11,v12),i(i(v12,v13),i(v11,v13))),v14),1).
weight(i(i(y,y),i(i(z,z),i(i(u,u),i(i(v,v),i(x,w))))),1).
weight(i(i(i(y,y),i(i(z,z),i(i(u,u),i(i(v,v),i(x,w))))),w),1).
weight(i(i(v18,v18),i(i(v19,v19),i(i(v20,v20),i(v16,v21)))),1).
weight(i(x,i(i(i(y,y),i(i(z,z),i(i(u,u),i(i(v,v),i(x,w))))),w)),1).
weight(i(i(v17,v17),i(i(v18,v18),i(i(v19,v19),i(i(v20,v20),i(v16,v21))))),1).
weight(i(i(i(v17,v17),i(i(v18,v18),i(i(v19,v19),i(i(v20,v20),i(v16,v21))))),v21),1).
weight(i(i(i(v9,i(v9,v10)),i(v9,v10)),i(i(i(v11,v12),i(i(v12,v13),i(v11,v13))),v14)),1).
weight(i(v16,i(i(i(v17,v17),i(i(v18,v18),i(i(v19,v19),i(i(v20,v20),i(v16,v21))))),v21)),1).
weight(i(i(i(i(v9,i(v9,v10)),i(v9,v10)),i(i(i(v11,v12),i(i(v12,v13),i(v11,v13))),v14)),v14),1).
weight(i(i(v16,i(i(i(v17,v17),i(i(v18,v18),i(i(v19,v19),i(i(v20,v20),i(v16,v21))))),v21)),v22),1).
weight(i(i(i(i(i(v9,i(v9,v10)),i(v9,v10)),i(i(i(v11,v12),i(i(v12,v13),i(v11,v13))),v14)),v14),v15),1).
weight(i(i(i(v6,v7),i(i(v7,v8),i(v6,v8))),i(i(i(i(i(v9,i(v9,v10)),i(v9,v10)),i(i(i(v11,v12),i(i(v12,v13),
    i(v11,v13))),v14)),v14),v15)),1).
weight(i(i(i(i(v6,v7),i(i(v7,v8),i(v6,v8))),i(i(i(i(i(v9,i(v9,v10)),i(v9,v10)),i(i(i(v11,v12),i(i(v12,v13),
    i(v11,v13))),v14)),v14),v15)),v15),1).
weight(i(i(i(i(i(v6,v7),i(i(v7,v8),i(v6,v8))),i(i(i(i(i(v9,i(v9,v10)),i(v9,v10)),i(i(i(v11,v12),i(i(v12,v13),
    i(v11,v13))),v14)),v14),v15)),v15),i(i(v16,i(i(i(v17,v17),i(i(v18,v18),i(i(v19,v19),i(i(v20,v20),
    i(v16,v21))))),v21)),v22)),1).
```

```
weight(i(i(x,i(i(i(y,y),i(i(z,z),i(i(u,u),i(i(v,v),i(x,w))))),w)),i(i(i(i(i(v6,v7),i(i(v7,v8),i(v6,v8))),i(i(i(i(i(v9,
    i(v9,v10)),i(v9,v10)),i(i(i(v11,v12),i(i(v12,v13),i(v11,v13))),v14)),v14),v15)),v15),
    i(i(v16,i(i(i(v17,v17),i(i(v18,v18), i(i(v19,v19),i(i(v20,v20),i(v16,v21))))),v21)),v22))),1).
%  Following are building blocks for Rezus and Rezus itself.
weight(i(i(x,y),i(i(y,z),i(x,z))),1).   % B'
weight(i(x,i(i(i(y,y),i(x,z)),z)),1).   % Rezus's K1
weight(i(x,i(i(i(y,y),i(i(z,z),i(i(u,u),i(i(v,v),i(x,w))))),w)),1). % Rezus's K4
weight(P(i(i(i(x,i(i(i(y,y),i(i(z,z),i(i(u,u),i(i(v,v),i(x,w))))),w)),i(i(i(i(i(v6,v7),i(i(v7,v8),i(v6,v8))),
    i(i(i(i(v9,i(v9,v10)), i(v9,v10)),i(i(i(v11,v12),i(i(v12,v13),i(v11,v13))),v14)),v14),v15)),v15),
    i(i(v16,i(i(i(v17,v17),i(i(v18,v18),i(i(v19,v19),i(i(v20,v20),i(v16,v21))))),v21)),v22))),v22)),1).
    % A Rezus-style single axiom for RI
end_of_list.

list(hints).
%  Following 20 derive B C I and W from the new Ulrich 35-symbol single axiom for R.
P(i(i(u,v),i(i(v,v),i(u,v)))).
P(i(i(u,i(i(v,w),i(x,v))),i(i(v,w),i(u,i(x,w))))).
P(i(i(u,i(i(v,v),i(w,v))),i(w,i(u,v)))).
P(i(i(u,i(i(i(v,w),i(v,w)),i(i(x,x),i(v,w)))),i(v,i(u,w)))).
P(i(u,i(i(u,v),v))).
P(i(u,i(i(i(v,v),i(u,w)),w))).
P(i(i(i(i(u,v),v),i(i(u,v),v)),i(u,i(i(u,v),v)))).
P(i(i(i(u,i(i(u,v),v)),w),w)).
P(i(i(i(u,u),i(i(v,i(i(i(w,w),i(v,x)),x)),y)),y)).
P(i(i(i(i(i(i(u,v),v),i(i(u,v),v)),i(u,i(i(u,v),v))),w),w)).
P(i(i(i(u,u),i(i(i(i(v,i(i(v,w),w)),x),x),y)),y)).
P(i(i(i(u,u),i(i(v,v),w)),w)).
P(i(u,u)). % I
P(i(i(u,i(v,w)),i(v,i(u,w)))). % C
P(i(u,i(i(i(v,v),i(i(i(i(w,i(i(w,x),x)),y),y),i(u,z))),z))).
P(i(i(u,v),i(i(v,w),i(u,w)))). % B
P(i(i(u,i(v,w)),i(i(i(v,w),v),i(u,w)))).
P(i(i(i(i(u,v),i(w,v)),x),i(i(w,u),x))).
P(i(i(i(i(u,v),v),i(u,v)),i(u,v))).
P(i(i(u,i(u,v)),i(u,v))). % W
end_of_list.

list(usable).
-P(i(x,y)) | -P(x) | P(y).
% -P(i(a,a)) | -P(i(i(a,i(b,c)),i(b,i(a,c)))) | -P(i(i(a,b),i(i(b,c),i(a,c)))) | -P(i(i(a,i(a,b)),i(a,b))) | $ANS(all).
end_of_list.

list(sos).
P(i(i(u,v),i(i(w,u),i(w,v)))). % B
P(i(i(u,i(v,w)),i(v,i(u,w)))). % C
P(i(u,u)). % I
P(i(i(u,i(u,v)),i(u,v))). % W
end_of_list.

list(passive).
%  Following 20 are negs of subformulas of Rezus, from Beeson.
-P(i(i(b6,b7),i(i(b7,b8),i(b6,b8)))) | $ANS(inter4).
-P(i(i(a3,a3),i(i(a4,a4),i(i(b,b),i(a1,a6))))) | $ANS(inter4).
-P(i(i(b11,b12),i(i(b12,b13),i(b11,b13)))) | $ANS(inter4).
-P(i(i(b19,b19),i(i(b20,b20),i(b16,b21)))) | $ANS(inter4).
```

```
-P(i(i(i(b11,b12),i(i(b12,b13),i(b11,b13))),b14)) | $ANS(inter4).
-P(i(i(a2,a2),i(i(a3,a3),i(i(a4,a4),i(i(b,b),i(a1,a6))))) | $ANS(inter4).
-P(i(i(i(a2,a2),i(i(a3,a3),i(i(a4,a4),i(i(b,b),i(a1,a6))))),a6)) | $ANS(inter4).
-P(i(i(b18,b18),i(i(b19,b19),i(i(b20,b20),i(b16,b21))))) | $ANS(inter4).
-P(i(a1,i(i(i(a2,a2),i(i(a3,a3),i(i(a4,a4),i(i(b,b),i(a1,a6))))),a6))) | $ANS(inter4).
-P(i(i(b17,b17),i(i(b18,b18),i(i(b19,b19),i(i(b20,b20),i(b16,b21)))))) | $ANS(inter4).
-P(i(i(i(b17,b17),i(i(b18,b18),i(i(b19,b19),i(i(b20,b20),i(b16,b21))))),b21)) | $ANS(inter4).
-P(i(i(i(b9,i(b9,b10)),i(b9,b10)),i(i(i(b11,b12),i(i(b12,b13),i(b11,b13))),b14))) | $ANS(inter4).
-P(i(b16,i(i(i(b17,b17),i(i(b18,b18),i(i(b19,b19),i(i(b20,b20),i(b16,b21))))),b21))) | $ANS(inter4).
-P(i(i(i(i(b9,i(b9,b10)),i(b9,b10)),i(i(i(b11,b12),i(i(b12,b13),i(b11,b13))),b14)),b14)) | $ANS(inter4).
-P(i(i(b16,i(i(i(b17,b17),i(i(b18,b18),i(i(b19,b19),i(i(b20,b20),i(b16,b21))))),b21)),b22)) | $ANS(inter4).
-P(i(i(i(i(i(b9,i(b9,b10)),i(b9,b10)),i(i(i(b11,b12),i(i(b12,b13),i(b11,b13))),b14)),b14),b15)) | $ANS(inter4).
-P(i(i(i(b6,b7),i(i(b7,b8),i(b6,b8))),i(i(i(i(i(b9,i(b9,b10)),i(b9,b10)),i(i(i(b11,b12),i(i(b12,b13),
    i(b11,b13))),b14)),b14),b15))) | $ANS(inter4).
-P(i(i(i(i(b6,b7),i(i(b7,b8),i(b6,b8))),i(i(i(i(i(b9,i(b9,b10)),i(b9,b10)),i(i(i(b11,b12),i(i(b12,b13),
    i(b11,b13))),b14)),b14),b15)),b15)) | $ANS(inter4).
-P(i(i(i(i(i(b6,b7),i(i(b7,b8),i(b6,b8))),i(i(i(i(i(b9,i(b9,b10)),i(b9,b10)),i(i(i(b11,b12),i(i(b12,b13),
    i(b11,b13))),b14)),b14),b15)),b15),i(i(b16,i(i(i(b17,b17),i(i(b18,b18),i(i(b19,b19),i(i(b20,b20),
    i(b16,b21))))),b21)),b22))) | $ANS(inter4).
-P(i(i(a1,i(i(i(a2,a2),i(i(a3,a3),i(i(a4,a4),i(i(b,b),i(a1,a6))))),a6)),i(i(i(i(i(b6,b7),i(i(b7,b8),i(b6,b8))),
    i(i(i(i(i(b9,i(b9,b10)),i(b9,b10)),i(i(i(b11,b12),i(i(b12,b13),i(b11,b13))),b14)),b14),b15)),b15),i(i(b16,
    i(i(i(b17,b17),i(i(b18,b18),i(i(b19,b19),i(i(b20,b20),i(b16,b21))))),b21)),b22)))) | $ANS(inter4).
%  Following are negs of B', K1, K4, and Rezus, the first 3 are building blocks for Rezus.
-P(i(i(a1,a2),i(i(a2,a3),i(a1,a3)))) | $ANS(BP).
-P(i(a1,i(i(i(a2,a2),i(a1,a3)),a3))) | $ANS(K1).
-P(i(a1,i(i(i(a2,a2),i(i(a3,a3),i(i(a4,a4),i(i(a5,a5),i(a1,a6))))),a6))) | $ANS(K4).
-P(i(i(i(a1,i(i(i(a2,a2),i(i(a3,a3),i(i(a4,a4),i(i(b,b),i(a1,a6))))),a6)),i(i(i(i(i(b6,b7),i(i(b7,b8),
i(b6,b8))),i(i(i(i(i(b9,i(b9,b10)),i(b9,b10)),i(i(i(b11,b12),i(i(b12,b13),i(b11,b13))),b14)),b14),b15)),b15),
    i(i(b16,i(i(i(b17,b17),i(i(b18,b18),i(i(b19,b19),i(i(b20,b20),i(b16,b21))))),b21)),b22))),b22)) |
    $ANS(REZ).
%  Following are related terms.
-P(i(i(i(i(i(a1,a2),i(a3,a1)),i(i(a1,a2),i(a3,a2))),i(i(a4,a4),i(a5,i(a6,a7)))),i(i(a8,a5),i(a6,i(a8,a7))))) |
    $ANS(U35).
-P(i(i(i(a1,b),i(a3,a1)),i(i(a1,b),i(a3,b)))) | $ANS(inter).
-P(i(i(a4,a4),i(i(a5,i(a6,b6)),i(i(b7,a5),i(a6,i(b7,b6)))))) | $ANS(inter).
-P(i(i(a5,i(a6,b6)),i(i(b7,a5),i(a6,i(b7,b6))))) | $ANS(inter).
end_of_list.

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

I assigned max_weight the value 24, in contrast to 15 (as in experiments discussed earlier), because I thought more latitude would be needed because the target is so complex, weight (in symbol count) 94. Conjecturing that the proof of Rezus would not require many more distinct variables than the formula itself, I also assigned max_distinct_vars the value 24 in part to curb the retention of new conclusions. However, after this experiment was under way, I decided (from curiosity) to make a second run in parallel with an assignment of the value 15 to max_weight. Indeed, perhaps 15 is a golden number: That value proved beneficial in other studies. In that run, I also assigned the value 6 to the pick_given_ratio. Automated reasoning owes McCune for the formulation of the pick_given_ratio

strategy. In that strategy, say with the assignment of the value 6, a reasoning program will choose for inference-rule initiating 6 clauses by complexity, 1 by first come first serve, 6, 1, and so on. I chose 6 to emphasize complexity quite a bit more than first come first serve.

The two runs behaved rather similarly, the most glaring exception being that the run with max_weight assigned the value 24 required the retention of clause (496621) to return its last success. Each run proved the same six subformulas (as theorems) as well as *B*', *K*1, and *K*4. However, even with much memory and much CPU-time, Rezus still eluded proof.

My conjecture had been born out, so I now turned to lemma adjunction. For the ''lemmas'', I relied on twenty-four formulas, proof steps of the items in list(passive) as described earlier, sorted to remove duplicates. I had two top-level choices: I could rely on a complexity-driven search, or I could rely on a level-saturation search. I chose the former, placing the selected twenty-four formulas in list(sos), including the command set(input_sos_first), and nothing of note resulted. The included command instructs the program to focus for inference-rule initiation on the items in the initial set of support before considering a newly retained item. I still did not rely on ancestor subsumption to save time. I then turned to level saturation by including the command set(sos_queue), commented out the use of the pick_given_ratio strategy, and assigned max_weight the value 15 (rather than 24) with the goal of enabling the program to examine more levels than it might otherwise. OTTER returned but one additional proof, but it was a proof of Rezus, of length 6 and level 3, with the retention of clause (256405) in approximately 6155 CPU-seconds.

The obvious conclusion is that the Rezus formula can be proved from *B*, *C*, *I*, and *W* in thirty or fewer steps. Indeed, you cannot assume that all twenty-four lemmas participated in the deduction of the additional six formulas that completed a proof of Rezus, and, even if they did in the run that found the proof, perhaps some of them would not be needed. Eager to see a proof, I placed the cited six steps as resonators in weight_list(pick_and_purge), each assigned a value of -2, then placed the twenty-four formulas that were used as lemmas but now as resonators, each assigned a value of -1, followed by weight templates for the twenty subformulas and those for the building blocks. I commented out sos_queue, commented in the pick_given_ratio strategy with an assignment of the value 6, and invoked ancestor subsumption. In other words, I was a bit greedy, seeking not only a proof of Rezus, but also seeking a proof possibly much shorter than length 30. Also, as expected, I removed from list(sos) the twenty-four formulas used as lemmas.

OTTER found two proofs of Rezus of lengths 22 then 21, both of level 8. So, typical of my research, I decided to see if I could find a shorter proof. (In keeping with history, some weeks earlier, as the result of a different set of experiments, I had in hand a proof of length 19. Now would it not be poetic justice if this systematic approach led to finding a proof of length less than 19?) As is my custom, I next tried blocking the steps of the 21-step proof one at a time with demodulation, demodulating each to junk, but nothing resulted. Then I adjoined resonators corresponding to the steps of the 21-step proof, each assigned a value (weight) of -3, to give preference to formulas that matched, at the functional level, one of the 21. The experiment was successful: The program found a 20-step proof of level 9, containing one step not in the original 21-step proof. I find it interesting to note that the newer 20-step proof differed by just one step, but demodulation blocking was not able to find this proof; approximately 1169 CPU-seconds was required.

Of course, I again tried blocking with demodulation the steps of the 20-step proof one at a time, with no progress occurring. So, in imitation of that which had just worked, I adjoined twenty resonators, corresponding to the 20-step proof, each assigned a value of -4. After patience (not my long suit) and a study, OTTER completed a new proof with retention of clause (471267), after approximately 18792 CPU-seconds, a proof of length 19. The (new) 19-step proof contains two formulas not in the 20-step proof. Although the new 19-step proof, as is true of the older 19-step proof, has level 9, it contains six steps not in the earlier proof, and its pseudo-size is 1505 in contrast to a size of 1470 for the older proof. (The size of a proof measures the number of symbols in the deduced steps, of course, not counting parentheses and commas.) Here I am citing pseudo-size, which counts predicate symbols, parentheses, and commas. (Along the way, I did try some *cramming*, to be discussed, with sos_queue, which has the program use level saturation, and I relied on eighteen added lemmas, taken from proof steps; the effort was to no avail.)

I pause here to discuss the *cramming strategy*. The cramming strategy derives its name from its nature. In effect, the strategy focuses on a chosen subproof of the total proof in hand and then attempts to cram steps of the chosen subproof into new subproofs needed to complete a new and shorter total proof of the conjunction. The degree of success (measured in the decrease in total proof length) depends on how many deduced steps of the chosen subproof can be made to play double-duty, triple-duty, or more, in the sense that (for a given deduced step) it is used in two subproofs, three subproofs, or more. Alternatively, the cramming strategy derives its name from emphasizing the role of the subproof of one element (formula or equation) and attempting to find subproofs of the remaining elements such that their proof lengths can be (so-to-speak) crammed into a chosen length *j*. Typically, *j* is chosen to be less than the length of the shortest proof known for the theorem under study. In a typical case, the goal is to prove the conjunction of a set of formulas or equations. In that situation, you can choose the proof (subproof) of one of its members on which to cram. For the strategy, you take the steps of the chosen subproof and place them in (the initial) list(sos) and have the program attempt to force some or all of those steps into what is needed to complete the so-called total proof. Because of the basic mechanism that is employed by the strategy, level saturation, it seems almost certain that none of the masters of logic or mathematics applied such an approach, whereas cramming is indeed well suited to the actions of an automated reasoning program.

Again, demodulation blocking yielded no progress. Since imitation is said to be one of the finer forms of flattery—even, I guess, if imitating oneself—I continued in the manner in focus. I adjoined nineteen resonators, each assigned a value of -5, those corresponding to the just-cited 19-step proof. After the retention of clause (467271) and the use of approximately 19181 CPU-seconds, this fine assistant (OTTER) broke through, returning an 18-step level-9 proof of Rezus. That proof, which I now give, has but one step not in the just-cited 19-step proof.

An 18-Step Proof of Rezus in RI

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on jaguar.mcs.anl.gov,
Fri Feb  8 18:56:47 2008
The command was "otter".  The process ID is 4325.

----> UNIT CONFLICT at 19181.97 sec ----> 467272 [binary,467271.1,29.1] $ANS(REZ).

Length of proof is 18.  Level of proof is 9.

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

1 [] -P(i(x,y)) | -P(x) | P(y).
2 [] P(i(i(u,v),i(i(w,u),i(w,v)))).
3 [] P(i(i(u,i(v,w)),i(v,i(u,w)))).
4 [] P(i(u,u)).
5 [] P(i(i(u,i(u,v)),i(u,v))).
29 [] -P(i(i(i(a1,i(i(i(a2,a2),i(i(a3,a3),i(i(a4,a4),i(i(b,b),i(a1,a6))))),a6)),i(i(i(i(i(b6,b7),i(i(b7,b8),i(b6,b8))),
     i(i(i(i(i(b9,i(b9,b10)),i(b9,b10)),i(i(i(b11,b12),i(i(b12,b13),i(b11,b13))),b14)),b14),b15)),b15),i(i(b16,
     i(i(i(b17,b17),i(i(b18,b18),i(i(b19,b19),i(i(b20,b20),i(b16,b21))))),b21)),b22))),b22)) | $ANS(REZ).
34 [hyper,1,3,3] P(i(x,i(i(y,i(x,z)),i(y,z)))).
35 [hyper,1,2,3] P(i(i(x,i(y,i(z,u))),i(x,i(z,i(y,u))))).
36 [hyper,1,3,2] P(i(i(x,y),i(i(y,z),i(x,z)))).
43 [hyper,1,34,4] P(i(i(x,i(i(y,y),z)),i(x,z))).
51 [hyper,1,34,36] P(i(i(x,i(i(i(y,z),i(i(z,u),i(y,u))),v)),i(x,v))).
54 [hyper,1,36,2] P(i(i(i(i(x,y),i(x,z)),u),i(i(y,z),u))).
58 [hyper,1,43,43] P(i(i(i(x,x),i(i(y,y),z)),z)).
92 [hyper,1,3,51] P(i(x,i(i(x,i(i(i(y,z),i(i(z,u),i(y,u))),v)),v))).
105 [hyper,1,54,35] P(i(i(x,i(y,z)),i(i(u,x),i(y,i(u,z))))).
160 [hyper,1,92,5] P(i(i(i(i(x,i(x,y)),i(x,y)),i(i(i(z,u),i(i(u,v),i(z,v))),w)),w)).
```

180 [hyper,1,105,58] P(i(i(x,i(i(y,y),i(i(z,z),i(u,v)))),i(u,i(x,v)))).
197 [hyper,1,34,160] P(i(i(x,i(i(i(i(i(y,i(y,z)),i(y,z)),i(i(i(u,v),i(i(v,w),i(u,w))),v6)),v6),v7)),i(x,v7))).
202 [hyper,1,180,180] P(i(x,i(i(i(y,y),i(i(z,z),i(i(u,u),i(i(v,v),i(x,w))))),w))).
268 [hyper,1,51,197] P(i(i(i(i(x,y),i(i(y,z),i(x,z))),i(i(i(i(i(u,i(u,v)),i(u,v)),i(i(i(w,v6),i(i(v6,v7),
i(w,v7))),v8)),v8),v9)),v9)).
323 [hyper,1,34,202] P(i(i(x,i(i(y,i(i(i(z,z),i(i(u,u),i(i(v,v),i(i(w,w),i(y,v6))))),v6)),v7)),i(x,v7))).
395 [hyper,1,34,268] P(i(i(x,i(i(i(i(i(y,z),i(i(z,u),i(y,u))),i(i(i(i(i(v,i(v,w)),i(v,w)),i(i(i(v6,v7),i(i(v7,v8),
i(v6,v8))),v9)),v9),v10)),v10),v11)),i(x,v11))).
569 [hyper,1,323,395] P(i(i(i(x,i(i(i(y,y),i(i(z,z),i(i(u,u),i(i(v,v),i(x,w))))),w)),i(i(i(i(i(v6,v7),i(i(v7,v8),
i(v6,v8))),i(i(i(i(i(v9,i(v9,v10)),i(v9,v10)),i(i(i(v11,v12),i(i(v12,v13),
i(v11,v13))),v14)),v14),v15)),v15),v16)),v16)).
467271 [hyper,1,323,569] P(i(i(i(x,i(i(i(y,y),i(i(z,z),i(i(u,u),i(i(v,v),i(x,w))))),w)),i(i(i(i(i(v6,v7),
i(i(v7,v8),i(v6,v8))),i(i(i(i(i(v9,i(v9,v10)),i(v9,v10)),i(i(i(v11,v12),i(i(v12,v13),
i(v11,v13))),v14)),v14),v15)),v15),i(i(v16,i(i(i(v17,v17),i(i(v18,v18),i(i(v19,v19),
i(i(v20,v20),i(v16,v21))))),v21)),v22))),v22)).

At this point, in early 2008—and here is a challenge for you if you have time and energy—various attempts have failed to yield a shorter proof than that of length 18. Yes, I again imitated that which has been described, adjoining eighteen new resonators, each assigned the value -6, but no further progress resulted. However, this iterative approach based on ever-expanding sets of resonators did prove quite effective. Of course, the experiments still relied on weight templates corresponding to twenty subformulas as well as those corresponding to building blocks pertinent to the Rezus formula.

I also tried cramming on a proof step not too far from the end of the 18th step of the Rezus proof. For clarity, with cramming, you need not focus on the proof of a member of a conjunction in order to make progress.

To complete this section, I keep my implied promise of satisfying the first objective, of giving a proof that the Rezus formula implies the join of *B*, *C*, *I*, and *W*. With that proof, the status of Rezus being a single axiom for *RI* is established.

A 17-Step Proof Showing Rezus Implies the Join

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on elephant.mcs.anl.gov,
Sat Feb 16 18:13:51 2008
The command was "otter". The process ID is 8484.

-----> EMPTY CLAUSE at 0.05 sec ----> 351 [hyper,67,150,166,268,115] $ANS(all).

Length of proof is 17. Level of proof is 13.

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

66 [] -P(i(x,y)) | -P(x) | P(y).
67 [] -P(i(a,a)) | -P(i(i(a,i(b,c)),i(b,i(a,c)))) | -P(i(i(a,b),i(i(c,a),i(c,b)))) | -P(i(i(a,i(a,b)),i(a,b))) |
$ANS(all).
68 [] P(i(i(i(x,i(i(i(y,y),i(i(z,z),i(i(u,u),i(i(v,v),i(x,w))))),w)),i(i(i(i(i(v6,v7),i(i(v7,v8),i(v6,v8))),
i(i(i(i(i(v9,i(v9,v10)),i(v9,v10)),i(i(i(v11,v12),i(i(v12,v13),i(v11,v13))),v14)),v14),v15)),v15),
i(i(v16,i(i(i(v17,v17),(i(v18,v18),i(i(v19,v19),i(i(v20,v20),i(v16,v21))))),v21)),v22))),v22)).
79 [hyper,66,68,68] P(i(i(i(x,x),i(i(y,y),i(i(z,z),i(i(u,u),i(i(v,i(i(i(w,w),i(i(v6,v6),i(i(v7,v7),
i(i(v8,v8),i(v,v9))))),v9)),v10))))),v10)).
80 [hyper,66,79,79] P(i(x,i(i(i(y,y),i(i(z,z),i(i(u,u),i(i(v,v),i(x,w))))),w))).
81 [hyper,66,68,79] P(i(i(i(i(x,y),i(i(y,z),i(x,z))),i(i(i(i(i(u,i(u,v)),i(u,v)),i(i(i(w,v6),
i(i(v6,v7),i(w,v7))),v8)),v8),v9)),v9)).
83 [hyper,66,81,80] P(i(i(x,y),i(i(y,z),i(x,z)))).
84 [hyper,66,83,83] P(i(i(i(i(x,y),i(z,y)),u),i(i(z,x),u))).

85 [hyper,66,81,83] P(i(i(i(x,y),i(i(y,z),i(x,z))),i(u,i(u,v))),i(u,v))).
86 [hyper,66,84,84] P(i(i(x,i(y,z)),i(i(u,y),i(x,i(u,z))))).
93 [hyper,66,84,86] P(i(i(x,y),i(i(z,x),i(i(y,u),i(z,u))))).
106 [hyper,66,93,84] P(i(i(x,i(i(i(y,z),i(u,z)),v)),i(i(i(i(u,y),v),w),i(x,w)))).
111 [hyper,66,106,80] P(i(i(i(i(i(x,x),i(i(y,y),i(i(z,z),i(u,v)))),v),w),i(u,w))).
112 [hyper,66,111,111] P(i(x,i(i(i(y,y),i(i(z,z),i(x,u))),u))).
115 [hyper,66,111,85] P(i(i(x,i(x,y)),i(x,y))).
134 [hyper,66,86,112] P(i(i(x,i(i(y,y),i(i(z,z),i(u,v)))),i(u,i(x,v)))).
150 [hyper,66,68,134] P(i(x,x)).
154 [hyper,66,134,93] P(i(x,i(i(x,y),y))).
166 [hyper,66,86,154] P(i(i(x,i(y,z)),i(y,i(x,z)))).
268 [hyper,66,166,83] P(i(i(x,y),i(i(z,x),i(z,y)))).

6. Conclusions, Notes, and Review

OTTER is not required to apply the techniques, approaches, and strategies offered in this and other notebooks on my website. Indeed, what is discussed is general in most cases. However, not surprising, I find McCune's program superior in many ways. This program is unusually robust, almost never encountering a bug, and providing (if asked) copious detail. You might prefer relying on a quite different automated reasoning program. Or, just perhaps, you will write your own program.

The nature of the subformula strategy is iterative. Nevertheless, occasionally a single run suffices to return the desired proof(s).

As evidenced in this notebook, I do rely sometimes on a level-saturation approach, an approach that is exclusively first come first serve, deducing all of the level-1 items, then of level-2, and the like. Unrestricted, almost always, this approach is impractical. In the vast majority of cases, this approach, to be effective at all, requires limits placed on it, limits that include setting a rather tight max_weight, to limit the complexity of retained information. Other restrictions have proved useful, such as blocking all newly deduced items if they rely on double negation, contain terms of the form $n(n(t))$ for some term t. Restrictions of this type force the program into another part of the universe of deducible conclusions. If you place a limit on max_distinct_vars, the number of distinct variables allowed in a retained conclusion, you can provide assistance to the program that might be required.

Clearly, my decades of experimentation with automated reasoning programs, various strategies, and various approaches have influenced me greatly, perhaps biased is more accurate. In these notebooks, I attempt to convey my views on goal seeking with a program that reasons logically, drawing conclusions that follow flawlessly from the given hypothesis or hypotheses.

Given these comments, you might wonder if level saturation encounters any other major danger. Yes, it does. In particular, a formula or equation might be forced into a partially complete proof that in turn forces the completion to be longer than it need be. Of course, this phenomenon can occur with a complexity-preference approach also. You see, as remarked in various notebooks, no algorithm exists (from what I know) for finding shorter and still shorter proofs, which explains in part why I have written notebooks. I know of no way to accurately identify and then avoid an item that (in effect) leads to a proof of undesirable length. However, as I have noted many times elsewhere, the use of demodulation to block proof steps one at a time does sometimes (in effect) identify a formula or equation whose presence leads to a proof of unnecessary length. A closer scrutiny of this technique, when successful, fits in well with the notion that certain items, if present in a proof before completion, can send the program down a path that terminates in a (total) proof longer than need be. Put another way, such an item can interfere with finding a shorter proof.

As proofs of individual items get shorter and shorter, the so-called total proof that results can also get longer and longer, or fluctuate. for an illustration, I draw on an experiment focusing on the Meredith single axiom for classical propositional calculus, where the target is an axiom system due to Hilbert. In that experiment, OTTER returned proofs of respective lengths 77, 125, 68, 77, 79, 84, and 92. The levels ranged from 27 to 28. The explanation for the sequence of proof lengths of the target axiom system rests with finding shorter and still shorter proofs of individual members of the axiom system.

My current, grand conclusion, based on the foregoing, is that automated reasoning now offers an approach to proving extremely complex formulas. The key, or at least one key, is the use of the sub-formula strategy. Overbeek, whose approach is (for years) to begin with max_weight=1, 2, and the like until something good happens, again plays a vital role in this field.