An Amazing Approach to Plane Geometry

Larry Wos

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

1. After More Than Thirty-Five Years, Success

Did you enjoy plane geometry in high school? Do you find triangles, rectangles, hexagons, and other such figures esthetically pleasing? Do you recall theorems that asserted the equivalence of two figures with a justification of, say, side-angle-side? Well, whether or not you were or are captivated by this area of mathematics, called plane geometry, you may find most intriguing the material presented in this notebook. Indeed, if you have known and consulted with a brilliant mind that aided you in your research, here you will (possibly again) learn of an assistant, an automated reasoning program, that can also be most effective in the context of finding proofs for deep theorems, theorems taken from geometry. You will find sprinkled throughout data regarding experiments. In poker, success is measured in terms of how often you win and how much you win. In automated reasoning, I measure success in less—shorter proof length, less CPU time, or fewer conclusions retained. Data of this type provides, at least indirectly, important clues to how you might proceed in the future.

You will encounter, for example, situations where the inclusion or omission of a single item can mean the difference between success and failure. Yes, the use of a program such as William McCune's powerful automated reasoning program OTTER can be straightforward or, on the other hand, can be an art. You will learn in these sections how a field, Tarskian geometry, that was virtually impenetrable more than thirty-five years ago, when the Argonne group studied it, finally yielded its treasure to OTTER and to two researchers, Michael Beeson and me—and treasure in abundance.

2. Wellspring and Some Background

The wellspring for this article is my colleague Beeson's intriguing plan for us to use an automated reasoning program to find proofs for all the theorems in the book by Schwabhauser, Szmielew, and Tarski titled *Metamathematische Methoden in der Geometrie* (hereafter referred to as SST). And some background is merited here to explain the origin of that book.

2.1. Geometry as a Testbed

Geometry has been a testbed for automated deduction almost as long as computers have existed. The first experiments were done in the 1950s. In the nineteenth century, geometry was the testbed for the development of the axiomatic method in mathematics, spurred by efforts to prove Euclid's parallel postulate from his other postulates and ultimately the development of non-Euclidean geometry. These efforts culminated in Hilbert's seminal 1899 book *Grundlagen der Geometrie* (*Foundations of Geometry*). In the period 1927-1965, Tarski developed his simple and short axiom system (presented in Section 3.1). In the late 1970s, some experimentation, at Argonne National Laboratory, was conducted aimed at finding proofs from Tarski's axioms; but despite success with proving very simple theorems, several problems were left unsolved. The subject was revisited by Art Quaife, who in his 1992 book *Automated Development of Fundamental Mathematical Theories* reported on the successful solution of some of those challenge problems

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

using an early version of McCune's automated reasoning program OTTER. But several theorems remained that Quaife was not able to prove with OTTER, and he stated them as "challenge problems" in his book.

Quaife's four challenge problems were the following: (1) every line segment has a midpoint; (2) every segment is the base of some isosceles triangle; (3) the outer Pasch axiom (assuming inner Pasch as an axiom); and (4) the first outer connectivity property of betweenness. These are to be proved without relying on any parallel axiom and without even relying on line-circle continuity. They are difficult theorems to prove, the first proofs of which were the heart of H. N. Gupta's Ph.D. thesis under Tarski.

To be more specific, the immediate stimulus for this notebook was the existence of the almost-formal development of many theorems in Tarskian geometry in Part I of SST. Part I is essentially the manuscript developed by Szmielew for her 1965 Berkeley lectures on the foundations of geometry, with "inessential modifications" by Schwabhauser. The manuscript consists of 16 chapters. Quaife's challenge problems occur in the first nine chapters. Gupta's thesis was never published, although it is available from a database of theses; but SST contains the only published version of his proofs.

2.2. Original Plans and a Note about Dependencies

The original goal that Beeson and I had established was to find proofs, not to check proofs, a topic for later in this notebook where the difference is discussed. Beeson says that all the proofs of the theorems through Chapter 12 of the book have been proof checked. Consistent with conversations I had many years ago with Bob Boyer, I warn the uninitiated that checking a proof is a totally different activity from finding a proof for a given theorem. Beeson's plan, though frequently interrupted by other demands, has us prove one theorem after another, then adjoin, for the next study, the theorem in an appropriate list and proceed to the next theorem. As I write this notebook, this plan is being followed and is far from complete.

As in earlier notebooks, you will find here no mention of ruler and compass, no use of actual diagrams—although, with clauses supplied by Beeson, the diagrams (in effect) do play a key role. Instead, you will learn of proofs for theorems in plane geometry that do not rely on your drawing of figures, computing angles, and using items such as a protractor. In early sections, the proofs given here are based on a set of thirteen axioms provided in 1959 by Alfred Tarski for the study of plane geometry. For quite a while, I thought two of these—transitivity and connectivity of betweenness—were dependent. (Indeed, I think it most likely that Tarski himself was unaware of the true facts.) My original plan for this notebook, therefore, called for a discussion and, possibly, proofs of the two dependencies. Thus, much of this notebook proceeds as if the two axioms were dependent. I suggest that as you read the next few sections, you keep in mind that I labored under this erroneous impression, which is born out by my often finding proofs in which both transitivity and connectivity are not employed. When I finally learned of the true situation, I had to abandon the plan. Thus, in Section 4 and following, I will note that transitivity of betweenness is in fact independent when the Tarski 1959 axiom system is in use; and I will, instead, prove two dependencies in another axiom system (closely related to that given in Section 3.1) for Tarskian geometry.

The task of proving that connectivity of betweenness is dependent presents quite a challenge for the system you are about to encounter, a challenge that I attempt to meet, as seen later in this notebook. (At this point in the notebook, I still have not seen such a proof; perhaps before the notebook is complete, I will have a proof.)

As you browse in this notebook, you will encounter what to me are most piquant phenomena, some that are successes, some that are failures, and some that are puzzles. Indeed, as is sometimes the case when you consult a learned individual, the outcome is hard to predict. For a tiny example, the likelihood of reaching your goal with the use of OTTER can be sharply affected by the values assigned to various parameters. Rather than being a discouragement, however, what you can do with a program such as OTTER will, for many, be astounding. Now, a bit of history is in order.

3. Some History and Some Challenges

I, with colleagues at Argonne National Laboratory, some thirty-five years or more ago sought proofs of elementary theorems with the use of the Tarski axiom system that I present shortly, in a notation that is acceptable to the automated reasoning program OTTER. Our attempts were, in the main, highly unsuccessful, to say the least. I will not present a detailed analysis of why we were so incompetent—although I will hazard some guesses in that regard—but, rather, show you how in 2012 and 2013, with my colleague Beeson and, of course, McCune's automated reasoning program OTTER, proofs of very difficult theorems (to prove) were obtained.

At least in the beginning, I will focus on the use of the axioms our group at Argonne National Laboratory used those decades ago. A slight variation of those axioms will come into play when I progress to my current studies with Beeson; however, because of the cited study of more than thirty-five years ago with its accompanying lack of success, I have chosen for the early part of this notebook to rely on the axioms used so long ago. By doing so, I can follow one of the principles of science: the importance of showing how progress occurred over many, many years. To immediately dispel a natural conjecture, I emphasize that the breakthrough is not mainly a result of far, far more powerful computers; of course, the added power is indeed valuable.

To provide you with a taste of the strangeness—I believe that is a good assessment—of what you will encounter, I offer a few intuitively appealing theorems to consider; indeed, you might treat each of them as a challenge. The axioms that Tarski gives rely on two predicates (relations): T(x,y,z), which may be thought of as asserting that "*y* is between *x* and *z*" (not necessarily strictly between), and E(x1,y1,x2,y2), which may be thought of as asserting that "the distance from *x1* to *y1* is the same as the distance from *x2* to *y2*".

Intuitively, anybody so-to-speak knows what it means to say that the point *B* is between the points *A* and *C*. Indeed, immediately, when you hear such, you think of a line that begins at *A*, passes through *B*, and ends at *C*. One of the theorems you could be asked to prove, relying on the Tarski system and whichever inference rules you had in mind—of course, the clause representation is what is used throughout this notebook—asserts that, for all *x*, *y*, and *z*, if *y* is between *x* and *z*, then *y* is between *z* and *x*. This theorem refers to the symmetry of betweenness. Now, if you (in effect) ask your niece or nephew about this result, you may be told something like "how silly, it's obvious". It certainly is obvious if you depend on a picture. If you are constrained to the use of the Tarski axioms, however, you might find the implied task, of proving this theorem focusing on symmetry of betweenness, far from trivial. You might also be surprised—but perhaps the cited strangeness gives you a warning—to find that axioms focusing on equidistance are utilized. Later in this notebook, I will present approaches to proving the symmetry of betweenness, as well as proving other theorems. Therefore, if you would enjoy seeking your own proof, read only to the end of this section, when you will have in hand the 1959 Tarski axiom system.

Instead, I offer you a simper theorem to consider, once you have in hand the Tarski system that I will shortly supply. You are asked to prove that *y* is between *x* and *y*, for all *x* and *y*. Yes, Tarski does not mean strictly between. A bit harder, I offer you the challenge of proving that *x* is between *x* and *y*, for all *x* and *y*. The challenge of proving the symmetry of betweenness is clearly more daunting than either of these other two challenges. Again, you are advised to read no further than the completion of this section if you wish to devise your own approach to proving one of the three cited theorems. Now, if you wish to attempt to prove a theorem I have at this point in the notebook not yet even considered for study (but I may focus on it in a later section), I suggest you try to prove that the diagonals of a rectangle bisect each other, relying, of course, on the Tarski system. (To be clear, I am using here the definition of a rectangle as a four-sided shape made up of two pairs of parallel lines and having four right angles. I am also assuming that the two diagonals do intersect.)

You might naturally wonder how those three challenges were met, if they were, by our automated reasoning group those more than thirty-five years ago, or more recently. As you know, recovering the precise details of decades ago is far from simple, and often the effort produces questionable information. However, I can draw on Chapter 6 of a book I wrote in 1987, a book offering "33 Basic Research Problems". My intention at that time, if memory serves, was to increase interest in the field of automated reasoning and present problems that, if solved, would add to the power of reasoning programs. Chapter 6 is devoted to test problems and experimentation, problems that can be used to begin an evaluation of a solution to one of the research problems offered in the cited book. Some of the test problems focus on Tarskian geometry. If you browse in the book, you will find that, in place of the predicate (relation) *T* (used here) for betweenness, the predicate *B* was used, and also that instead of the predicate *E* (used here), the predicate *L* was used.

3.1. Five-Point Theorem

Test Problem 10 asks you to prove the five-point theorem. In particular, you are told that point *A*3 is between points *A*2 and *A*4 and that each of *A*2 and *A*4 is between *A*1 and *A*5. As you may have surmised, the theorem asks you to prove that *A*3 is between *A*1 and *A*5. As for axioms, you are required to use the Tarski axiom set, which consists of the following twenty (so-to-speak) clauses, not counting axioms concerning the substitutivity of equality (as well as other obvious properties of equality), on which I will comment further.

```
identity axiom for betweenness:
-T(x,y,x) | (x = y).
transitivity axiom of betweenness:
-T(x,y,u) | -T(y,z,u) | T(x,y,z).
connectivity axiom for betweenness:
-T(x,y,z) | -T(x,y,u) | (x = y) | T(x,z,u) | T(x,u,z).
reflexivity axiom for equidistance:
E(x,y,y,x).
identity axiom for equidistance:
-E(x,y,z,z) | (x = y).
transitivity axiom for equidistance:
-E(x,y,z,u) | -E(x,y,v,w) | E(z,u,v,w).
Outer Pasch's axiom (2 clauses):
-T(x,v,u) | -T(y,u,z) | T(x,f1(v,x,y,z,u),y).
-T(x,v,u) | -T(y,u,z) | T(z,v,f1(v,x,y,z,u)).
Euclid's axiom (three clauses):
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,z,f2(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,y,f3(v,x,y,z,u)).
% Note that one reader has expressed concern that the following clause is not correctly represented.
% This clause appears in several places in the input files in this notebook.
-T(x,u,v) | -T(y,u,z) | (x = u) | T(f2(v6,x,y,z,u),v6,f3(v6,x,y,z,u)).
five-segment axiom:
-E(x1,y1,x2,y2) | -E(y1,z1,y2,z2) | -E(x1,u1,x2,u2) | -E(y1,u1,y2,u2) | -T(x1,y1,z1) | -T(x2,y2,z2) |
 (x1 = y1) | E(z1,u1,z2,u2).
axiom of segment construction (two clauses):
T(x,y,f4(x,y,u,v)).
E(y,f4(x,y,u,v),u,v).
lower dimension axiom (three clauses):
-T(c1,c2,c3).
-T(c2,c3,c1).
-T(c3,c1,c2).
upper dimension axiom:
-E(x,u,x,v) | -E(y,u,y,v) | -E(z,u,z,v) | (u = v) | T(x,y,z) | T(y,z,x) | T(z,x,y).
weakened continuity axiom (two clauses):
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | E(u1,y1,u1,f5(x1,y1,z1,x2,z2,u1)).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | T(x2,f5(x1,y1,z1,x2,z2,u1),z2).
```

The cited book does not offer a proof of the five-point theorem. Indeed, I feel it safe to say that we could not have dented the proof in 1987. If you consult the book, however, you will find in Chapter 6 two proofs relying on the Tarski system. The first proof establishes that, for all *x* and *y*, *x* is between *x* and *y*. The second proof shows that betweenness is indeed symmetric. As the first proof proceeds, you learn that, for all *x* and *y*, *y* is between *x* and *y*. In each of the two proofs—and here you gain much insight into how we attacked Tarskian geometry—the same axiom is cited before deductions are presented. That axiom, the following, is for the substitutivity of equality in the context of the predicate for betweenness. (As in preceding notebooks, "-" denotes logical **not**, and " | " denotes logical **or**.)

u != v | -T(x,y,u) | T(x,y,v).

If you consult Chapter 6, you will immediately ask why paramodulation was not being used. After all, that inference rule (which generalizes equality substitution) was published in the late 1960s. Further confusion may result when you learn that, if you supply the various axioms focusing on equality properties in the context of the Tarski 13-axiom system, you must adjoin thirty-five additional (so-to-speak) clauses. In the treatment of plane geometry featured in this notebook, paramodulation plays a vital role. And you now have a powerful clue that explains why we, our group, were thwarted those many, many years ago by Tarski; indeed, not using paramodulation was, I fear, more than an oversight. I suggest that, without its use, a challenge offered in 1990 by Art Quaife would not have been met, a challenge that asks for a proof of the dependence of connectivity of betweenness when you use a Tarskian axiom system that relies on inner Pasch. The successful meeting of that challenge is perhaps the high point of what you will read here.

Just as I began this notebook, I told my colleague Ross Overbeek of the joint effort focusing on Tarski being conducted by Beeson and me. He was most pleased. When I told him of the five-point theorem, he suggested the following theorem for study, a theorem to be called the four-point theorem. (As noted, the five-point theorem says that, if A3 is between A2 and A4 and if each of A2 and A4 is between A1 and A5, then A3 is provably between A1 and A5.) For the four-point theorem, if the points $A2$ and $A3$ are each between the points $A1$ and $A4$, then either $A2$ is between $A1$ and $A3$ or the point $A3$ is between $A1$ and $A2$. Overbeek was unable to use OTTER to prove the theorem he suggested; but shortly thereafter, he was able to use the program to obtain a proof of the five-point theorem. At this time in my writing of this notebook, I had as yet not found a proof for the four-point theorem; however, as it turned out, much later in my study of Tarskian geometry, the proof was obtained.

3.2. Proof Shortening, Assumption Reduction, and Other Topics to Come

Two other topics will appear in this notebook. The first concerns proof shortening, an activity that interested various mathematicians and logicians that include Hilbert, C. A. Meredith, A. N. Prior, and D. Ulrich. If, in particular, you find a proof in a book or paper or have found a proof with a program or with your own mind and if you wish to find a more elegant proof, where elegance is concerned with proof length (number of deduced steps), here you will learn how OTTER and various methodologies can often get you your wish. Somewhat similar to proof shortening is what might be termed axiom-and-theorem reduction or assumption reduction. Specifically, in an OTTER proof, you find, before any deduced item is presented, all of the items it used to obtain a proof, items taken from the input. Possibly because of Jesse Alama's success with finding dependent axioms among those offered for some area or for some theorem, which he communicated to me by email, you will find studies aimed at reducing, for a given theorem and proof of that theorem, the number of items (taken from the input) needed to obtain a proof. Those items (from the Szmielew book) ordinarily in this study of Tarski include axioms she supplied and theorems already proved and, typically, adjoined to an input file. (Near the end of the writing of this notebook, I learned by email from Alama that he also studies the reduction of items needed to obtain a proof of some given theorem.)

Well, the time for your independent and uninfluenced consideration of the challenge theorems is up, or you may wish to wait a bit longer before turning to the next section. Indeed, I will, in the next section, discuss possible approaches to proving the four theorems cited in this section. Specifically, to be in focus are $T(x,y,y)$, $T(x,x,y)$, the four-point theorem suggested by Overbeek, and the symmetry of betweenness theorem. These approaches apply not only to the Tarski treatment of geometry but also to areas of algebra, to other fields of mathematics, and to various areas of logic. As this notebook proceeds, focusing on ever-more-difficult theorems, studied jointly by Beeson and me, the methodologies will be utilized again and again. Near the end of this notebook, you will find what might be most intriguing, namely, a theorem whose proof is given from the viewpoint of mathematics and yet resists various attempts to obtain a proof with OTTER even though a first-order proof does exist.

4. Approaches for Proving Theorems and Various Proofs Obtained

I hope your curiosity is mounting, not only in the context of how to use an automated reasoning program to attack the four challenge theorems (focusing on betweenness), but also in the context of how the proofs involve notions that pertain to equidistance. Even if you have but a small interest in geometry, or perhaps none, you might as a puzzle solver find what follows quite entertaining. As you read through this section, I will take you on a labyrinthian journey whose goal, in addition to finding various proofs and illustrating diverse approaches, is the discovery of an approach that is an alternative to the Beeson plan. Specifically, although his intriguing plan of proving one theorem after another and, when successful, adding the theorem to the input file in search of the next proof—which is in the spirit of lemma adjunction—clearly merits intense consideration and is often quite effective, how satisfying it would be to discover an input file whose use would prove, in one run, many, many theorems of the Szmielew book. For a foretaste of what you will find in this section, I note that you will encounter a number of treatments of a single theorem, the five-point theorem. The diverse treatments illustrate many, many different aspects, and they may provide you with alternatives useful in contexts unrelated to that in focus in this notebook.

You might naturally wonder about various approaches to proving theorems in Tarskian geometry. Or, you might have no strong interest in geometry of any type and still wonder about methodologies for proving theorems, especially proving deep theorems. In this section, you will learn of various techniques, accompanied in some cases by commentary focusing on the obstacles to be encountered, the disadvantages, and the advantages of various methodologies.

Before you choose the approach you will use, from among those to be discussed in this section, you are asked to choose which type of proof you wish your program to attempt to complete. I have in mind three types of proof: forward proof, backward proof, and bidirectional proof. In a forward proof, the program relies on the axioms and lemmas that are placed in the input file and uses the denial of the theorem, or part of that denial, only to signal proof completion. With OTTER, when seeking a forward proof, the denial or part of the denial is placed in list(passive). Elements of list(passive) do not participate in the reasoning; they are used to detect unit conflict, for a determination of the completion of a proof, and for subsumption.

Overbeek, upon reading this notebook, correctly pointed out that the concept of forward proof is more complicated than it might at first appear. For a simple example of seeking a forward proof, if the goal is to prove T(x,y,y)—for all *x* and for all *y*, *y* is between *x* and *y*—you place in list(passive) -T(a,b,b). If you intend to use just the Tarski 13-axiom system, supplying no additional lemmas, the program will reason from those axioms with the goal of deducing something that contradicts -T(a,b,b), ordinarily, T(x,y,y). With OTTER, you must place some of the Tarski axioms in list(sos) to enable the program to get started. You could place all of the axioms in list(sos). Forward proofs offer a pleasing characteristic of showing you which deduced lemmas were used in the proof in case you wish to seek a proof that avoids any or all of such.

For a second example of seeking a forward proof, an example that illustrates Overbeek's concern, I offer the theorem that says betweenness is symmetric: For all *x*, *y*, and *z*, if *y* is between *x* and *z*, then *y* is between *z* and *x*. The denial of this theorem can be captured with two clauses, T(a,b,c) and -T(c,b,a). (You see this theorem considered with the input file given shortly, but with the use of three different constants.) If I were seeking a forward proof of this symmetry, I would place T(a,b,c) in list(sos) and place -T(c,b,a) in list(passive). In other words, only part of the denial is placed in list(passive). If both unit clauses, corresponding to the denial, were placed in list(passive), a proof would not be forthcoming. OTTER, given this theorem to prove under such conditions, would complete a forward proof by deducing T(c,b,a).

For a third example, taken from group theory, commutativity can be proved for groups in which the square of *x*, for every *x*, is the identity *e*. Typically, to seek a forward proof, I would place in list(sos) f(x,x) = e, where *f* denotes product, and place f(a,b) != f(b,a) in list(passive). OTTER would ordinarily deduce f(x,y) = f(y,x), commutativity.

In contrast, a backward proof reasons from the denial and from the deductions that result, never reasoning from sets of axioms alone. With OTTER, I would place for a backward proof the denial in list(sos) and place all of the axioms in list(usable). Placing all of the axioms in list(usable) and also placing any lemmas that might be used and that are input prevent any forward reasoning from occurring.

And, as you correctly conclude, for a bidirectional proof, you enable the program to reason forward from some or all of the axioms and (input) lemmas and backward from the denial of the theorem. With OTTER, you place the denial of the theorem in list(sos) and place at least one of the axioms (or input lemmas) in that list also. Evidence supports the position that success is most likely—and frequently in less CPU time—if you seek a bidirectional proof. For an explanation regarding the increased likelihood of reaching your goal when you employ a bidirectional search, I note that such proofs typically are of a lower level, where input items are at level 0, and a deduced item is of level one greater than the maximum of the levels of the parents used.

If you would enjoy a research problem that I suspect is not easy to solve, you might study the problem of so-called converting a bidirectional proof to a forward proof, constrained (of course) by the inference rules in use. Too often, with OTTER, I find a bidirectional proof for a specific theorem but cannot find a forward proof. The fault does not rest with OTTER. Especially when paramodulation is used in a proof, you or an automated reasoning program may repeatedly fail to complete a forward proof even though a bidirectional or backward proof is in hand. With paramodulation—and here is, I think, a clue—you usually do not permit paramodulating *from* or *into* variables. The sought-after proof may require the program to relax this constraint, permitting one or both types of paramodulation. (Overbeek says he believes he could write a postprocessor to convert bidirectional proofs into forward proofs, using instantiation heavily; instantiation is almost never offered by an automated reasoning program as an inference rule.)

The approaches to be discussed in this notebook are applicable to any of the three types of proof. You will see that I typically prefer a forward proof, but I do give an example of a simple theorem for which I have obtained only a backward proof at this time. One of the simplest and most direct approaches to proof finding asks you to place all of the problem description in list(sos), for OTTER, so that the program can reason from any subset of the items that are found in the input file. You add no lemmas and give no guidelines or hints to influence the program's search. Of course, no surprise, this simple approach sometimes fails to prove the theorem(s) you intend to prove, which is what did indeed occur with a theorem called Z34 (discussed later in this section). With OTTER, you can seek proofs of more than one theorem at a time, as you see in the following input file that focuses on some early theorems from the Szmielew book (that Beeson and I are closely studying), theorems that are in the main intuitively obvious.

Now, before you visit the first of the various input files I offer, you may well enjoy a list of theorems and definitions pertinent to this study of Tarskian geometry. In this list of theorems, you will find some that I have not yet proved. Indeed, some of the theorems are quite difficult to prove, even if you include (besides a set of axioms) previously proved theorems, included as so-to-speak lemmas. You might nevertheless enjoy proceeding on your own. Note that these use different Skolem symbols from the ones used elsewhere in this notebook.

Theorems and Definitions

```
E(x,y,y,x). % A1 from page 10 of sst
E(x,y,x,y). % Satz 2.1
-E(xa,xb,xc,xd) | E(xc,xd,xa,xb). % Satz 2.2
-E(xa,xb,xc,xd) | E(xb,xa,xc,xd). % Satz 2.4
-E(xa,xb,xc,xd) | -E(xc,xd,xe,xf) | E(xa,xb,xe,xf). %Satz 2.3
-E(xa,xb,xc,xd) | E(xa,xb,xd,xc). % Satz 2.5
E(x,x,y,y). % Satz 2.8
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xb,xa1,xb1) |
-E(xb,xc,xb1,xc1) | E(xa,xc,xa1,xc1). % Satz 2.11
xq = xa | -T(xq,xa,u) | -E(xa,u,xc,xd) | ext(xq,xa,xc,xd) = u. % Satz 2.12
T(x,y,y). % Satz 3.1, also CH3
-E(u,v,x,x) | u=v. % Not one of Szmielew's theorems but we proved it.
-T(xa,xb,xc) | T(xc,xb,xa). % Satz 3.2, also CH4
T(xa,xa,xb). % Satz 2.3, also CH2
```

```
-T(xa,xb,xc) | -T(xb,xa,xc) | xa = xb. % Satz 3.4.
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xb,xc). % Satz 3.51.
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xc,xd). % Satz 3.52.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd). % Satz 3.61.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd). % Satz 3.61.
alpha != beta. %related to Satz 3.14; easily provable if added to sst3h.in.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd). % Satz 3.62.
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd). % Satz 3.71
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd). % Satz 3.72
-IFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xd,xb1,xd1). % Satz 4.2
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xc,xa1,xc1)
| -E(xb,xc,xb1,xc1) | E(xa,xb,xa1,xb1). % Satz 4.3

alpha != beta. % Satz 3.13
beta != gamma.
alpha != gamma.
T(xa,xb,ext(xa,xb,alpha,gamma)). % Satz 3.14, first half
xb != ext(xa,xb,alpha,gamma). % Satz 3.14, second half
% The following many clauses are Definition 4.1
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | T(xa,xb,xc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | T(za,zb,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xa,xc,za,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xb,xc,zb,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xa,xd,za,zd).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xc,xd,zc,zd).
-T(xa,xb,xc) | -T(za,zb,zc) | -E(xa,xc,za,zc) | -E(xb,xc,zb,zc)
| -E(xa,xd,za,zd) | -E(xc,xd,zc,zd) | IFS(xa,xb,xc,xd,za,zb,zc,zd).

% Following 4 are definition 4.4 for n=3
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa1,xa2,xb1,xb2).
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa1,xa3,xb1,xb3).
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa2,xa3,xb2,xb3).
-E(xa1,xa2,xb1,xb2) | -E(xa1,xa3,xb1,xb3) | -E(xa2,xa3,xb2,xb3)
| E3(xa1,xa2,xa3,xb1,xb2,xb3).
% Following three lines are Satz 4.5
-T(xa,xb,xc) | -E(xa,xc,xa1,xc1) | T(xa1,insert(xa,xb,xa1,xc1),xc1).
-T(xa,xb,xc) | -E(xa,xc,xa1,xc1) | E3(xa,xb,xc,xa1,insert(xa,xb,xa1,xc1),xc1).
insert(xa,xb,xa1,xc1) = ext(ext(xc1,xa1,alpha,gamma),xa1,xa,xb).
-E3(x,y,z,u,v,w) | E3(x,z,y,u,w,v). % See sst4q.in, not in Szmielew
-T(xa,xb,xc) | -E3(xa,xb,xc,xa1,xb1,xc1) | T(xa1,xb1,xc1). % Satz 4.6

% following is Definition 4.10
-Col(xa,xb,xc) | T(xa,xb,xc) | T(xb,xc,xa) | T(xc,xa,xb).
Col(xa,xb,xc) | -T(xa,xb,xc).
Col(xa,xb,xc) | -T(xb,xc,xa).
Col(xa,xb,xc) | -T(xc,xa,xb).
% Following are Satz 4.11
-Col(x,y,z) | Col(y,z,x).
-Col(x,y,z) | Col(z,x,y).
-Col(x,y,z) | Col(z,y,x).
-Col(x,y,z) | Col(y,x,z).
-Col(x,y,z) | Col(x,z,y).
Col(x,x,y). % Satz 4.12
```

```
-Col(xa,xb,xc) | - E3(xa,xb,xc,xa1,xb1,xc1) | Col(xa1,xb1,xc1). % Satz 4.13
-Col(xa,xb,xc) | -E(xa,xb,xa1,xb1)
| E3(xa,xb,xc,xa1,xb1,insert5(xa,xb,xc,xa1,xb1)). % Satz 4.14
% following is Definition 4.15
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | Col(xa,xb,xc).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E3(xa,xb,xc,xa1,xb1,xc1).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xa,xd,xa1,xd1).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xd,xb1,xd1).
-Col(xa,xb,xc) | - E3(xa,xb,xc,xa1,xb1,xc1) | - E(xa,xd,xa1,xd1)
| -E(xb,xd,xb1,xd1) | FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | xa = xb | E(xc,xd,xc1,xd1). % Satz 4.16
xa = xb | -Col(xa,xb,xc) | -E(xa,xp,xa,xq) | -E(xb,xp,xb,xq) | E(xc,xp,xc,xq). % Satz 4.17
xa = xb | -Col(xa,xb,xc) | -E(xa,xc,xa,xc1) | -E(xb,xc,xb,xc1) | xc = xc1. % Satz 4.18
-T(xa,xc,xb) | -E(xa,xc,xa,xc1) | -E(xb,xc,xb,xc1) | xc = xc1. % Satz 4.19
% inner Pasch, two clauses.
-T(xa,xp,xc) | -T(xb,xq,xc) | T(xp,ip(xa,xp,xc,xb,xq),xb).
-T(xa,xp,xc) | -T(xb,xq,xc) | T(xq,ip(xa,xp,xc,xb,xq),xa).
```

In part because of the earlier discussion of forward proofs, access to the positive form of the different theorems, found in the just-given list, may prove most useful when diverse denials come into play. Indeed, for example, when I seek a forward proof of a theorem that takes the form of a nonunit clause, you will see that I often place part of the denial in list(passive) and part in list(sos). Such is the case, in the input file I give shortly, when the theorem that asserts betweenness is symmetric comes into focus, CH4 (also called Satz 3.2). With the various input files, I illustrate approaches that often succeed; I do not seek to find the most effective approach for any given theorem. You will find comments concerning the effects of avoiding the use of diverse inference rules; however, I am not studying this aspect in depth in this notebook (but such a study might prove interesting). For example, with the following input file, the avoidance of binary resolution does not prevent the program from finding the five proofs being sought; in contrast, the avoidance of paramodulation prevents OTTER from proving three of the five. The five theorems in focus with this input file are Satz 2.1, 2.2, 3.1 (CH3), 3.2 (CH4), and 3.3 (CH2). (One reader has pointed out that if you want to use this input file, and others in this notebook, you can cut and paste from the pdf, then insert line breaks after every period except those included in the comments. Doing so will avoid having a comment line continue to include all subsequent lines until a blank line is reached. Also, remember to remove the page numbers!)

Input File 1 Illustrating a Simple Approach

```
set(hyper_res).
clear(order_hyper).
set(para_into).
set(para_from).
set(binary_res).
set(ur_res).
set(order_history).
assign(report,5400).
% assign(max_seconds,4).
assign(max_mem,840000).
clear(print_kept).
set(input_sos_first).
set(back_sub).
assign(max_weight,22).
assign(max_distinct_vars,4).
assign(pick_given_ratio,4).
```

```
assign(max_proofs,6).
assign(heat,0).

list(sos).
x = x.
% following 20 are translations of first 20 clauses from ch6 of my 1987 book
% where I focused on Tarski's 12-axiom system
-T(x,y,x) | (x = y).
-T(x,y,u) | -T(y,z,u) | T(x,y,z).
-T(x,y,z) | -T(x,y,u) | (x = y) | T(x,z,u) | T(x,u,z).
E(x,y,y,x).
-E(x,y,z,z) | (x = y).
-E(x,y,z,u) | -E(x,y,v,w) | E(z,u,v,w).
-T(x,v,u) | -T(y,u,z) | T(x,f1(v,x,y,z,u),y).
-T(x,v,u) | -T(y,u,z) | T(z,v,f1(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,z,f2(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,y,f3(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(f2(v6,x,y,z,u),v6,f3(v6,x,y,z,u)).
-E(x1,y1,x2,y2) | -E(y1,z1,y2,z2) | -E(x1,u1,x2,u2) | -E(y1,u1,y2,u2) | -T(x1,y1,z1) | -T(x2,y2,z2) |
 (x1 = y1) | E(z1,u1,z2,u2).
T(x,y,f4(x,y,u,v)).
E(y,f4(x,y,u,v),u,v).
% -T(c1,c2,c3).
% -T(c2,c3,c1).
% -T(c3,c1,c2).
-E(x,u,x,v) | -E(y,u,y,v) | -E(z,u,z,v) | (u = v) | T(x,y,z) | T(y,z,x) | T(z,x,y).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | E(u1,y1,u1,f5(x1,y1,z1,x2,z2,u1)).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | T(x2,f5(x1,y1,z1,x2,z2,u1),z2).
end_of_list.

list(sos).
E(b1,b2,b3,b4).
T(a7,a8,a9).
end_of_list.

list(passive).
-E(a7,a8,a7,a8) | $ANS(CHZ1).
-E(b3,b4,b1,b2) | $ANS(CHZ2).
-T(a1,a1,a2) | $ANS(CH2).
-T(a1,a2,a2) | $ANS(CH3).
-T(a9,a8,a7) | $ANS(CH4).
end_of_list.
```

Note that the list(passive) in the preceding input file includes negations, or part of each, of the five theorems to be proved. (If you are still seeking your own proofs of theorems cited in Section 1, you might wish to pause here to avoid encountering clues.) The first theorem to prove, Z1, asserts the profound fact that the distance from *x* to *y* is equal to the distance from *x* to *y* for all *x* and *y*. The second theorem, Z2, asserts a type of symmetry, namely, if the distance from *x* to *y* is equal to the distance from *u* to *V* for any four points, then the distance from *u* to *V* is equal to the distance from *x* to *y*. For that proof, in contrast to Z1, you must look to an element in the second occurrence of list(sos), as well as consider the cited denial found in list(passive), in order to view the negation of the theorem in its entirety. Again, I am certain you are not impressed, nor would you be impressed by the proofs of these two theorems that you would obtain if you submitted the given input file to OTTER.

The third, fourth, and fifth theorems, as you no doubt have noted, are three theorems cited in the first section of this notebook. Although the proof of the fourth theorem, CH3, requires but two deductions, as is the case for the proof of the second theorem (Z2), the proof OTTER supplied for CH3 relies on axioms concerning equidistance as well as betweenness. And you now begin to taste the complexity and intrigue offered by Tarski with his axiom system. Yes, although I have just provided a clue, I have not yet given proofs, suspecting that you may still be seeking your own proofs. If you run OTTER with the given input file, you will find yet another item of interest in the context of CH3 versus Z1 and Z2. Indeed, whereas hyperresolution was the only inference rule used in the proofs for Z1 and Z2, with CH3 binary resolution and paramodulation were used. For the proof of CH3, axioms involving equidistance, as noted, were indeed used.

In contrast to CH3, whose proof has length 2, the proof of CH2 has length 5. That 5-step proof relies on more input axioms than does the proof for CH3, some focusing on betweenness, as expected, and some focusing on equidistance. Binary resolution is used twice, paramodulation is used twice, and factoring (quite seldom used) is used once. (Factoring is an inference rule that relies on a single premiss or parent.) For CH4—which is expressed as a nonuit clause and hence its denial produces two unit clauses, one of which was placed in list(passive)—the symmetry of betweenness, again a proof of length 5 was obtained, relying on two applications of binary resolution, one of hyperresolution, and two of paramodulation. And, of course, axioms focusing on equidistance play a key role. Interesting at least to me, in that proof, T(x,y,y) was deduced. What a mind had Tarski!

As part of this discussion of a simple approach to proving theorems, now is the time for identifying which of the thirteen Tarski axioms are dependent, a fact conveyed to me by Beeson. (As noted in Section 1, only the second of the two following axioms is dependent, but I did not learn this for quite a while.)

transitivity axiom of betweenness
-T(x,y,u) | -T(y,z,u) | T(x,y,z).
connectivity axiom for betweenness
-T(x,y,z) | -T(x,y,u) | (x = y) | T(x,z,u) | T(x,u,z).

To prove the dependence of either of the axioms is far from easy—especially since the first is not dependent—but I had intended to consider the corresponding tasks later in this notebook. I introduce the topic of axiom dependence in part because a natural run or experiment asks for proofs of the five theorems in focus with an input file that omits the two (thought-to-be) dependent axioms. In this case a shortcut is available, as you see from the following proof of CH4.

A 5-Step Proof of CH4

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Wed Nov 14 10:16:12 2012
The command was "otter". The process ID is 16942.
----> UNIT CONFLICT at 0.11 sec ----> 381 [binary,380.1,25.1] $ANS(CH4).

Length of proof is 5. Level of proof is 4.

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

2 [] -T(x,y,x)|x=y.
6 [] -E(x,y,z,z)|x=y.
8 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
9 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
14 [] T(x,y,f4(x,y,u,v)).
15 [] E(y,f4(x,y,u,v),u,v).
20 [] T(a7,a8,a9).
```

25 [] -T(a9,a8,a7)|$ANS(CH4).
31 [binary,8.3,2.1] -T(x,y,z)| -T(x,z,u)|x=f1(y,x,x,u,z).
34 [binary,15.1,6.1] x=f4(y,x,z,z).
132 [para_from,34.1.2,14.1.3] T(x,y,y).
200 [hyper,31,132,20] a7=f1(a8,a7,a7,a9,a8).
380 [para_from,200.1.2,9.3.3,unit_del,132,20] T(a9,a8,a7).
381 [binary,380.1,25.1] $ANS(CH4).

An inspection of this proof shows that neither of the two dependent axioms is relied upon. Further, such is the case for the other four proofs that are yielded by using the given input file. (Since this notebook is just that, a notebook, I ask you to think with me as I labored under the impression that transitivity of betweenness is dependent, which it is not in the given 13-axiom system of Tarski.)

Before turning to another approach or methodology, one additional element merits discussion. In particular, when I discussed the use of binary resolution with Overbeek, he asserted that it was not needed, not at all. I said that I believed I had proofs obtained from a related Tarski system that appeared to be out of reach if binary resolution was blocked from use. Overbeek scoffed. So, naturally, I at this point in the notebook made a run with the given input file with but one change, namely, the removal (commenting out) of binary resolution. I will almost immediately supply the proof obtained for CH4, but first note that, for example, the last line of the cited 5-step proof (not counted among the five) cites binary resolution in a manner that shows the proof to be a forward proof because of listing an input clause. Often, I do not include such a line because it is typically the empty clause. (Of course, when the theorem has the form of a nonunit clause, as is the case for the symmetry of betweenness, then the last line, before binary resolution with no actual clause being displayed, cannot be the theorem to prove.) For your possible entertainment, I now give the proof that was obtained with the use of binary resolution blocked.

A 6-Step Proof of CH4 That Avoids Binary Resolution

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Wed Nov 14 16:26:38 2012
The command was "otter". The process ID is 20917.
----> UNIT CONFLICT at 0.62 sec ----> 616 [binary,615.1,25.1] $ANS(CH4).

Length of proof is 6. Level of proof is 5.

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

2 [] -T(x,y,x)|x=y.
6 [] -E(x,y,z,z)|x=y.
8 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
9 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
14 [] T(x,y,f4(x,y,u,v)).
15 [] E(y,f4(x,y,u,v),u,v).
20 [] T(a7,a8,a9).
25 [] -T(a9,a8,a7)|$ANS(CH4).
31 [hyper,6,15] x=f4(y,x,z,z).
97 [para_from,31.1.2,14.1.3] T(x,y,y).
115 [hyper,9,20,97] T(a9,a8,f1(a8,a7,x,a9,a9)).
118 [hyper,8,20,97] T(a7,f1(a8,a7,x,a9,a9),x).
581 [hyper,2,118] a7=f1(a8,a7,a7,a9,a9).
615 [para_from,581.1.2,115.1.3] T(a9,a8,a7).
616 [binary,615.1,25.1] $ANS(CH4).

Overbeek has not yet examined this last proof. When and if he does, he will be most pleased. Nevertheless, I hope that you and I, and perhaps Overbeek, will learn later in this notebook that when far more difficult theorems will be in focus, binary resolution is indeed needed; evidence will be given in this context. I strongly suspect that those many, many years ago, in addition to failing to use paramodulation, our automated reasoning group at Argonne National Laboratory did not use binary resolution in our studies of Tarskian geometry. If so, these two failings, and the presence of so many nonunit clauses of a non-Horn nature, in part explain, it seems, why we made so little progress in the mid-1980s and earlier. I have in hand two input files, *A* and *B*, that are identical except for the fact that *A* excludes the use of binary resolution and *B* includes it. With *A*, OTTER finds a proof of interest, a proof of a theorem that I call Z11, which is Satz 2.11 found in the list of theorems and Definitions given earlier; in contrast, with *B*, that proof is not found. On the one hand, you might think that *B* would be the choice because of having access to one additional inference rule, binary resolution. On the other hand, so it appears, its availability may simply drown the program in unneeded conclusions.

Of the numerous paths this notebook might take, some based on current experiments and some on earlier runs, you probably would prefer the path that focuses on a second methodology. The methodology is far from complicated, consisting merely of the inclusion of a so-called fuller axiom set. For example, in the study of group theory, one axiom system relies on associativity of, say, multiplication, the existence of a left identity element e with $ex = x$, and, with respect to e, a left inverse that asserts that, for every x, there exists a y with $yx = e$. McCune often conducted studies in group theory with this axiom system. For such studies, I preferred the axiom system that offers a two-sided identity e, with $ex = xe = x$, and a two-sided inverse, for all x a y exists with $yx = xy = e$. (In answer to your possible curiosity, I believe McCune's preference was based on esthetics; mine, I strongly suspect, was based on my introduction to group theory at the University of Chicago and my doctoral study of groups.) Yes, as many of you know, the system that McCune preferred consists of an independent set of axioms; the one I prefer includes two dependent axioms, that of right identity and that of right inverse. The axiom system with dependent items can be thought of as a fuller axiom system.

You might wonder about the use of the fuller Tarski axiom system versus an axiom system that does not include the two (thought-to-be) dependent items. To explore this question, I conducted two experiments. In the first, I used a Tarski system that avoids the presence of the two cited items and the use of binary resolution. In the second, I took that just-described input file but amended it by including the two cited items. The following proof was found with the second of the two input files, but that proof was not found with the first.

A Proof Relying on Thought-to-Be Dependent Axioms

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Thu Nov 15 07:17:37 2012
The command was "otter". The process ID is 28025.
----> UNIT CONFLICT at 0.02 sec ----> 399 [binary,398.1,49.1] $ANS(CHZ34).

Length of proof is 2. Level of proof is 2.

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

2 [] -T(x,y,x)|x=y.
3 [] -T(x,y,u)| -T(y,z,u)|T(x,y,z).
41 [] T(da,db,dc).
42 [] T(db,da,dc).
49 [] da!=db|$ANS(CHZ34).
263 [hyper,3,41,42] T(da,db,da).
398 [hyper,2,263] da=db.
```

399 [binary,398.1,49.1] $ANS(CHZ34).

The proof uses as an axiom, the transitivity of betweenness, obtaining a 2-step proof. The theorem that is proved, Z34 (Satz 3.4), simply says that if you have three points such that the second is between the first and third and the first is between the second and third, then the first and second points are the same point. Hardly profound! You might immediately deduce that the axiom of transitivity is indeed powerful and, further, conjecture that its proof, from an 11-axiom set of independent axioms, might be difficult to obtain. Indeed, at the moment my attempts to complete a proof of this theorem, from the independent Tarski 11-axiom system, focusing on transitivity have all failed. (Of course, as you learn later in this notebook, none could succeed since transitivity of betweenness is indeed independent of the 11-axiom system.) More important from the viewpoint of this section, you see that the approach of using a fuller set of axioms can offer much. You might also conjecture that the proof of Z34 that is theoretically obtainable from the 11-axiom system would have length greater than 2.

So you have evidence of the value of including axioms that are dependent when you are searching for various proofs. But, as you might immediately suggest, a disadvantage must exist also. Indeed, because of the presence of additional axioms to use, the CPU time and the number of clauses to be retained on the way to some proof will likely increase. The proof of Z11, a deeper theorem, when the two (thought-to-be) dependent axioms were omitted required 5,737 CPU-seconds and the retention of 415,387 clauses; when the two so-called dependent axioms were included in the input, the proof of Z11 required 19,783 CPU-seconds to complete and the retention of 992,081 clauses. As a reminder, in clause form, Z11, as a theorem, is the following.

-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xb,xa1,xb1) | -E(xb,xc,xb1,xc1) | E(xa,xc,xa1,xc1).

So, in addition to esthetic notions, in view of the preceding comments, perhaps McCune preferred to employ sets of axioms free of dependencies for efficiency considerations, of course recognizing the dangers.

Perhaps one of the most natural ways to proceed when seeking a proof, especially if you suspect the proof will be difficult to obtain, is to (in effect) emulate work in published papers and books. Similarly, you might emulate the approach taken by some mathematician you have observed as that person begins the study of a purported theorem. Specifically, such an approach often consists of conjecturing about how a proof might proceed, where the conjecture focuses on a number of lemmas or lesser theorems that, when suitably used together with some additional steps, reaches the desired conclusion. With the conjecture in hand, the conjecturer turns to proving the lemmas and lesser theorems that, in effect, provide an outline of a hoped-for proof.

For example, consider the case in which the goal is to prove the five-point theorem: that is, if you know that *A*3 is between *A*2 and *A*4, and you know that each of *A*2 and *A*4 is between *A*1 and *A*5, then you are to prove that *A*3 is between *A*1 and *A*5. A reasonable conjecture, one made by Overbeek in fact, asserts that the following three lemmas, if proved, would or could play a key role in obtaining the sought-after proof. First, you might prove that for all *x* and for all *y*, *y* is between *x* and *y*. As for this lemma or lesser theorem, you see that the notion of betweenness is not requiring strictly between. Also, I am (in effect) leading the witness in that this lemma is easier to prove than is the next possibly useful lemma. Second, you are to prove that for all *x* and for all *y*, *x* is between *x* and *y*. Each of the cited two lemmas is clearly true, intuitively. So, too, is the third lemma to prove, namely, a lemma about symmetry of betweenness: For all *x*, *y* and *z*, if *y* is between *x* and *z*, then *y* is between *z* and *x*. And yes, the third cited lemma is harder to prove than either of its two predecessors.

In the approach under discussion, once you have proved the three lemmas, various choices exist. You could simply adjoin the statement of each of the three to the set of Tarski axioms; after all, as a reminder, the goal originally was to prove the five-point theorem from the thirteen Tarski axioms. (I do not know whether Overbeek stopped to prove the three lemmas; he did use them, successfully, to seek a proof of the five-point theorem, adjoining to the usable list the statement of each.) Instead, you could adjoin to those axioms all the proof steps of the proofs of the three lemmas, of course observing the warnings about proofs of nonunit lemmas. Since this notebook is about automated reasoning as well as about geometry, you might wish me to pause at this point to focus on the two given choices and provide some detail about what you might do precisely.

Well, one clear choice is to adjoin each of the three statements to the input to be used and place each in list(sos). To ensure that each will receive immediate and close consideration, you would also include the command set(input_sos_first). Or, in the context of the second cited choice, you could amend list(sos) for the next experiment with all the proof steps of the three proofs, of course omitting the last step that simply shows, with binary resolution, that success has occurred with no mention of a formula or equation. You must be careful, especially with regard to adjoining the proof steps. In particular, consider the case in which you are seeking a proof of symmetry of betweenness, and you succeed. The proof might list, say, T(a,b,c), before listing any deduced clause, and it might list -T(c,b,a). For this illustration, I will assume that -T(c,b,a) is placed in list(passive) with the intention of producing a forward proof. The proof you might obtain could have as its last step T(c,b,a), the last step before presenting the contradiction by way of employing binary resolution. This step, taken together with -T(c,b,a), is a contradiction. You would not include T(c,b,a) as a so-called lemma, because of its being a proof step—in fact, the conclusion of a proof of the symmetry of betweenness. The inclusion of that clause clearly is not justified. Instead, you must keep in mind that the theorem to be proved takes the form of a nonunit clause—in the case of symmetry of betweenness, -T(x,y,z) | T(z,y,x)—a clause that, when proved, can be adjoined to the input you plan to use in your attempt to reach your goal, for example, a proof of the five-point theorem.

Overbeek, of course, had in his input file, in addition to the thirteen Tarski axioms, three clauses, the following.

```
T(x,y,y).
T(x,x,y)
-T(x,y,z) | T(z,y,x).
```

To be patently clear, if he had proved symmetry of betweenness, he would not have taken the last line of the proof, whether a forward proof, a backward proof, or a bidirectional proof. If, instead of so-called lemma adjunction you were to rely on resonators, a different story could be told.

Resonators are equations or formulas that do not take on a **true** or a **false** value. They are patterns that are conjectured to be of interest, to be used to direct a program's reasoning. The idea is that if a deduced item matches (where any variable is counted as matching any other available as well as itself and is in effect ignored) a resonator, then that deduced item should be focused on very soon, because it is conjectured to merit attention. Whereas the adjunction of proof steps or the last line of proofs, on the way to reaching the goal, emulates what is done in unaided research, to a far lesser extent the use of resonators does not emulate a person. However, the use of resonators can be astoundingly powerful. As for the warning about lemmas that take the form of nonunit clauses, so-to-speak **if-then** statements, you can use the last lines of their proofs as is, use intermediate proof steps, and use (what can be thought of as) parts of either as resonators.

For example, consider the case in which you have just proved symmetry of betweenness (represented with the following clause), believing it to be an important lemma on the way to proving, say, the five-point theorem.

```
-T(x,y,z) | T(z,y,x).
```

You could include as resonators either of the literals of this clause without the worry of soundness. Of course, you could follow the resonator approach and also, at the same time, follow the (preceding) lemma adjunction approach. So, for greater clarity, you could be told about a possible proof, or you could guess at how a proof might proceed; and then you could try to prove as lemmas or lesser theorems the elements of the so-called outline. In succeeding runs or experiments or attempts, you could, often profitably (ignoring the last line that cites the binary resolution for unit conflict), use the results of success with each lemma and/or the proof steps thereof as resonators and as adjoined lemmas. You must be careful, as discussed, regarding the form taken by lemmas and lesser theorems, when proved, keeping in mind those that take the form of unit clauses and those that take the form on nonunit clauses.

For a fuller example, in the context of resonators, you could, and probably would, take all the proof steps of the three proofs of the three cited lemmas, to be used as resonators, assign to each a small value, and place this set in weight_list(pick_and_purge). Indeed, let *S* be a proof step in one of the three proofs. You would then find, in the amended input file, something like the following.

weight(S,2).

Of course, with the just-described approaches you run the risk that your outline for a proof is not useful, that the lemmas and lesser theorems you prove provide little assistance in completing the sought-after proof. Further, with (possibly) many additional items to reason from, items that turn out to be of little or no use, your program can focus on each of them and deduce so much irrelevant information that it virtually drowns. Similarly, when you include resonators that prove to be of essentially no value, their use can lead the program down one path after another, paths that are fruitless. Indeed, with small values, high priorities for directing the reasoning, assigned to such resonators, equations or formulas that match a resonator of no value will be chosen quickly as the focus of attention and lead the program away from key information.

Instead of considering the methodologies just discussed, you might wish to simply dive right in and submit to OTTER the thirteen Tarski axioms and the theorem to be proved. What would happen, you ask, if you instructed OTTER to prove the five-point theorem without supplying any lemmas or lesser theorems, in other words, without conjecturing about the nature of the proof? Well, I did just that, and OTTER returned the following.

A Proof of the Five-Point Theorem Based on Tarski's Thirteen Axioms Alone

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Wed Nov 14 09:33:18 2012
The command was "otter". The process ID is 16380.
----> UNIT CONFLICT at 280.70 sec ----> 130710 [binary,130709.1,25.1] $ANS(CH1).

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

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

2 [] -T(x,y,x)|x=y.
4 [] -E(x,y,z,z)|x=y.
6 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
7 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
12 [] T(x,y,f4(x,y,u,v)).
13 [] E(y,f4(x,y,u,v),u,v).
18 [] T(a2,a3,a4).
19 [] T(a1,a2,a5).
20 [] T(a1,a4,a5).
25 [] -T(a1,a3,a5)|$ANS(CH1).
37 [binary,6.3,2.1] -T(x,y,z)| -T(x,z,u)|x=f1(y,x,x,u,z).
39 [binary,13.1,4.1] x=f4(y,x,z,z).
95 [binary,20.1,7.2] -T(x,y,a4)|T(a5,y,f1(y,x,a1,a5,a4)).
189 [para_from,39.1.2,12.1.3] T(x,y,y).
195 [binary,189.1,7.2] -T(x,y,z)|T(z,y,f1(y,x,u,z,z)).
315 [hyper,37,189,18] a2=f1(a3,a2,a2,a4,a3).
328 [para_from,37.3.2,7.3.3,factor_simp,factor_simp] -T(x,y,z)| -T(x,z,u)|T(u,y,x).
693 [para_from,315.1.2,7.3.3,unit_del,189,18] T(a4,a3,a2).
739 [hyper,7,693,19] T(a5,a3,f1(a3,a4,a1,a5,a2)).
746 [hyper,6,693,19] T(a4,f1(a3,a4,a1,a5,a2),a1).
13908 [para_into,95.2.3,37.3.2,unit_del,20,factor_simp] -T(a1,x,a4)|T(a5,x,a1).
50002 [para_into,195.2.3,37.3.2,unit_del,189,factor_simp] -T(x,y,z)|T(z,y,x).
50295 [binary,50002.2,13908.1] -T(a4,x,a1)|T(a5,x,a1).
```

```
56288 [binary,50295.1,746.1] T(a5,f1(a3,a4,a1,a5,a2),a1).
130709 [hyper,328,739,56288] T(a1,a3,a5).
130710 [binary,130709.1,25.1] $ANS(CH1).
```

You quickly deduce from the given proof that rather than simply supplying appropriate clauses, I did make some decisions. In particular, I chose which inference rules to use. Indeed, binary resolution was used six times to deduce appropriate conclusions; hyperresolution was used four times; paramodulation was used five times. Of course, I am not counting the citing of binary resolution in the line that establishes UNIT CONFLICT or in the last line that presents no clause other than the empty clause. More information concerning my choices can be gleaned from a further examination of the given proof. First, since clause (25), which is part of the denial of the conclusion of the five-point theorem, was not used as a parent from any deduction of a new conclusion, you might correctly surmise that that clause was placed in list(passive), blocking its use in making deductions. Second, since clause (25) was cited in the UNIT CONFLICT line and it appears before any deduced item, you can correctly conclude that the given proof is a forward proof. The presence of various applications of paramodulation and of binary resolution causes me to strongly hazard that, those many, many years ago, our group used neither in our attempts with Tarski. Also, the abundance of many nonunit deduced clauses suggests to me that, long ago, nonunit clauses often presented a problem for us.

Especially if you wish to glean much about automated reasoning, and perhaps about proof in general, you might wonder about the need for all four cited inference rules. After all, Overbeek (in effect) asserts that binary resolution best not be used. You might also wonder about the effect of placing clause (25) in list(sos) to seek a bidirectional proof. Although you cannot tell from the given proof, I note that all but the denial clause were placed in list(sos); therefore, you might wonder what would occur if list(usable) were used. Then you might be curious about the results if some obvious lemmas were included in the input, rather than relying solely on the Tarski system. Further, you might ask about relying on the Tarski system with the two thought-to-be dependent axioms removed. Answers to questions of this type might eventually lead to important insights about the geometry, about Tarski, and about automated reasoning, perhaps in the context of strategy. Let us supply some answers, in part based on experiments.

I begin with the item focusing on the need for all four inference rules: binary resolution, hyperresolution, paramodulation, and UR-resolution. Because equality is present in the Tarski axioms, that suggests that paramodulation merits use; because nonunit clauses containing one positive literal are present, that naturally suggests hyperresolution should be used; because unit clauses are often so valuable, UR-resolution is suggested; and the obvious candidate for removal is binary resolution. By commenting out binary resolution from the input file that was used to obtain the earlier-cited 15-step proof, you can see what will happen with the following input file.

A Second Input File, a File Proving the Five-Point Theorem

```
set(hyper_res).
clear(order_hyper).
set(para_into).
set(para_from).
set(binary_res).
set(ur_res).
set(order_history).
assign(report,5400).
assign(max_mem,840000).
clear(print_kept).
set(input_sos_first).
set(back_sub).
assign(max_weight,22).
assign(max_distinct_vars,4).
```

```
assign(pick_given_ratio,4).
assign(max_proofs,6).
assign(heat,0).

list(sos).
x = x.
% following 20 are translations of first 20 from ch6 of book2 for Tarski
-T(x,y,x) | (x = y).
% -T(x,y,u) | -T(y,z,u) | T(x,y,z).
% -T(x,y,z) | -T(x,y,u) | (x = y) | T(x,z,u) | T(x,u,z).
E(x,y,y,x).
-E(x,y,z,z) | (x = y).
-E(x,y,z,u) | -E(x,y,v,w) | E(z,u,v,w).
-T(x,v,u) | -T(y,u,z) | T(x,f1(v,x,y,z,u),y).
-T(x,v,u) | -T(y,u,z) | T(z,v,f1(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,z,f2(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,y,f3(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(f2(v6,x,y,z,u),v6,f3(v6,x,y,z,u)).
-E(x1,y1,x2,y2) | -E(y1,z1,y2,z2) | -E(x1,u1,x2,u2) | -E(y1,u1,y2,u2) |
 -T(x1,y1,z1) | -T(x2,y2,z2) | (x1 = y1) | E(z1,u1,z2,u2).
T(x,y,f4(x,y,u,v)).
E(y,f4(x,y,u,v),u,v).
-E(x,u,x,v) | -E(y,u,y,v) | -E(z,u,z,v) | (u = v) | T(x,y,z) | T(y,z,x) | T(z,x,y).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | E(u1,y1,u1,f5(x1,y1,z1,x2,z2,u1)).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | T(x2,f5(x1,y1,z1,x2,z2,u1),z2).
end_of_list.

list(sos).
E(b1,b2,b3,b4).
T(a2,a3,a4).
T(a1,a2,a5).
T(a1,a4,a5).
T(a7,a8,a9).
T(A2,a6,a4).
end_of_list.

list(passive).
-E(a7,a8,a7,a8) | $ANS(CHZ1).
-E(b3,b4,b1,b2) | $ANS(CHZ2).
-T(a1,a3,a5) | $ANS(CH1).
-T(a5,a3,a1) | $ANS(CH11).
-T(a1,a1,a2) | $ANS(CH2).
-T(a1,a2,a2) | $ANS(CH3).
-T(a9,a8,a7) | $ANS(CH4).
-T(a2,a3,a6) | $ANS(CH5A).
-T(a2,a6,a3) | $ANS(CH5B).
end_of_list.
```

A glance at the input file shows that two of Tarski's axioms, that for transitivity of betweenness and that for connectivity of betweenness, have been commented out (by using a percent sign). Certain subquestions, so to speak, arise. You could ask about the results to be obtained if binary resolution is not used. You could ask about the results that would be obtained with an input file that, in addition to avoiding the use of binary resolution, permitted the use of the two clauses commented out in the just-given input file.

You could ask about the results of permitting the use of binary resolution and the use of the two commented-out clauses. I conducted these very experiments. They complete an obvious set of four, where the fourth involves blocking the use of two thought-to-be dependent clauses while permitting the use of binary resolution.

After many hours, only one of the three experiments produced a proof of the five-point theorem. The experiment that succeeded relied on binary resolution and had access to the clauses for transitivity of betweenness and connectivity of betweenness. I will shortly discuss the proofs that were found, each early. But first, I note that as I waited impatiently, not long after submitting the three experiments, I examined what was happening. And here is what I found (after I first label the experiments for possible convenience), in the context of proofs relevant to the use of lemmas, if that is a choice to be considered.

The experiment described earlier that yielded the 15-step proof is called EXP1; it relies on the use of binary resolution and excludes access to the clauses for both transitivity and connectivity of betweenness. EXP2 blocks the use of binary resolution and is otherwise identical to EXP1. EXP3 blocks binary resolution, but it permits the use of both transitivity and connectivity of betweenness. EXP4 allows the use of binary resolution and also allows the program to reason from both transitivity and connectivity of betweenness. Lemma 1 asserts that T(x,y,y); Lemma 2 asserts that T(x,x,y); Lemma 3 asserts the symmetry of betweenness.

My typical approach to experimentation is to include, in addition to the main goal, other goals that are conjectured to be more easily reached. These lesser goals, if and when proved, often provide useful clauses for later experiments, especially when the main goal has not yet been reached. (By focusing on related experiments and the data that results, you can glean some understanding and appreciation of choices that can be made.) In all four experiments, Lemmas 1, 2, and 3 (as well as others of lesser interest) were included as targets. In all four experiments, the three lemmas were proved, but not in an identical manner. A closer examination of EXP1 yields the following interesting results.

In the 5-step proof of Lemma 2, you find (as will be seen shortly) the deduction of Lemma 1, T(x,y,y).

A 5-Step Proof of Lemma 2 with Experiment 1

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Wed Nov 14 09:33:18 2012
The command was "otter". The process ID is 16380.
----> UNIT CONFLICT at 0.05 sec ----> 330 [binary,329.1,27.1] $ANS(CH2).

Length of proof is 5. Level of proof is 3.

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

2 [] -T(x,y,x)|x=y.
4 [] -E(x,y,z,z)|x=y.
6 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
7 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
12 [] T(x,y,f4(x,y,u,v)).
13 [] E(y,f4(x,y,u,v),u,v).
27 [] -T(a1,a1,a2)|$ANS(CH2).
37 [binary,6.3,2.1] -T(x,y,z)| -T(x,z,u)|x=f1(y,x,x,u,z).
39 [binary,13.1,4.1] x=f4(y,x,z,z).
189 [para_from,39.1.2,12.1.3] T(x,y,y).
328 [para_from,37.3.2,7.3.3,factor_simp,factor_simp] -T(x,y,z)| -T(x,z,u)|T(u,y,x).
329 [factor,328.1.2,unit_del,189] T(x,x,y).
```

And here is a proof of Lemma 1 that was found in Experiment 1 before the 5-step proof just given, a proof that, when compared with the 5-step proof, offers perhaps surprising similarity.

A 2-Step Proof of Lemma 1 from Experiment 1

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Wed Nov 14 09:33:18 2012
The command was "otter". The process ID is 16380.
----> UNIT CONFLICT at 0.03 sec ----> 190 [binary,189.1,28.1] $ANS(CH3).

Length of proof is 2. Level of proof is 2.

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

4 [] -E(x,y,z,z)|x=y.
12 [] T(x,y,f4(x,y,u,v)).
13 [] E(y,f4(x,y,u,v),u,v).
28 [] -T(a1,a2,a2)|$ANS(CH3).
39 [binary,13.1,4.1] x=f4(y,x,z,z).
189 [para_from,39.1.2,12.1.3] T(x,y,y).
```

Then, I offer you a proof of Lemma 3 from Experiment 1, a proof that so-to-speak borrows from the proofs it found while proving Lemmas 1 and 2.

A 5-Step Proof of Lemma 3 from Experiment 1

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Wed Nov 14 09:33:18 2012
The command was "otter". The process ID is 16380.
----> UNIT CONFLICT at 0.08 sec ----> 485 [binary,484.1,29.1] $ANS(CH4).

Length of proof is 5. Level of proof is 4.

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

2 [] -T(x,y,x)|x=y.
4 [] -E(x,y,z,z)|x=y.
6 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
7 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
12 [] T(x,y,f4(x,y,u,v)).
13 [] E(y,f4(x,y,u,v),u,v).
21 [] T(a7,a8,a9).
29 [] -T(a9,a8,a7)|$ANS(CH4).
37 [binary,6.3,2.1] -T(x,y,z)| -T(x,z,u)|x=f1(y,x,x,u,z).
39 [binary,13.1,4.1] x=f4(y,x,z,z).
189 [para_from,39.1.2,12.1.3] T(x,y,y).
312 [hyper,37,189,21] a7=f1(a8,a7,a7,a9,a8).
484 [para_from,312.1.2,7.3.3,unit_del,189,21] T(a9,a8,a7).
```

As noted before and exemplified by the proof just presented, the last line of a proof cannot necessarily simply be adjoined as a lemma in future investigations. In the case of Lemma 3, if your program supplies a proof of the type just viewed, you can adjoin a clause for symmetry of betweenness (as a lemma) in further experimentation. After all, the lemma or lesser theorem that was proved is a nonunit clause whose negation or denial had you place two unit clauses in the input file designed to prove Lemma 3.

If you are new to the use of OTTER and perhaps to the use of any automated reasoning program, you have strong evidence, if you review the results of Experiment 1, of the patience you must sometimes exercise. In particular, the five-point theorem (the main goal) was also proved in the experiment, but only after a (at least subjectively) long wait. Indeed, as you see from the data, apparently—and that is the correct term—nothing was occurring for more than 4 CPU-minutes. Of course, OTTER was continuously drawing conclusions. Sometimes, days of CPU time elapse between proofs that are found in a single run.

Next in order is a glance at the other three experiments, beginning with EXP2, which differs from EXP1 only in the blocking of the use of binary resolution.

As with EXP1, EXP2 quickly yielded proofs of Lemmas 1, 2, and 3, in that order. The proofs were somewhat different from those cited earlier from EXP1, since the use of binary resolution was blocked. Again, as you will shortly see, the proof of Lemma 2 contains a proof of Lemma 1.

A 6-Step Proof of Lemma 2 from Experiment 2

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Sat Nov 24 14:28:22 2012
The command was "otter". The process ID is 28354.
----> UNIT CONFLICT at 0.60 sec ----> 964 [binary,963.1,27.1] $ANS(CH2).

Length of proof is 6. Level of proof is 5.

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

2 [] -T(x,y,x)|x=y.
4 [] -E(x,y,z,z)|x=y.
6 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
7 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
12 [] T(x,y,f4(x,y,u,v)).
13 [] E(y,f4(x,y,u,v),u,v).
27 [] -T(a1,a1,a2)|$ANS(CH2).
37 [hyper,4,13] x=f4(y,x,z,z).
138 [para_from,37.1.2,12.1.3] T(x,y,y).
179 [hyper,7,138,138] T(x,x,f1(x,y,z,x,x)).
190 [hyper,6,138,138] T(x,f1(y,x,z,y,y),z).
960 [hyper,2,190] x=f1(y,x,x,y,y).
963 [para_from,960.1.2,179.1.3] T(x,x,y).
```

The initial segment of this proof closely resembles the initial segment of the 5-step proof of Lemma 2 obtained with EXP1 with the exception of stating that hyperresolution was used in place of binary resolution. Indeed, the two clauses, parents, that are used for the first step yield the same conclusion whether hyperesolution or binary resolution is used. Whereas hyperresolution is used in this 6-step proof, it is not used, in the beginning, in the 5-step proof yielded with EXP1.

As for Lemma 3, instead of a 5-step proof, EXP2 yielded a 6-step proof, the following.

A 6-Step Proof of Lemma 3 from Experiment 2

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
```

Sat Nov 24 14:28:22 2012
The command was "otter". The process ID is 28354.
----> UNIT CONFLICT at 0.82 sec ----> 1414 [binary,1413.1,29.1] $ANS(CH4).

Length of proof is 6. Level of proof is 5.

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

2 [] -T(x,y,x)|x=y.
4 [] -E(x,y,z,z)|x=y.
6 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
7 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
12 [] T(x,y,f4(x,y,u,v)).
13 [] E(y,f4(x,y,u,v),u,v).
21 [] T(a7,a8,a9).
29 [] -T(a9,a8,a7)|$ANS(CH4).
37 [hyper,4,13] x=f4(y,x,z,z).
138 [para_from,37.1.2,12.1.3] T(x,y,y).
181 [hyper,7,21,138] T(a9,a8,f1(a8,a7,x,a9,a9)).
192 [hyper,6,21,138] T(a7,f1(a8,a7,x,a9,a9),x).
1405 [hyper,2,192] a7=f1(a8,a7,a7,a9,a9).
1413 [para_from,1405.1.2,181.1.3] T(a9,a8,a7).

You see that the avoidance of using binary resolution, at least for proofs of Lemmas 1, 2, and 3, has relatively little effect, almost none in CPU time. However, whereas EXP1 yielded a proof of the five-point theorem, EXP2 did not, even after more than 20 CPU-hours when the program ran out of the alloted memory, slightly less than 1,000 megabytes. Does this data suggest that the use of binary resolution is crucial? Perhaps, as in EXP3 the addition of two input clauses, for transitivity and connectivity of betweenness, will enable the program to prove the five-point theorem.

In EXP3, as noted, again binary resolution was blocked, but two clauses not allowed to participate in EXP2 were allowed to participate in EXP3. In particular, as you have predicted, the two correspond to transitivity and connectivity of betweenness. I thought that, just possibly, EXP3 would produce results a bit unlike those of EXP2. Such was not the case. Indeed, ignoring the clause numbers, all is the same. The presence of the additional clauses did not result in a proof of the five-point theorem either. So, on the surface and to the possible annoyance of Overbeek, evidence was accruing to support the position that binary resolution is needed if a proof of the five-point theorem is to be found without including various lemmas.

In EXP4, the program not only was permitted the use of binary resolution but also was given access to those two elusive clauses, transitivity and connectivity of betweenness. The program quickly proved Lemmas 1, 2, and 3; and the proofs looked like those found in EXP1. With regard to the five-point theorem, however, in contrast to EXP1, you will see in the following proof that far more computer time was needed. In other words, if you had been sitting near a terminal and watching for a proof of the five-point theorem, having in mind the few CPU-minutes required in EXP1, you would most likely have experienced much impatience.

A 13-Step Proof of the Five-Point Theorem from Experiment 4

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Sat Nov 24 15:54:36 2012
The command was "otter". The process ID is 29017.
----> UNIT CONFLICT at 23457.24 sec ----> 617695 [binary,617694.1,27.1] $ANS(CH1).

Length of proof is 13. Level of proof is 7.

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

2 [] -T(x,y,x)|x=y.
6 [] -E(x,y,z,z)|x=y.
8 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
9 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
14 [] T(x,y,f4(x,y,u,v)).
15 [] E(y,f4(x,y,u,v),u,v).
20 [] T(a2,a3,a4).
21 [] T(a1,a2,a5).
22 [] T(a1,a4,a5).
27 [] -T(a1,a3,a5)|$ANS(CH1).
40 [binary,8.3,2.1] -T(x,y,z)| -T(x,z,u)|x=f1(y,x,x,u,z).
43 [binary,15.1,6.1] x=f4(y,x,z,z).
83 [binary,21.1,9.2] -T(x,y,a2)|T(a5,y,f1(y,x,a1,a5,a2)).
124 [hyper,9,20,22] T(a5,a3,f1(a3,a2,a1,a5,a4)).
126 [hyper,8,20,22] T(a2,f1(a3,a2,a1,a5,a4),a1).
285 [para_from,43.1.2,14.1.3] T(x,y,y).
292 [binary,285.1,9.2] -T(x,y,z)|T(z,y,f1(y,x,u,z,z)).
430 [para_from,40.3.2,9.3.3,factor_simp,factor_simp] -T(x,y,z)| -T(x,z,u)|T(u,y,x).
5855 [para_into,83.2.3,40.3.2,unit_del,21,factor_simp] -T(a1,x,a2)|T(a5,x,a1).
61112 [para_into,292.2.3,40.3.2,unit_del,285,factor_simp] -T(x,y,z)|T(z,y,x).
61533 [binary,61112.2,5855.1] -T(a2,x,a1)|T(a5,x,a1).
109892 [binary,61533.1,126.1] T(a5,f1(a3,a2,a1,a5,a4),a1).
617694 [hyper,430,124,109892] T(a1,a3,a5).

Of course, you would expect, as I did, that the two elusive clauses play a role in this 13-step proof. No, they do not. Why, then, is the proof so much different from the 15-step proof found in EXP1? I cannot say anything of substance. I leave the analysis to you.

And now, perhaps, you have concluded that binary resolution not only is useful but, perhaps, is needed to prove the five-point theorem. But wait: Overbeek's patience has run out. Indeed, he almost shouts, directing the researcher to the use of lemmas, for example, Lemmas 1, 2, and 3. Well, rather than conducting the experiment, perhaps called EXP5, I simply browsed among my numerous experiments for that which he would approve of. And I found an input file whose max_weight was assigned the value of 15 rather than 22 as in preceding data; and its output file, I found, does support Overbeek's conviction that binary resolution is not needed for the proof of the five-point theorem. You will see, in the following proof, taken from the experiment of weeks ago, that binary resolution was not used. To my surprise, however, neither Lemma 1 nor Lemma 2 was used in the proof; Lemma 3 was used.

A 10-Step Proof of the Five-Point Theorem without the Use of Binary Resolution

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on steamroller,
Sat Nov 17 10:43:06 2012
The command was "otter". The process ID is 32251.
----> UNIT CONFLICT at 32868.65 sec ----> 297169 [binary,297168.1,28.1] $ANS(CH1).

Length of proof is 10. Level of proof is 8.

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

1 [] -T(x,y,x)|x=y.
7 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
8 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
23 [] -T(x,y,z)|T(z,y,x).
24 [] T(a2,a3,a4).
25 [] T(a1,a2,a5).
26 [] T(a1,a4,a5).
28 [] -T(a1,a3,a5)|$ANS(CH1).
81 [hyper,26,8,24] T(a5,a3,f1(a3,a2,a1,a5,a4)).
86 [hyper,26,7,24] T(a2,f1(a3,a2,a1,a5,a4),a1).
602 [hyper,86,23] T(a1,f1(a3,a2,a1,a5,a4),a2).
5246 [hyper,602,7,25] T(a1,f1(f1(a3,a2,a1,a5,a4),a1,a1,a5,a2),a1).
101468 [hyper,5246,1] a1=f1(f1(a3,a2,a1,a5,a4),a1,a1,a5,a2).
101473 [para_from,101468.1.2,8.3.3,unit_del,602,25] T(a5,f1(a3,a2,a1,a5,a4),a1).
101476 [hyper,101473,8,81] T(a1,a3,f1(a3,a5,a5,a1,f1(a3,a2,a1,a5,a4))).
101479 [hyper,101473,7,81] T(a5,f1(a3,a5,a5,a1,f1(a3,a2,a1,a5,a4)),a5).
297163 [hyper,101479,1] a5=f1(a3,a5,a5,a1,f1(a3,a2,a1,a5,a4)).
297168 [para_from,297163.1.2,101476.1.3] T(a1,a3,a5).

You now have in hand evidence of how sturdy OTTER is. How piquant that Lemma 3, symmetry of betweenness, is used as a parent in but one deduction! From a global viewpoint, you have a charming example of the methodology for finding proofs that is based on lemma adjunction. This 10-step proof is a forward proof; but, as seen before, its last line does not provide you with a lemma to use in further studies. Part of the denial of the five-point theorem is used to draw conclusions. I leave to you the investigation that focuses on variants of Experiments 2 and 3, for example, where the research focuses on assigning various values to the diverse parameters offered by McCune's program.

Other experiments with the five-point theorem offer food for thought. For a first thought, both the 15-step and the 13-step proof of the five-point theorem are forward proofs, neither reasoning from the denial of the conclusion. A natural experiment has you seek a bidirectional proof; after all, as observed much earlier in this notebook, bidirectional proofs often complete in less CPU time, sometimes far less. A powerful example of what can occur when seeking a bidirectional proof was found by including Lemmas 1, 2, and 3 and a lemma or theorem that Overbeek used in his study of the five-point theorem. In particular, he included, as I did in the following, his four-point theorem, discussed early in this notebook: If a3 and a6 are, respectively, between a2 and a4, then either a3 is between a2 and a6 or a6 is between a2 and a3. Yes, your objection is valid: as yet I have not seen from anybody a proof of the four-point theorem, proved by using the Tarski 13-axiom system or smaller. Nevertheless, I now present the first bidirectional proof (of the five-point theorem), of the ten proofs it found, as an illustration of what can occur.

A 42-Step Bidirectional Proof of the Five-Point Theorem

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Thu Nov 29 15:54:09 2012
The command was "otter". The process ID is 28156.
----> UNIT CONFLICT at 28.31 sec ----> 47093 [binary,47092.1,46664.1] $ANS(CH1).

Length of proof is 42. Level of proof is 11.

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

1 [] -T(x,y,x)|x=y.
2 [] -T(x,y,u)| -T(y,z,u)|T(x,y,z).
3 [] -T(x,y,z)| -T(x,y,u)|x=y|T(x,z,u)|T(x,u,z).

7 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
8 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
21 [] T(x,y,y).
22 [] T(x,x,y).
23 [] -T(x,y,z)|T(z,y,x).
24 [] -T(x,u,z)| -T(x,v,z)|T(x,u,v)|T(x,v,u).
25 [] T(a2,a3,a4).
26 [] T(a1,a2,a5).
27 [] T(a1,a4,a5).
28 [] -T(a1,a3,a5)|$ANS(CH1).
33 [hyper,25,23] T(a4,a3,a2).
52 [hyper,26,23] T(a5,a2,a1).
66 [hyper,26,7,22] T(x,f1(x,x,a1,a5,a2),a1).
72 [hyper,27,23] T(a5,a4,a1).
82 [hyper,27,8,25] T(a5,a3,f1(a3,a2,a1,a5,a4)).
87 [hyper,27,7,25] T(a2,f1(a3,a2,a1,a5,a4),a1).
88 [hyper,27,7,22] T(x,f1(x,x,a1,a5,a4),a1).
93 [ur,28,23] $ANS(CH1)| -T(a5,a3,a1).
130 [hyper,52,8,33] T(a1,a3,f1(a3,a4,a5,a1,a2)).
135 [hyper,52,7,33] T(a4,f1(a3,a4,a5,a1,a2),a5).
141 [hyper,72,24,52] T(a5,a2,a4)|T(a5,a4,a2).
153 [hyper,72,8,25] T(a1,a3,f1(a3,a2,a5,a1,a4)).
158 [hyper,72,7,25] T(a2,f1(a3,a2,a5,a1,a4),a5).
228 [hyper,141,2,25] T(a5,a4,a2)|T(a5,a2,a3).
426 [hyper,228,2,33] T(a5,a2,a3)|T(a5,a4,a3).
820 [hyper,426,23] T(a5,a4,a3)|T(a3,a2,a5).
2697 [hyper,66,2,21] T(x,a1,f1(a1,a1,a1,a5,a2)).
2698 [hyper,66,1] a1=f1(a1,a1,a1,a5,a2).
2798 [para_from,2698.1.1,93.2.3] $ANS(CH1)| -T(a5,a3,f1(a1,a1,a1,a5,a2)).
2808 [para_from,2698.1.1,52.1.3] T(a5,a2,f1(a1,a1,a1,a5,a2)).
3140 [ur,82,2,93] -T(a3,a1,f1(a3,a2,a1,a5,a4))|$ANS(CH1).
3186 [hyper,87,23] T(a1,f1(a3,a2,a1,a5,a4),a2).
3265 [hyper,88,1] a1=f1(a1,a1,a1,a5,a4).
3386 [para_from,3265.1.1,93.2.3] $ANS(CH1)| -T(a5,a3,f1(a1,a1,a1,a5,a4)).
3396 [para_from,3265.1.1,72.1.3] T(a5,a4,f1(a1,a1,a1,a5,a4)).
4597 [ur,130,2,28] -T(a3,a5,f1(a3,a4,a5,a1,a2))|$ANS(CH1).
4724 [hyper,135,23] T(a5,f1(a3,a4,a5,a1,a2),a4).
7586 [hyper,2697,1] f1(a1,a1,a1,a5,a2)=a1.
9173 [ur,3186,2,3140] -T(a3,a1,a2)|$ANS(CH1).
9210 [ur,9173,2,26] $ANS(CH1)| -T(a3,a1,a5).
9241 [ur,9210,23] $ANS(CH1)| -T(a5,a1,a3).
9307 [para_into,9241.2.2,7586.1.2] $ANS(CH1)| -T(a5,f1(a1,a1,a1,a5,a2),a3).
9310 [para_into,9241.2.2,3265.1.1] $ANS(CH1)| -T(a5,f1(a1,a1,a1,a5,a4),a3).
14411 [hyper,3396,3,820,unit_del,3386,9310] a5=a4|T(a3,a2,a5)|$ANS(CH1).
14413 [hyper,3396,3,426,unit_del,3386,9310] a5=a4|T(a5,a2,a3)|$ANS(CH1).
16727 [hyper,14413,3,2808,unit_del,9307,2798] a5=a4|$ANS(CH1)|a5=a2.
17799 [para_from,16727.2.1,28.1.3] -T(a1,a3,a2)|$ANS(CH1)|a5=a4.
44934 [ur,4597,2,21] $ANS(CH1)| -T(a5,f1(a3,a4,a5,a1,a2),a5).
45902 [para_into,4724.1.3,17799.2.2,unit_del,44934] -T(a1,a3,a2)|$ANS(CH1).
45912 [para_into,4724.1.3,14411.1.2,unit_del,44934] T(a3,a2,a5)|$ANS(CH1).
46664 [ur,45902,2,153] $ANS(CH1)| -T(a3,a2,f1(a3,a2,a5,a1,a4)).
47092 [hyper,45912,2,158] $ANS(CH1)|T(a3,a2,f1(a3,a2,a5,a1,a4)).
47093 [binary,47092.1,46664.1] $ANS(CH1).

This proof, if shown to Overbeek, would please him, in part. Indeed, binary resolution is not used to deduce any steps, appearing only in the UNIT CONFLICT step and in the final step that asserts (in effect) a deduction of the empty clause. Of the forty-two steps, eight were with UR-resolution, twenty-five with hyperreslution, and nine with paramodulation.

To obtain the proof, I used the following input file.

Input File for the 42-Step Proof of the Five-Point Theorem

```
assign(max_weight,15).
assign(max_proofs,10).
set(hyper_res).
clear(order_hyper).
set(para_into).
set(para_from).
set(ur_res).
%set(binary_res).
set(unit_deletion).

list(usable).
%  following 20 are translations of first 20 from ch6 of book2 for Tarski
-T(x,y,x) | (x = y).
-T(x,y,u) | -T(y,z,u) | T(x,y,z).
-T(x,y,z) | -T(x,y,u) | (x = y) | T(x,z,u) | T(x,u,z).
E(x,y,y,x).
-E(x,y,z,z) | (x = y).
-E(x,y,z,u) | -E(x,y,v,w) | E(z,u,v,w).
-T(x,v,u) | -T(y,u,z) | T(x,f1(v,x,y,z,u),y).
-T(x,v,u) | -T(y,u,z) | T(z,v,f1(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,z,f2(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,y,f3(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(f2(v6,x,y,z,u),v6,f3(v6,x,y,z,u)).
-E(x1,y1,x2,y2) | -E(y1,z1,y2,z2) | -E(x1,u1,x2,u2) | -E(y1,u1,y2,u2) | -T(x1,y1,z1) | -T(x2,y2,z2) | (x1 = y1) | E(z1,u1,z2,u2).
T(x,y,f4(x,y,u,v)).
E(y,f4(x,y,u,v),u,v).
-T(c1,c2,c3).
-T(c2,c3,c1).
-T(c3,c1,c2).
-E(x,u,x,v) | -E(y,u,y,v) | -E(z,u,z,v) | (u = v) | T(x,y,z) | T(y,z,x) | T(z,x,y).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | E(u1,y1,u1,f5(x1,y1,z1,x2,z2,u1)).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | T(x2,f5(x1,y1,z1,x2,z2,u1),z2).
T(x,y,y).
T(x,x,y).
-T(x,y,z) | T(z,y,x).
-T(x,u,z) | -T(x,v,z) | T(x,u,v) | T(x,v,u).
%  -T(x,y1,z) | -T(x,y2,z) | T(x,y1,y2) | T(x,y2,y1).
end_of_list.

list(sos).
T(a2,a3,a4).
T(a1,a2,a5).
T(a1,a4,a5).
-T(a1,a3,a5) | $ANS(CH1).
```

```
end_of_list.

list(passive).
-T(a2,a3,a6) | $ANS(CH5A).
-T(a2,a6,a3) | $ANS(CH5B).
-T(a4,a3,a6) | $ANS(CH5C).
-T(a4,a6,a3) | $ANS(CH5D).
end_of_list.
```

You well might wish to know what might occur if the clause for the four-point theorem is deleted. (You must wait at this time to see if I later succeed in proving the Overbeek four-point theorem.) To meet your wish, I needed only to comment out the unwanted clause, the following.

```
% -T(x,u,z) | -T(x,v,z) | T(x,u,v) | T(x,v,u).
```

In other words, I used the same input file given earlier but now with both the penultimate and the ultimate clauses in the list_usable commented out.

Again OTTER produced a proof, actually ten proofs, the first one of length 30 rather than 42, and a proof that relies on the same elements of the input file except, of course, the clause for the four-point theorem. Yes, I was surprised, expecting that with one additional hypothesis, the four-point theorem clause, most likely a shorter proof would have been found. That was not the case. The 30-step proof contains twenty steps not in the 42-step proof. Because you might be curious about the nature of this 30-step proof, I now present it.

A 30-Step Bidirectional Proof of the Five-Point Theorem

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Thu Nov 29 15:48:56 2012
The command was "otter". The process ID is 28103.
----> UNIT CONFLICT at 4545.44 sec ----> 318935 [binary,318934.1,278209.1] $ANS(CH1).

Length of proof is 30. Level of proof is 9.

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

1 [] -T(x,y,x)|x=y.
2 [] -T(x,y,u)| -T(y,z,u)|T(x,y,z).
3 [] -T(x,y,z)| -T(x,y,u)|x=y|T(x,z,u)|T(x,u,z).
7 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
8 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
21 [] T(x,y,y).
22 [] T(x,x,y).
23 [] -T(x,y,z)|T(z,y,x).
24 [] T(a2,a3,a4).
25 [] T(a1,a2,a5).
26 [] T(a1,a4,a5).
27 [] -T(a1,a3,a5)|$ANS(CH1).
32 [hyper,24,23] T(a4,a3,a2).
46 [hyper,24,7,22] T(x,f1(x,x,a2,a4,a3),a2).
51 [hyper,25,23] T(a5,a2,a1).
```

70 [hyper,26,23] T(a5,a4,a1).
80 [hyper,26,8,24] T(a5,a3,f1(a3,a2,a1,a5,a4)).
85 [hyper,26,7,24] T(a2,f1(a3,a2,a1,a5,a4),a1).
128 [hyper,51,8,32] T(a1,a3,f1(a3,a4,a5,a1,a2)).
133 [hyper,51,7,32] T(a4,f1(a3,a4,a5,a1,a2),a5).
265 [hyper,46,2,21] T(x,a2,f1(a2,a2,a2,a4,a3)).
673 [hyper,80,23] T(f1(a3,a2,a1,a5,a4),a3,a5).
727 [hyper,85,23] T(a1,f1(a3,a2,a1,a5,a4),a2).
1482 [ur,128,2,27] -T(a3,a5,f1(a3,a4,a5,a1,a2))|$ANS(CH1).
1609 [hyper,133,23] T(a5,f1(a3,a4,a5,a1,a2),a4).
3037 [hyper,265,7,21] T(x,f1(a2,x,y,f1(a2,a2,a2,a4,a3),a2),y).
3047 [hyper,265,1] f1(a2,a2,a2,a4,a3)=a2.
5191 [hyper,727,7,25] T(a1,f1(f1(a3,a2,a1,a5,a4),a1,a1,a5,a2),a1).
9594 [ur,1609,2,1482] -T(a3,a5,a4)|$ANS(CH1).
9635 [ur,9594,2,70] $ANS(CH1)| -T(a3,a5,a1).
9664 [ur,9635,23] $ANS(CH1)| -T(a1,a5,a3).
274135 [para_into,3037.1.2.4,3047.1.1] T(x,f1(a2,x,y,a2,a2),y).
274547 [hyper,274135,2,21] T(x,y,f1(a2,y,y,a2,a2)).
275483 [hyper,274547,2,21] T(x,f1(a2,y,y,a2,a2),y).
276936 [ur,274547,2,27] -T(a1,a3,f1(a2,a5,a5,a2,a2))|$ANS(CH1).
278209 [hyper,275483,2,673] T(f1(a3,a2,a1,a5,a4),a3,f1(a2,a5,a5,a2,a2)).
317503 [hyper,5191,1] a1=f1(f1(a3,a2,a1,a5,a4),a1,a1,a5,a2).
317538 [para_from,317503.1.2,8.3.3,unit_del,727,25] T(a5,f1(a3,a2,a1,a5,a4),a1).
317539 [hyper,317538,23] T(a1,f1(a3,a2,a1,a5,a4),a5).
317723 [hyper,317539,2,673] T(a1,f1(a3,a2,a1,a5,a4),a3).
317771 [hyper,317723,3,317539,unit_del,9664,27] a1=f1(a3,a2,a1,a5,a4)|$ANS(CH1).
318934 [para_from,317771.1.1,276936.1.1] -T(f1(a3,a2,a1,a5,a4),a3,f1(a2,a5,a5,a2,a2))|$ANS(CH1).
318935 [binary,318934.1,278209.1] $ANS(CH1).

You have strong evidence, if you compare the 42-step proof with the 30-step proof, of the value of giving the program the additional clause for the four-point theorem. Indeed, although a shorter proof was found in its absence. far, far less CPU time was required (to complete the 42-step proof) and far, far fewer new conclusions were retained. On the other hand, a possible mystery is offered, for you might guess that access to an additional hypothesis would ordinarily lead to a shorter proof. A possible answer asserts that the 42-step proof resulted because the presence of the additional hypothesis led OTTER down a sharply different path, an answer supported by the difference between the two proofs. When a further modification of the input file that produced the 30-step proof was made, namely, removing access to all lemmas and reasoning from just the Tarski 13-axiom system, no proof was obtained. You thus have more evidence of relying on an approach that uses lemmas. As an intermediate summary, Overbeek's position about not needing binary resolution has been supported by the data presented from the last discussed experiments.

For a more startling example of how the presence of but one additional hypothesis in the input file can produce dramatic results, I first remind you of a 10-step proof of the five-point theorem I presented earlier, one that completed in just over 32,868 CPU-seconds. The input file gave access for OTTER to Lemmas 1, 2, and 3, in addition to the Tarski 13-axiom system, including, therefore, the axioms of both transitivity and connectivity of betweenness. Almost immediately after I studied the bidirectional proofs just discussed, I turned to seeking a forward proof. Specifically, I gave the program an input file that contained the Tarski 13-axiom system and the three lemmas just cited and one additional lemma (or lesser theorem), namely, the four-point theorem of Overbeek. After all, as the data I've supplied shows, its presence can indeed have a sharp effect. Well, it did. The CPU time to complete a proof, in contrast to the approximately 32,868 CPU-seconds, was approximately 1,666 CPU-seconds. As noted, the 10-step proof required the retention of (130709) clauses; the proof under discussion required (48751) clauses.

And now for the drama: rather than completing a proof of roughly length 10, the proof that was returned to me has length 138. Again you see how the presence of one additional hypothesis, rather than leading to a shorter proof, led to a far longer proof; certainly a different path of reasoning was followed. The 10-step proof used but one added lemma, that for symmetry of betweenness; the 138-step used that lemma and also used the clause corresponding to the Overbeek four-point theorem. So, if a proof is finally found for his theorem and I learn of it, or discover it myself—which, actually, I did—the data suggests it will be most useful in the ways discussed here and in preceding paragraphs.

A bit more on the five-point theorem is still in order, especially in view of Overbeek's position about using lemmas and avoiding the use of binary resolution and in view of an important choice that you can make when seeking a proof of an interesting theorem. Specifically, at least with OTTER, you are asked to choose from among the clauses presenting the problem, which are to be placed in list(usable) and which in list(sos). As a reminder, OTTER is not permitted to reason from a set of clauses all of which are in the (initial) list(usable). Such clauses are used to complete the application of an inference rule. Therefore, the placement of an item in the usable list blocks the program from choosing it to begin a path of reasoning. The more items you place in list(usable), as opposed to placing such in list(sos), the more you are restricting the program's attack, an action that in the main adds to efficiency in the study of many areas of mathematics and logic. In general, restriction of reasoning increases the likelihood that unprofitable paths will be ignored. On the other hand, if a so-called key item is placed in list(usable), rather than in list(sos), then a path that might be required for exploration in order to find a proof may be blocked. So you see how complex can be the decisions to make when using an automated reasoning program. As I have continually noted, no algorithm is known for making choices of the types featured here in this notebook.

An experiment that strongly asks to be run has you place all the clauses in list(sos), as opposed to what was done in many of the preceding experiments that placed quite a few clauses in list(usable). That experiment also asks that binary resolution be avoided and that, perhaps in part as a consequence, that Lemmas 1, 2, and 3 be used. Also, in contrast to an experiment featured rather early in this section that yielded a 15-step proof, all thirteen of Tarski's axioms are to be used, including the two for, respectively, transitivity and connectivity of betweenness. For clarity, the clause corresponding to the Overbeek four-point theorem is not to be included. With an assignment of the value 15 to max_weight, the experiment was conducted. An 11-step forward proof was completed in approximately 16,096 CPU-seconds, with the retention of clause (164950). Present in the 11-step proof are the use of both Lemmas 1 and 3, T(x,y,y) and symmetry of betweenness.

But, are the three lemmas needed? Indeed, would the Tarski 13-axiom system suffice? The answer is that his axiom system does suffice, with no added lemmas—not even the four-point theorem was needed—as established by simply conducting another experiment, one in which the three lemmas are commented out. That experiment yielded a 15-step proof in approximately 2,003 CPU-seconds with retention of clause (107217). This 15-step proof is sharply different from the 15-step proof offered much earlier in this section. Oh, the wonders of mathematics and of automated reasoning! Indeed, rather than aiding the search for a proof by including three lemmas, their inclusion, although leading to an 11-step proof rather than a 15-step proof, resulted in far more computer time before the goal was reached. An examination of the just-cited proof shows that transitivity of betweenness was used, but not connectivity. So, yes, one more experiment is merited; indeed, some might naturally wonder how all would go if those two axioms, of the Tarski thirteen, were not available.

OTTER produced a 14-step forward proof in approximately 236 CPU-seconds with retention of clause (19465). Binary resolution was not used; indeed, ten of the fourteen steps were obtained with hyperresolution and four with paramodulation. And again charm is present: only six of Tarski's axioms are used. When and if the use of binary resolution will be necessary, as it appears to be required in a proof found in Section 8, awaits my study of deeper theorems in Tarskian geometry.

5. Emulation of Earlier Successes and More Proofs

OTTER's allowing one to prove many theorems in a single run is precisely what I almost demand, since I am an impatient researcher. Of course, the set of theorems to be considered in an experiment must share the axiom set being used—not necessarily using all of them, clearly. I now present an input file that proved almost all the theorems in Chapter 2 of the Szmielew book and some of the theorems in Chapter 3.

I also list a superset of those theorems, shortly after I offer you an input file.

A Most Useful Input File

```
assign(max_weight,22).
assign(max_proofs,20).
set(hyper_res).
set(input_sos_first).
clear(order_hyper).
clear(print_kept).
set(para_into).
set(para_from).
set(ur_res).
% set(binary_res).
set(unit_deletion).

weight_list(pick_and_purge).
weight(f4(*(0),(0),*(0),*(0)),1).
end_of_list.

list(sos).
x = x.
ac1 != ac2.
T(ac1,ac2,ac3).
E(ac2,ac3,a4,a5).
E(b1,b2,b3,b4).
T(a7,a8,a9).
T(aa,ab,ac).
T(aa1,ab1,ac1).
T(da,db,dd).
T(db,dc,dd).
T(ba,bb,bc).
T(ba1,bb1,bc1).
T(a7,a8,a9).
E(aa,ab,aa1,ab1).
E(ab,ac,ab1,ac1).
E(eb,ec,eb1,ec1).
E(ba,bb,ba1,bb1).
E(bb,bc,bb1,bc1).
E(d,e,d3,d3).
E(b3,b4,b5,b6).
E(b1,b2,b3,b4).
T(da,db,dc).
T(db,da,dc).
T(a1,a2,a5).
T(a1,a4,a5).
T(a2,a3,a4).
T(A2,a6,a4).
T(a,b,c).
T(cba,cbb,cbc).
T(cbb,cbc,cbd).
T(fa,fb,fd).
T(fb,fc,fd).
```

```
T(ga,gb,gc).
T(ga,gc,gd).
T(gga,ggb,ggc).
T(gga,ggc,ggd).
T(ggga,gggb,gggc).
T(gggb,gggc,gggd).
T(cba,cbb,cbc).
T(cbb,cbc,cbd).
cbb != cbc.
T(ggga,gggb,gggc).
T(gggb,gggc,gggd).
gggb != gggc.
T(gga,ggb,ggc).
T(gga,ggc,ggd).
T(ga,gb,gc).
T(ga,gc,gd).
T(fa,fb,fd).
T(fb,fc,fd).
T(a,b,c).
% following 20 clauses are translations of first 20 clauses from ch6 of my 1987 book
% where I focused on Tarski's 13-axiom system

-T(x,y,x) | (x = y).
-T(x,y,u) | -T(y,z,u) | T(x,y,z).
-T(x,y,z) | -T(x,y,u) | (x = y) | T(x,z,u) | T(x,u,z).
E(x,y,y,x).
-E(x,y,z,z) | (x = y).
-E(x,y,z,u) | -E(x,y,v,w) | E(z,u,v,w).
-T(x,v,u) | -T(y,u,z) | T(x,f1(v,x,y,z,u),y).
-T(x,v,u) | -T(y,u,z) | T(z,v,f1(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,z,f2(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,y,f3(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(f2(v6,x,y,z,u),v6,f3(v6,x,y,z,u)).
-E(x1,y1,x2,y2) | -E(y1,z1,y2,z2) | -E(x1,u1,x2,u2) | -E(y1,u1,y2,u2) | -T(x1,y1,z1) | -T(x2,y2,z2) |
 (x1 = y1) | E(z1,u1,z2,u2).
T(x,y,f4(x,y,u,v)).
E(y,f4(x,y,u,v),u,v).
-T(c1,c2,c3).
-T(c2,c3,c1).
-T(c3,c1,c2).
-E(x,u,x,v) | -E(y,u,y,v) | -E(z,u,z,v) | (u = v) | T(x,y,z) | T(y,z,x) | T(z,x,y).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | E(u1,y1,u1,f5(x1,y1,z1,x2,z2,u1)).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | T(x2,f5(x1,y1,z1,x2,z2,u1),z2).
% T(x,y,y).
% T(x,x,y).
% -T(x,y,z) | T(z,y,x).
% -T(x,y1,z) | -T(x,y2,z) | T(x,y1,y2) | T(x,y2,y1).
end_of_list.
list(sos).
T(a2,a3,a4).
T(a1,a2,a5).
T(a1,a4,a5).
T(A2,a6,a4).
```

```
end_of_list.

list(passive).
f4(ac1,ac2,a4,a5) != ac3 | $ANS(CHZ12).
-T(cba,cbb,cbd) | $ANS(CH372).
-T(ggga,gggc,gggd) | $ANS(CHZ371).
-T(gga,ggb,ggd) | $ANS(CHZ362).
-T(gb,gc,gd) | $ANS(CHZ361).
-T(ga,gb,gd) | $ANS(CHZ361a).
-T(fa,fc,fd) | $ANS(CHZ352).
-T(c,b,a) | $ANS(CHZ32).
can != cb | $ANS(CHZ34).
d != e | $ANS(CHZZZ).
-E(a7,a8,a7,a8) | $ANS(CHZ1).
-E(b1,b1,b2,b2) | $ANS(CHZ8).
-E(b1,b2,b4,b3) | $ANS(CHZ5).
-E(b1,b2,b5,b6) | $ANS(CHZ3).
-E(b2,b1,b3,b4) | $ANS(CHZ4).
-E(b3,b4,b1,b2) | $ANS(CHZ2).
-E(ba,bc,ba1,bc1) | $ANS(CHZ11).
-T(a1,a1,a2) | $ANS(CH2).
-T(a1,a2,a2) | $ANS(CH3).
-T(da,db,dc) | $ANS(CHZ351).
% -T(a1,a3,a5) | $ANS(CH1).
% -T(a2,a3,a6) | $ANS(CH5A).
% -T(a2,a6,a3) | $ANS(CH5B).
% -T(a4,a3,a6) | $ANS(CH5C).
% -T(a4,a6,a3) | $ANS(CH5D).
end_of_list.
```

Note that both transitivity and connectivity of betweenness are included in the preceding file.

Especially in view of the title of this section, you might naturally ask about the birth of the given input file. I borrowed (emulated) that which worked and which I found pleasing from the files relied upon in the preceding section. Specifically, with a few modifications to parameter values of the input file, cited near the end of the preceding section that reached its goal with retention of clause (107217), I produced an appropriate input file. An inspection of the just-given input file reveals the presence of quite a number of ground clauses, clauses without reliance on variables. Their presence is explained by the items found in list(passive), the theorems to prove. Those uncommented in the passive list that have, for example, CH2 in the ANS literal correspond to theorems from Szmielew's Chapter 3. In particular, the denial of a nonunit clause (corresponding to a target theorem) produces a set of unit clauses. I now offer, for convenience, a superset of theorems that were the targets, found in list(passive), taken from the Szmielew book. To aid you in connecting these theorems to the items found in the passive list of the just-given input file, I will add to each the corresponding notation that you will find in the input file.

```
E(x,y,x,y). % Satz 2.1, CHZ1
-E(xa,xb,xc,xd) | E(xc,xd,xa,xb). % Satz 2.2, CHZ2
-E(xa,xb,xc,xd) | -E(xc,xd,xe,xf) | E(xa,xb,xe,xf). % Satz 2.3, CHZ3
-E(xa,xb,xc,xd) | E(xb,xa,xc,xd). % Satz 2.4, CHZ4
-E(xa,xb,xc,xd) | E(xa,xb,xd,xc). % Satz 2.5, CHZ5
E(x,x,y,y). % Satz 2.8, CHZ8
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xb,xa1,xb1) | -E(xb,xc,xb1,xc1) | E(xa,xc,xa1,xc1).
% Satz 2.11, CHZ11
xq = xa | -T(xq,xa,u) | -E(xa,u,xc,xd) | ext(xq,xa,xc,xd) = u. % Satz 2.12, CHZ12
T(x,y,y). % Satz 3.1, CH3
```

-E(u,v,x,x) | u=v. % Not one of Szmielew's theorems but we proved it, CHZZ
-T(xa,xb,xc) | T(xc,xb,xa). % Satz 3.2., CH4
T(xa,xa,xb). % Satz 3.3, CH2
-T(xa,xb,xc) | -T(xb,xa,xc) | xa = xb. % Satz 3.4, CHZ34
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xb,xc). % Satz 3.51, CHZ351
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xc,xd). % Satz 3.52, CHZ352
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd). % Satz 3.61, CHZ361
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd). % Satz 3.61a, CHZ361a
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd). % Satz 3.71, CHZ71
alpha != beta. % related to Satz 3.14; easily provable if added to sst 3h.in.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd). % Satz 3.62, CHZ362
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd). % Satz 3.71
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd). % Satz 3.72, CH72

First, you quickly observe that 3.51 is an odd inclusion, for it is the clause, among Tarski's thirteen axioms, for transitivity of betweenness and, therefore, is not actually a target. Second, you note that, in list(passive), where a nonunit theorem is concerned, you just find the denial of the positive literal(s), with the remaining literals, which arise from the denial of the theorem, found in other lists. The use of this input file yielded many proofs (to be discussed shortly). When a nonunit theorem was proved, the last line of that proof (before the item that signaled completion with binary resolution) is often a ground unit clause. As noted, such last lines must not be adjoined in future runs as so-called lemmas, but they could clearly be used as resonators.

The use of the sought-after single input file, designed to find many proofs, led to the following proofs.

----> UNIT CONFLICT at 0.00 sec ----> 130 [binary,129.1,86.1] $ANS(CHZZZ).
Length of proof is 1. Level of proof is 1.
----> UNIT CONFLICT at 0.00 sec ----> 135 [binary,134.1,87.1] $ANS(CHZ1).
Length of proof is 1. Level of proof is 1.
----> UNIT CONFLICT at 0.00 sec ----> 137 [binary,136.1,91.1] $ANS(CHZ4).
Length of proof is 1. Level of proof is 1.
----> UNIT CONFLICT at 0.03 sec ----> 563 [binary,562.1,85.1] $ANS(CHZ34).
Length of proof is 2. Level of proof is 2.
----> UNIT CONFLICT at 0.07 sec ----> 1129 [binary,1128.1,92.1] $ANS(CHZ2).
Length of proof is 2. Level of proof is 2.
----> UNIT CONFLICT at 0.12 sec ----> 1205 [binary,1204.1,89.1] $ANS(CHZ5).
Length of proof is 3. Level of proof is 3.
----> UNIT CONFLICT at 0.12 sec ----> 1207 [binary,1206.1,90.1] $ANS(CHZ3).
Length of proof is 3. Level of proof is 3.
----> UNIT CONFLICT at 0.28 sec ----> 1984 [binary,1983.1,88.1] $ANS(CHZ8).
Length of proof is 2. Level of proof is 2.
----> UNIT CONFLICT at 0.28 sec ----> 1986 [binary,1985.1,95.1] $ANS(CH3).
Length of proof is 2. Level of proof is 2.
----> UNIT CONFLICT at 2.59 sec ----> 11009 [binary,11008.1,94.1] $ANS(CH2).
Length of proof is 6. Level of proof is 5.
----> UNIT CONFLICT at 3.60 sec ----> 17481 [binary,17480.1,83.1] $ANS(CHZ32).
Length of proof is 6. Level of proof is 5.
----> UNIT CONFLICT at 23.29 sec ----> 80083 [binary,80082.1,81.1] $ANS(CHZ361a).
Length of proof is 9. Level of proof is 6.
----> UNIT CONFLICT at 23.71 sec ----> 81223 [binary,81222.1,79.1] $ANS(CHZ362).
Length of proof is 9. Level of proof is 6.
----> UNIT CONFLICT at 951.40 sec ----> 281761 [binary,281760.1,82.1] $ANS(CHZ352).
Length of proof is 14. Level of proof is 8.
----> UNIT CONFLICT at 1006.63 sec ----> 299952 [binary,299951.1,80.1] $ANS(CHZ361).

Length of proof is 15. Level of proof is 9.
----> UNIT CONFLICT at 5704.82 sec ----> 904661 [binary,904660.1,76.1] $ANS(CHZ12).
Length of proof is 6. Level of proof is 5.

Of the targets found in list(passive) of this input file, three were not reached: CHZ11, CHZ71, and CH72. In other words, with a single run, sixteen theorems of interest were proved. Further and most pleasing, all the sixteen proofs are forward proofs. Binary resolution was not used, nor were any lemmas; indeed, just the thirteen Tarski axioms sufficed.

As the following experiment with its results illustrates, automated reasoning and, it seems, mathematics continue to offer surprises. In particular, among my experiments, I modified the Most Useful Input File by commenting out both transitivity and connectivity of betweenness with the goal of—odd though it might be—proving one or more of the three targets not yet reached. The vague notion behind this move was that those two axioms might be so fruitful that OTTER would never reach any of the three sought-after proofs. I remind you that McCune frequently omitted axioms that were dependent when he studied various areas of abstract algebra. And, as you more than suspect because of my inclusion of this bit of discussion, OTTER did eventually find a gem, the following.

A Sought-After Proof of CHZ11

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Sat Dec 15 07:56:48 2012
The command was "otter". The process ID is 2021.
----> UNIT CONFLICT at 141384.79 sec ----> 2334603 [binary,2334602.1,91.1] $ANS(CHZ11).

Length of proof is 32. Level of proof is 13.

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

11 [] T(ba,bb,bc).
12 [] T(ba1,bb1,bc1).
17 [] E(ba,bb,ba1,bb1).
18 [] E(bb,bc,bb1,bc1).
52 [] -T(x,y,x)|x=y.
53 [] E(x,y,y,x).
54 [] -E(x,y,z,z)|x=y.
55 [] -E(x,y,z,u)| -E(x,y,v,w)|E(z,u,v,w).
56 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
57 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
61 [] -E(x1,y1,x2,y2)| -E(y1,z1,y2,z2)| -E(x1,u1,x2,u2)| -E(y1,u1,y2,u2)| -T(x1,y1,z1)|
 -T(x2,y2,z2)|x1=y1|E(z1,u1,z2,u2).
62 [] T(x,y,f4(x,y,u,v)).
63 [] E(y,f4(x,y,u,v),u,v).
91 [] -E(ba,bc,ba1,bc1)|$ANS(CHZ11).
103 [hyper,55,53,53] E(x,y,x,y).
118 [hyper,55,18,53] E(bb1,bc1,bc,bb).
119 [hyper,55,17,53] E(ba1,bb1,bb,ba).
413 [hyper,63,54] x=f4(y,x,z,z).
475 [hyper,103,55,18] E(bb1,bc1,bb,bc).
476 [hyper,103,55,17] E(ba1,bb1,ba,bb).
496 [hyper,118,55,53] E(bc1,bb1,bc,bb).
497 [hyper,118,55,53] E(bc,bb,bc1,bb1).
498 [hyper,119,55,53] E(bb1,ba1,bb,ba).
```

499 [hyper,119,55,53] E(bb,ba,bb1,ba1).
615 [para_from,413.1.2,63.1.2] E(x,x,y,y).
617 [para_from,413.1.2,62.1.3] T(x,y,y).
830 [hyper,617,57,12] T(bc1,bb1,f1(bb1,ba1,x,bc1,bc1)).
831 [hyper,617,57,11] T(bc,bb,f1(bb,ba,x,bc,bc)).
883 [hyper,617,56,12] T(ba1,f1(bb1,ba1,x,bc1,bc1),x).
884 [hyper,617,56,11] T(ba,f1(bb,ba,x,bc,bc),x).
1141 [hyper,615,61,476,475,498,12,11] ba1=bb1|E(bc1,ba1,bc,ba).
1221 [hyper,615,61,17,18,499,11,12] ba=bb|E(bc,ba,bc1,ba1).
1287 [para_from,1141.1.2,496.1.2] E(bc1,ba1,bc,bb)|E(bc1,ba1,bc,ba).
1297 [hyper,1221,55,103] ba=bb|E(bc1,ba1,bc,ba).
14003 [hyper,883,52] ba1=f1(bb1,ba1,ba1,bc1,bc1).
14091 [para_from,14003.1.2,830.1.3] T(bc1,bb1,ba1).
14402 [hyper,884,52] ba=f1(bb,ba,ba,bc,bc).
14490 [para_from,14402.1.2,831.1.3] T(bc,bb,ba).
14522 [hyper,14490,61,497,499,615,18,14091] bc=bb|E(ba,bc,ba1,bc1).
14674 [hyper,14522,55,53] bc=bb|E(ba1,bc1,bc,ba).
14969 [hyper,14674,55,53] bc=bb|E(bc1,ba1,bc,ba).
15694 [para_into,14969.1.2,1297.1.2] bc=ba|E(bc1,ba1,bc,ba).
16401 [para_into,15694.1.1,14969.1.1] bb=ba|E(bc1,ba1,bc,ba).
2334496 [para_into,1287.1.4,16401.1.1] E(bc1,ba1,bc,ba).
2334519 [hyper,2334496,55,53] E(bc,ba,ba1,bc1).
2334602 [hyper,2334519,55,53] E(ba,bc,ba1,bc1).
2334603 [binary,2334602.1,91.1] $ANS(CHZ11).

The retention of more than 2 million clauses and the use of approximately 141,300 CPU-seconds to complete this 32-step proof provide a fine example of the value of waiting for a long, long time as the program tirelessly seeks to complete an assignment. Again you encounter that mystery with no answer at this time: When is it more effective to exclude axioms? Yes, intuition suggests that more axioms and, perhaps, additional lemmas, should in general lead to proofs in shorter time, the retention of fewer clauses, and the finding of proofs net yet found. By the way, that experiment did not prove either Z34 or CHZ361. (As noted much earlier, CH4, the symmetry of betweenness, was proved with Input File 1.)

CHZ371 and CH372 still remain unproven in this discussion. Of course, perhaps some modifications of the parameter values and options in the Most Useful Input File would lead to the two sought-after proofs. As an aside, I have heard disappointment expressed when OTTER does not find a desired proof. My response has typically been that, if you consult a fellow researcher without gaining what is needed, you do not from thence forward never consult that person again. Usually what I do with OTTER is try many approaches, some of which I have detailed earlier in this notebook. Rather than listing various experiments, I have selected one that did in fact produce a proof of CHZ371. The corresponding input file—which I discuss in the continued goal of providing you more and more knowledge about research—included the following five lemmas, placed in list(sos).

T(x,y,y).
T(x,x,y).
-T(x,y,z) | T(z,y,x).
E(x,y,x,y). % Satz 2.1
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd). % Satz 3.61.

These five lemmas, as you recognize, are among the theorems proved with the Most Useful Input File. The first three are, of course, the three lemmas in focus in Section 1. But, instead of presenting the input file that was used to obtain a proof of CHZ71, I think it more instructive to discuss an extension of the Most Useful Input File.

In particular, I took that input file and adjoined in list(sos) the five lemmas (lesser theorems), expecting, because of my knowledge of a proof of CHZ371, that all five would most likely be used. Well, such is not the case, as you see from the following proof.

A 4-Step Proof of CHZ371

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Sat Dec 22 09:24:59 2012
The command was "otter". The process ID is 17316.
----> UNIT CONFLICT at 250.51 sec ----> 374532 [binary,374531.1,83.1] $ANS(CHZ371).

Length of proof is 4. Level of proof is 4.

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

4 [] E(x,y,x,y).
5 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xb,xc,xd).
47 [] T(ggga,gggb,gggc).
48 [] T(gggb,gggc,gggd).
49 [] gggb!=gggc.
61 [] -E(x,y,z,z)|x=y.
68 [] -E(x1,y1,x2,y2)| -E(y1,z1,y2,z2)| -E(x1,u1,x2,u2)| -E(y1,u1,y2,u2)| -T(x1,y1,z1)| -T(x2,y2,z2)|x1=y1|E(z1,u1,z2,u2).
69 [] T(x,y,f4(x,y,u,v)).
70 [] E(y,f4(x,y,u,v),u,v).
83 [] -T(ggga,gggc,gggd)|$ANS(CHZ371).
871 [hyper,69,5,47] T(gggb,gggc,f4(ggga,gggc,x,y)).
4152 [hyper,871,68,4,70,4,4,48,unit_del,49] E(f4(ggga,gggc,gggc,gggd),x,gggd,x).
374446 [hyper,4152,61] f4(ggga,gggc,gggc,gggd)=gggd.
374531 [para_from,374446.1.1,69.1.3] T(ggga,gggc,gggd).
374532 [binary,374531.1,83.1] $ANS(CHZ371).

In contrast to the earlier proof of CHZ371 (which gave me the idea for the extension of the Most Useful Input File), whose proof relied on so many of the five lemmas, as you see, this proof relies on just the last two lemmas now cited explicitly.

E(x,y,x,y).
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd).

I had OTTER continue to use the input file that yielded the cited proof of CHZ371, expecting that nothing would occur in the context of a proof of CH372, but knowing that the cost of doing so was zero. I find that my impatience is well served by running many experiments at the same time. Therefore, so-to-speak in parallel, I turned to the Beeson approach, that of lemma adjunction. Specifically, I adjoined to list(sos) Satz 3.71, equivalently, CHZ371. The effectiveness of this move, of this incarnation of lemma adjunction, is nicely demonstrated with the following proof of CH372, partially annotated to highlight the use of lemmas

A 9-Step Proof of CH372, Relying on CHZ371

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Fri Dec 28 08:21:06 2012
The command was "otter". The process ID is 18187.
----> UNIT CONFLICT at 0.23 sec ----> 3362 [binary,3361.1,81.1] $ANS(CH372).

Length of proof is 9. Level of proof is 4.

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

2 [] -T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xc,xd). % CHZ371
4 [] -T(x,y,z) | T(z,y,x). % CH4, sym of between
5 [] E(x,y,x,y). % CHZ1
34 [] T(cba,cbb,cbc). % from denial of CH372
35 [] T(cbb,cbc,cbd). % from denial of CH372
46 [] cbb!=cbc. % from denial of CH372
60 [] -E(x,y,z,z)|x=y.
61 [] -E(x,y,z,u)| -E(x,y,v,w)|E(z,u,v,w).
69 [] E(y,f4(x,y,u,v),u,v).
81 [] -T(cba,cbb,cbd)|$ANS(CH372).
100 [hyper,34,4] T(cbc,cbb,cba).
101 [hyper,35,4] T(cbd,cbc,cbb).
122 [ur,60,46] -E(cbb,cbc,x,x).
950 [hyper,69,60] x=f4(y,x,z,z).
1848 [hyper,101,2,100] cbc=cbb|T(cbd,cbb,cba).
3214 [ur,122,61,5] -E(x,x,cbb,cbc).
3221 [para_from,950.1.2,69.1.2] E(x,x,y,y).
3315 [para_from,1848.1.1,3214.1.4,unit_del,3221] T(cbd,cbb,cba).
3361 [hyper,3315,4] T(cba,cbb,cbd).
3362 [binary,3361.1,81.1] $ANS(CH372).

Precisely as I was writing this material, I discovered a lemma that Beeson had proved, taken from Szmielew, that, apparently, I never proved. I offer it to you as a challenge in case you wish to pause and play the game. You are asked to prove that E(x,y,y,x) is a theorem in Tarskian geometry, an intuitively obvious fact about equidistance. Since, from a cursory examination of my files, I cannot find a proof, I have no strong opinion about which, if any, lemmas to use, lemmas that you may have already proved, for example. It would be nice to see a proof of this symmetry of equidistance based solely on the thirteen Tarski axioms (given in his 1959 system), without including any other results in Tarskian geometry.

Now I return to the discussion of proving CH372, with a focus on a result that is, for those who enjoy huge numbers, most entertaining. As I noted a bit earlier, the input file that produced the cited proof of CHZ371 was allowed to continue to be in use, running all the time I have been writing the preceding. From basketball: No harm, no foul. I expected nothing but focused on other experiments and on the writing of this notebook. Well, rather than nothing, you are now presented with the following proof.

A Startling Proof of CH372

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Sat Dec 22 09:24:59 2012
The command was "otter". The process ID is 17316.
----> UNIT CONFLICT at 111285.77 sec ----> 5546609 [binary,5546608.1,82.1] $ANS(CH372).

Length of proof is 9. Level of proof is 8.

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

3 [] -T(x,y,z)|T(z,y,x). % CH4, sym of between
4 [] E(x,y,x,y). % CHZ1
5 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xb,xc,xd). % CHZ361
44 [] T(cba,cbb,cbc). % from denial of CH372
45 [] T(cbb,cbc,cbd). % from denial of CH372

```
46 [] cbb!=cbc. % from denial of CH372
57 [] -T(x,y,x)|x=y.
61 [] -E(x,y,z,z)|x=y.
63 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
64 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
68 [] -E(x1,y1,x2,y2)| -E(y1,z1,y2,z2)| -E(x1,u1,x2,u2)| -E(y1,u1,y2,u2)| -T(x1,y1,z1)|
 -T(x2,y2,z2)|x1=y1|E(z1,u1,z2,u2).
69 [] T(x,y,f4(x,y,u,v)).
70 [] E(y,f4(x,y,u,v),u,v).
82 [] -T(cba,cbb,cbd)|$ANS(CH372).
873 [hyper,69,5,44] T(cbb,cbc,f4(cba,cbc,x,y)).
4261 [hyper,873,68,4,70,4,4,45,unit_del,46] E(f4(cba,cbc,cbc,cbd),x,cbd,x).
379633 [hyper,4261,61] f4(cba,cbc,cbc,cbd)=cbd.
379718 [para_from,379633.1.1,69.1.3] T(cba,cbc,cbd).
381141 [hyper,379718,64,44] T(cbd,cbb,f1(cbb,cba,cba,cbd,cbc)).
381309 [hyper,379718,63,44] T(cba,f1(cbb,cba,cba,cbd,cbc),cba).
5542561 [hyper,381309,57] cba=f1(cbb,cba,cba,cbd,cbc).
5543626 [para_from,5542561.1.2,381141.1.3] T(cbd,cbb,cba).
5546608 [hyper,5543626,3] T(cba,cbb,cbd).
5546609 [binary,5546608.1,82.1] $ANS(CH372).
```

To quote J Moore of the Boyer-Moore brilliant work in program verification, how robust OTTER is, retaining more than 5.5 million new conclusions before the proof was completed. And, how marvelous, you have two 9-step proofs, but how they do differ! You might note that, in lieu of access to CHZ371, OTTER turned to CHZ361 and reliance on Tarski axioms not used when CHZ371 was in play.

Now to return to a theorem in focus early in this notebook. Have you, on your own, found a proof of the four-point theorem, suggested by Overbeek? Have you tried and tried, as I have throughout this writing, sought a proof, and with no success? As you recall, the theorem, though unproved at the time, was used at some points to prove other theorems, specifically, the five-point theorem. After all, intuitively at least, it was clearly true. The question that was present implicitly asked how to prove the four-point theorem, especially from the thirteen Tarski axioms with no assistance from lemmas. Well, I now officially thank Beeson, and his indispensable aid in the study of the four-point theorem, for providing me with what was sufficient to obtain a proof. He has an advantage over various researchers, including me; indeed, I am almost certain he is comfortable thinking in terms of Tarskian geometry. He suggested that I define a point, q, that extends the line involving a2 and a4, in the following manner.

```
q = f4(a4,a2,a4,a2).
```

The axiom that is pertinent, I believe, is the following.

```
axiom of segment construction (two clauses):
T(x,y,f4(x,y,u,v)).
E(y,f4(x,y,u,v),u,v).
```

You might well wish to see the set of clauses that I then used to rely on Beeson's suggestion, the following.

```
% following is for the four-point theorem.
q = f4(a4,a2,a4,a2).
T(a2,a3,a4).
T(a2,a6,a4).
-T(a2,a3,a6) | $ANS(CH5A).
-T(a2,a6,a3) | $ANS(CH5B).
-T(a3,a6,a4) | $ANS(CH5c).
-T(a6,a3,a4) | $ANS(CH5d).
```

The reason for including the variations that focus on different combinations is that I could not predict the direction OTTER would take. For example, would a2 play the key role, or would a4? I included those famous three lemmas you visited in the beginning, the following.

```
-T(x,y,z) | T(z,y,x).
T(x,y,y).
T(x,x,y).
```

I have not succeeded in producing a proof of the four-point theorem without these three lemmas. I present that task to you as a challenge. As you will see in the following proof OTTER found, the third of the three lemmas was in fact not used. You will also see how vital q is to the proof, appearing in nineteen lines. And you will see, as indicated with the last line, that the proof is bidirectional.

A Bidirectional Proof of the Four-Point Theorem

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Thu Dec 13 11:23:50 2012
The command was "otter". The process ID is 17686.
----> UNIT CONFLICT at 0.18 sec ----> 1495 [binary,1494.1,955.1] $ANS(CH5B)|$ANS(CH5A).

Length of proof is 26. Level of proof is 10.

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

1 [] q=f4(a4,a2,a4,a2).
2 [] T(a2,a3,a4).
3 [] T(a2,a6,a4).
4 [] -T(a2,a3,a6)|$ANS(CH5A).
5 [] -T(a2,a6,a3)|$ANS(CH5B).
10 [] -T(x,y,u)| -T(y,z,u)|T(x,y,z).
11 [] -T(x,y,z)| -T(x,y,u)|x=y|T(x,z,u)|T(x,u,z).
12 [] E(x,y,y,x).
13 [] -E(x,y,z,z)|x=y.
14 [] -E(x,y,z,u)| -E(x,y,v,w)|E(z,u,v,w).
21 [] T(x,y,f4(x,y,u,v)).
22 [] E(y,f4(x,y,u,v),u,v).
26 [] -T(x,y,z)|T(z,y,x).
27 [] T(x,y,y).
45 [binary,10.2,3.1] -T(x,a2,a4)|T(x,a2,a6).
46 [binary,10.2,2.1] -T(x,a2,a4)|T(x,a2,a3).
59 [binary,14.3,13.1] -E(x,y,z,u)| -E(x,y,v,v)|z=u.
60 [hyper,14,12,12] E(x,y,x,y).
83 [para_into,21.1.3,1.1.2] T(a4,a2,q).
87 [para_into,22.1.2,1.1.2] E(a2,q,a4,a2).
98 [binary,26.2,5.1] -T(a3,a6,a2)|$ANS(CH5B).
99 [binary,26.2,4.1] -T(a6,a3,a2)|$ANS(CH5A).
295 [binary,83.1,26.1] T(q,a2,a4).
395 [ur,10,27,99] $ANS(CH5A)| -T(a3,a2,a3).
520 [binary,295.1,46.1] T(q,a2,a3).
521 [binary,295.1,45.1] T(q,a2,a6).
581 [ur,10,27,395] $ANS(CH5A)| -T(a2,a3,a2).
625 [binary,520.1,26.1] T(a3,a2,q).
665 [binary,521.1,26.1] T(a6,a2,q).
817 [ur,10,625,581] -T(a2,a3,q)|$ANS(CH5A).
```

818 [ur,10,625,99] -T(a6,a3,q)|$ANS(CH5A).
870 [ur,10,665,98] -T(a3,a6,q)|$ANS(CH5B).
955 [binary,817.1,26.2] $ANS(CH5A)| -T(q,a3,a2).
964 [binary,818.1,26.2] $ANS(CH5A)| -T(q,a3,a6).
980 [binary,870.1,26.2] $ANS(CH5B)| -T(q,a6,a3).
1183 [ur,11,520,521,964,980] $ANS(CH5B)|q=a2|$ANS(CH5A).
1204 [para_from,1183.1.2,2.1.1] T(q,a3,a4)|$ANS(CH5B)|$ANS(CH5A).
1407 [para_into,87.1.1,1183.1.2] E(q,q,a4,a2)|$ANS(CH5B)|$ANS(CH5A).
1481 [hyper,59,1407,60] $ANS(CH5B)|$ANS(CH5A)|a4=a2.
1494 [para_from,1481.1.1,1204.1.3,factor_simp,factor_simp] T(q,a3,a2)
$ANS(CH5B)|$ANS(CH5A).
1495 [binary,1494.1,955.1] $ANS(CH5B)|$ANS(CH5A).

As for two of the questions that naturally arise, the answer to the first is no, I have not been able to find a forward proof yet; and another challenge is issued. The answer to the second is yes, I have a proof that avoids the use of binary resolution, the following.

A Proof of the Four-Point Theorem Avoiding Binary Resolution

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Sat Dec 29 09:52:40 2012
The command was "otter". The process ID is 23428.
----> UNIT CONFLICT at 0.38 sec ----> 1971 [binary,1970.1,922.1] $ANS(CH5A)|$ANS(CH5B).

Length of proof is 26. Level of proof is 11.

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

1 [] q=f4(a4,a2,a4,a2).
2 [] T(a2,a3,a4).
3 [] T(a2,a6,a4).
4 [] -T(a2,a3,a6)|$ANS(CH5A).
5 [] -T(a2,a6,a3)|$ANS(CH5B).
10 [] -T(x,y,u)| -T(y,z,u)|T(x,y,z).
11 [] -T(x,y,z)| -T(x,y,u)|x=y|T(x,z,u)|T(x,u,z).
12 [] E(x,y,y,x).
13 [] -E(x,y,z,z)|x=y.
14 [] -E(x,y,z,u)| -E(x,y,v,w)|E(z,u,v,w).
21 [] T(x,y,f4(x,y,u,v)).
22 [] E(y,f4(x,y,u,v),u,v).
26 [] -T(x,y,z)|T(z,y,x).
27 [] T(x,y,y).
56 [para_into,21.1.3,1.1.2] T(a4,a2,q).
58 [hyper,14,22,12] E(x,y,f4(z,u,x,y),u).
59 [hyper,13,22] x=f4(y,x,z,z).
60 [para_into,22.1.2,1.1.2] E(a2,q,a4,a2).
68 [ur,26,5] -T(a3,a6,a2)|$ANS(CH5B).
69 [ur,26,4] -T(a6,a3,a2)|$ANS(CH5A).
250 [hyper,26,56] T(q,a2,a4).
379 [ur,10,27,69] $ANS(CH5A)| -T(a3,a2,a3).
559 [hyper,10,250,3] T(q,a2,a6).
560 [hyper,10,250,2] T(q,a2,a3).

578 [ur,10,27,379] $ANS(CH5A)| -T(a2,a3,a2).
618 [hyper,26,559] T(a6,a2,q).
659 [hyper,26,560] T(a3,a2,q).
812 [ur,10,618,68] -T(a3,a6,q)|$ANS(CH5B).
865 [ur,10,659,578] -T(a2,a3,q)|$ANS(CH5A).
866 [ur,10,659,69] -T(a6,a3,q)|$ANS(CH5A).
916 [ur,26,812] $ANS(CH5B)| -T(q,a6,a3).
922 [ur,26,865] $ANS(CH5A)| -T(q,a3,a2).
930 [ur,26,866] $ANS(CH5A)| -T(q,a3,a6).
940 [para_from,59.1.2,58.1.3] E(x,x,y,y).
1064 [ur,11,559,560,916,930] $ANS(CH5A)|q=a2|$ANS(CH5B).
1074 [para_from,1064.1.1,60.1.2] E(a2,a2,a4,a2)|$ANS(CH5A)|$ANS(CH5B).
1092 [para_from,1064.1.2,2.1.1] T(q,a3,a4)|$ANS(CH5A)|$ANS(CH5B).
1510 [hyper,14,1074,940] $ANS(CH5A)|$ANS(CH5B)|E(a4,a2,x,x).
1949 [hyper,13,1510] $ANS(CH5A)|$ANS(CH5B)|a4=a2.
1970 [para_from,1949.1.1,1092.1.3] T(q,a3,a2)|$ANS(CH5A)|$ANS(CH5B).
1971 [binary,1970.1,922.1] $ANS(CH5A)|$ANS(CH5B).

As for a comparison of the two 26-step proofs, nine of the deduced steps are not shared. In the second proof, UR-resolution is applied far more. However, a careful consideration of the axioms, of the inclusion of the definition of the point q, and of the theorem to be proved perhaps produces little or no surprise that both proofs share, for example, the following three deductions.

T(a4,a2,q).
T(a3,a2,q).
T(a6,a2,q).

In the 26-step proof of the four-point theorem, you find interesting deductions such as clause (56), which says that a2 is between a4 and q. Clause (60) asserts that the distance from a2 to q equals the distance from a4 to a2. And clause (560) says that a2 is between q and a3. Sometimes, observations like these, based on examining deductions in a proof, can provide insight in the context of how the reasoning progresses. A comparison of using and not using binary resolution may, just possibly, lead to something useful. Again, Overbeek would assert binary resolution need not be used.

Before ending this part of the journey and turning to a new country, one additional experiment merits discussion. In that experiment, the main goal was to prove two theorems from Szmielew that offer difficulty and resistance.

-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd). % Satz 3.71
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd). % Satz 3.72

Their resistance to proof caused me to make one change to a preceding experiment, namely, the adjoining of the following three lemmas.

T(x,y,y).
T(x,x,y).
-T(x,y,z) | T(z,y,x).

In the preceding experiment, which I will call R, one theorem, CHZ12, had been proved. In the current experiment, called R1, three theorems were proved: CHZ371, Ch372, and (again) CHZ12. At least superficially, the inclusion of the cited three lemmas made the difference, although the proof of CHZ12 in R1 was the same as it was in experiment R. Indeed, CHZ71 was proved in 306 CPU-seconds with the retention of clause (375739) and with a length of 8. The theorem CH372 was proved in 64,109 CPU-seconds with retention of clause (3498451) and with a proof of length 13. For those who enjoy extremes, the preceding citation is impressive; but the following establishes a real high, showing how robust OTTER is. The proof of CHZ12 that was presented was completed in 326,455 CPU-seconds with retention of clause (8544653) and with a proof length of 6, as before.

I drew two conclusions about the inclusion of the three lemmas in R1 when compared with R. First, their presence enabled the program to produce two proofs that had resisted completion, which is as it should be. Second, the inclusion of the three lemmas caused the program to consume much, much more CPU time. Indeed, the earlier 6-step proof of CHZ12 was found in 6,144 CPU-seconds with retention of clause (966407) and with a proof length of 6.

Axiom Modification

I am now about to take (in the next section) a significant change in direction, one that requires replacement of some of the Tarski axioms by others. You could pause here for a long, long time and decide to consider, perhaps again, the challenges offered earlier in this notebook. You could proceed with or without the aid of an automated reasoning program. You might prefer some other program than OTTER. For any of the challenges, you might or might not include lemmas or might (in effect) include, perhaps with resonators, a so-called outline of the proof being sought. You might, for your choice of inference rules, keep in mind the failures of those decades ago and contrast them with some of the successes cited so far. With OTTER, in particular, choices must be made about options and parameter values. And, not the least of the possible obstacles or difficulties to be faced, you are asked to choose from among the three types of proof: forward, backward, and bidirectional. Despite this perhaps-dark view of how to proceed, its darkness is now misleading, when compared with the early 1980s. To provide more encouraging evidence, I now turn to an approach, axiomatically, that is taken from Szmielew to a great extent. These axioms are those studied jointly by Beeson and me.

6. Other Axioms, More Experiments

In place of the Tarski 13-axiom system offered in Section 1, one crucial change is made, one replacement. In place of the two clauses for Pasch, outer Pasch, two clauses for inner Pasch are included.

```
% Outer Pasch's axiom (2 clauses):
% -T(x,v,u) | -T(y,u,z) | T(x,f1(v,x,y,z,u),y).
% -T(x,v,u) | -T(y,u,z) | T(z,v,f1(v,x,y,z,u)).
% A7, inner Pasch, two clauses, replacing preceding two for outer
-T(x,v,u) | -T(y,z,u) | T(v,f1(x,v,u,y,z),y).
-T(x,v,u) | -T(y,z,u) | T(z,f1(x,v,u,y,z),x).
```

For a small clue that might give you some insight regarding inner Pasch, first draw an equilateral triangle with a vertex at the top. Then remove the line at the bottom, and draw two additional lines, one from the bottom left to some point on the right side (perhaps the midpoint), and one from the bottom right to some point on the left side (perhaps the midpoint). Inner Pasch asserts the existence of a point that is the intersection of the two lines.

Certain experiments virtually demand to be made in the context of the modified axiom set, say, with the following input file that will remind you of A Most Useful Input file.

Another Most Useful Input File

```
assign(max_weight,22).
assign(max_proofs,20).
set(hyper_res).
set(input_sos_first).
clear(order_hyper).
clear(print_kept).
set(para_into).
set(para_from).
set(ur_res).
% set(binary_res).
set(unit_deletion).
```

```
weight_list(pick_and_purge).
weight(f4(*(0),(0),*(0),*(0)),1).
end_of_list.

list(sos).
x = x.
ac1 != ac2.
T(ac1,ac2,ac3).
E(ac2,ac3,a4,a5).
E(b1,b2,b3,b4).
T(a7,a8,a9).
T(aa,ab,ac).
T(aa1,ab1,ac1).
T(da,db,dd).
T(db,dc,dd).
T(ba,bb,bc).
T(ba1,bb1,bc1).
T(a7,a8,a9).
E(aa,ab,aa1,ab1).
E(ab,ac,ab1,ac1).
E(eb,ec,eb1,ec1).
E(ba,bb,ba1,bb1).
E(bb,bc,bb1,bc1).
E(d,e,d3,d3).
E(b3,b4,b5,b6).
E(b1,b2,b3,b4).
T(da,db,dc).
T(db,da,dc).
T(a1,a2,a5).
T(a1,a4,a5).
T(a2,a3,a4).
T(A2,a6,a4).
T(a,b,c).
T(cba,cbb,cbc).
T(cbb,cbc,cbd).
T(fa,fb,fd).
T(fb,fc,fd).
T(ga,gb,gc).
T(ga,gc,gd).
T(gga,ggb,ggc).
T(gga,ggc,ggd).
T(ggga,gggb,gggc).
T(gggb,gggc,gggd).
T(cba,cbb,cbc).
T(cbb,cbc,cbd).
cbb != cbc.
T(ggga,gggb,gggc).
T(gggb,gggc,gggd).
gggb != gggc.
T(gga,ggb,ggc).
T(gga,ggc,ggd).
T(ga,gb,gc).
T(ga,gc,gd).
```

```
T(fa,fb,fd).
T(fb,fc,fd).
T(a,b,c).
% following 20 are translations of first 20 from ch6 of book2 for Tarski
-T(x,y,x) | (x = y).
% -T(x,y,u) | -T(y,z,u) | T(x,y,z).
% -T(x,y,z) | -T(x,y,u) | (x = y) | T(x,z,u) | T(x,u,z).
E(x,y,y,x).
-E(x,y,z,z) | (x = y).
-E(x,y,z,u) | -E(x,y,v,w) | E(z,u,v,w).
% -T(x,v,u) | -T(y,u,z) | T(x,f1(v,x,y,z,u),y).
% -T(x,v,u) | -T(y,u,z) | T(z,v,f1(v,x,y,z,u)).
% A7, inner Pasch, two clauses, replacing preceding two for outer
-T(x,v,u) | -T(y,z,u) | T(v,f1(x,v,u,y,z),y).
-T(x,v,u) | -T(y,z,u) | T(z,f1(x,v,u,y,z),x).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,z,f2(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,y,f3(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(f2(v6,x,y,z,u),v6,f3(v6,x,y,z,u)).
-E(x1,y1,x2,y2) | -E(y1,z1,y2,z2) | -E(x1,u1,x2,u2) | -E(y1,u1,y2,u2) | -T(x1,y1,z1) |
 -T(x2,y2,z2) | (x1 = y1) | E(z1,u1,z2,u2).
T(x,y,f4(x,y,u,v)).
E(y,f4(x,y,u,v),u,v).
-T(c1,c2,c3).
-T(c2,c3,c1).
-T(c3,c1,c2).
-E(x,u,x,v) | -E(y,u,y,v) | -E(z,u,z,v) | (u = v) | T(x,y,z) | T(y,z,x) | T(z,x,y).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | E(u1,y1,u1,f5(x1,y1,z1,x2,z2,u1)).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | T(x2,f5(x1,y1,z1,x2,z2,u1),z2).
% T(x,y,y).
% T(x,x,y).
% -T(x,y,z) | T(z,y,x).
end_of_list.
list(sos).
T(a2,a3,a4).
T(a1,a2,a5).
T(a1,a4,a5).
T(A2,a6,a4).
end_of_list.

list(passive).
f4(ac1,ac2,a4,a5) != ac3 | $ANS(CHZ12).
-T(cba,cbb,cbd) | $ANS(CH372).
-T(ggga,gggc,gggd) | $ANS(CHZ371).
-T(gga,ggb,ggd) | $ANS(CHZ362).
-T(gb,gc,gd) | $ANS(CHZ361).
-T(ga,gb,gd) | $ANS(CHZ361a).
-T(fa,fc,fd) | $ANS(CHZ352).
-T(c,b,a) | $ANS(CHZ32).
can != cb | $ANS(CHZ34).
da != db | $ANS(CHZ34).
d != e | $ANS(CHZZZ).
-E(a7,a8,a7,a8) | $ANS(CHZ1).
-E(b1,b1,b2,b2) | $ANS(CHZ8).
```

```
-E(b1,b2,b4,b3) | $ANS(CHZ5).
-E(b1,b2,b5,b6) | $ANS(CHZ3).
-E(b2,b1,b3,b4) | $ANS(CHZ4).
-E(b3,b4,b1,b2) | $ANS(CHZ2).
-E(ba,bc,ba1,bc1) | $ANS(CHZ11).
-T(a1,a1,a2) | $ANS(CH2).
-T(a1,a2,a2) | $ANS(CH3).
-T(a9,a8,a7) | $ANS(CH4).
-T(da,db,dc) | $ANS(CHZ351).
% -T(a1,a3,a5) | $ANS(CH1).
% -T(a2,a3,a6) | $ANS(CH5A).
% -T(a2,a6,a3) | $ANS(CH5B).
% -T(a4,a3,a6) | $ANS(CH5C).
% -T(a4,a6,a3) | $ANS(CH5D).
end_of_list.
```

If you compare this input file with its earlier-offered cousin, in addition to replacing outer Pasch with inner, you will see that two clauses, for transitivity and connectivity of betweenness, have now been commented out.

```
-T(x,y,u) | -T(y,z,u) | T(x,y,z).
-T(x,y,z) | -T(x,y,u) | (x = y) | T(x,z,u) | T(x,u,z).
```

These two properties (in the presence of inner Pasch replacing outer Pasch), transitivity and connectivity of betweenness, can be proved to be dependent on the remaining items not commented out. (In contrast, with outer Pasch in place of inner Pasch, connectivity of betweenness is dependent, but transitivity of betweenness is not dependent.) I plan to include the appropriate proofs of the two dependencies, in the presence of inner Pasch. Now, if you wish yet additional challenges, you might try to prove the two dependencies on your own.

(For the careful and curious reader and the reader who wondered about earlier remarks about "thought-to-be dependent", you see that if you remove the axioms of transitivity and connectivity from the system relying on inner Pasch, because the two are dependent, no theorems are lost; but, since transitivity is not dependent when inner Pasch is replaced by outer Pasch, the removal of the two weakens the resulting axiom system. For greater clarity, until essentially this point in the development of this notebook, as so often noted in earlier sections, I thought, erroneously, that transitivity of betweenness was dependent in the 1959 system that relies on outer Pasch.) This latest input file also includes the three now-favorite lemmas, commented out, but to be featured in later experiments with the axiom system that relies on inner Pasch. (As an aside, I note that when Beeson and I made our study—a study that in fact continues—rather than the function f1, we used the function ip. A cursory examination suggests, and I do not know why, that the use of ip rather than f1 has OTTER find proofs a bit faster.)

Similar to what you might expect, I took the input file, preventing the program from relying on two dependent properties that have just been cited, and asked OTTER to prove a large number of theorems selected from Szmielew. The symmetry of betweenness is one of the theorems to be proved, whereas it was not considered with the Most Useful Input File. Also, the presence of CHZ351 is to be ignored for the moment because, if proved, the proof would be irrelevant in that its positive form is present in the file. With the given input file, OTTER produced the following.

```
----> UNIT CONFLICT at 0.00 sec ----> 100 [binary,99.1,84.1] $ANS(CHZZZ).
Length of proof is 1. Level of proof is 1.
----> UNIT CONFLICT at 0.00 sec ----> 105 [binary,104.1,85.1] $ANS(CHZ1).
Length of proof is 1. Level of proof is 1.
----> UNIT CONFLICT at 0.00 sec ----> 107 [binary,106.1,89.1] $ANS(CHZ4).
Length of proof is 1. Level of proof is 1.
----> UNIT CONFLICT at 0.02 sec ----> 452 [binary,451.1,90.1] $ANS(CHZ2).
Length of proof is 2. Level of proof is 2.
```

----> UNIT CONFLICT at 0.07 sec ----> 528 [binary,527.1,87.1] $ANS(CHZ5).
Length of proof is 3. Level of proof is 3.
----> UNIT CONFLICT at 0.07 sec ----> 530 [binary,529.1,88.1] $ANS(CHZ3).
Length of proof is 3. Level of proof is 3.
----> UNIT CONFLICT at 0.19 sec ----> 596 [binary,595.1,86.1] $ANS(CHZ8).
Length of proof is 2. Level of proof is 2.
----> UNIT CONFLICT at 0.19 sec ----> 598 [binary,597.1,93.1] $ANS(CH3).
Length of proof is 2. Level of proof is 2.
----> UNIT CONFLICT at 0.54 sec ----> 2491 [binary,2490.1,83.1] $ANS(CHZ34).
Length of proof is 6. Level of proof is 6.
----> UNIT CONFLICT at 2.72 sec ----> 16798 [binary,16797.1,92.1] $ANS(CH2).
Length of proof is 5. Level of proof is 5.
----> UNIT CONFLICT at 3.64 sec ----> 19923 [binary,19922.1,81.1] $ANS(CHZ32).
Length of proof is 5. Level of proof is 5.
----> UNIT CONFLICT at 5.10 sec ----> 29787 [binary,29786.1,94.1] $ANS(CH4).
Length of proof is 5. Level of proof is 5.
----> UNIT CONFLICT at 612.38 sec ----> 371593 [binary,371592.1,78.1] $ANS(CHZ361).
Length of proof is 11. Level of proof is 8.
----> UNIT CONFLICT at 8909.32 sec ----> 1312760 [binary,1312759.1,74.1] $ANS(CHZ12).
Length of proof is 6. Level of proof is 5.

Compared with the results obtained with A Most Useful Input File, the only new proof is that for CH4, the symmetry of betweenness. In the other direction, the proofs not found here, but found with the earlier useful file, are for CHZ361a, CHZ362, and CHZ352. As for the use of the function ip rather than f1, for an example of time difference, with the ip notation, CHZ12 was proved in approximately 8,379 CPU-seconds, again with retention of clause (1312759). From a cursory perusal, the proofs are the same for the ip and f1 usages.

Naturally you wonder, as I did, what would result from permitting OTTER to employ the two dependent clauses. Well, nothing of interest took place. Indeed, a cursory examination shows that the proofs are the same. However, the presence of the two dependent axioms, neither of which (apparently) participated in a proof of interest, caused the program to take substantially longer to complete the sought-after proofs. (An important word of warning, especially for the individual whose fascination produces possible haste: In an input file, you must be sure that the ground clauses, those free of variables, adjoined to prove a theorem are not misused to prove another theorem in the same run, where the fault can be with the use of constants that are not distinct enough from theorem to theorem.)

The next experiment includes the three now well-known lemmas. You would correctly predict, based on the earlier discussion in this notebook, that their presence could slow the program so much that some theorems proved when the lemmas are not present would become out of reach. You would also suggest that perhaps one of the targets not yet reached would be proved. Well, yes, CHZ371 was proved. However, in addition to not proving CH2 and CH3, which are present among the three included lemmas, CHZ12 was also not proved. Therefore, by way of a small summary, still to be proved are CHZ361a, CHZ362, CH372, and CHZ11, from among the targets considered earlier.

In that I had now a proof of CHZ371—whose proof steps I may have used in the next experiment, but I cannot be certain—I chose to find a proof of CH372 because of their similarity, as seen in the following.

-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd). % Satz 3.71
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd). % Satz 3.72

This experiment, designed to begin filling in missing pieces by seeking a proof of CH372, led to the discovery of a small amount of gold, as seen with the following.

----> UNIT CONFLICT at 0.37 sec ----> 4053 [binary,4052.1,80.1] $ANS(CH372).
Length of proof is 8. Level of proof is 5.
----> UNIT CONFLICT at 0.38 sec ----> 4243 [binary,4242.1,84.1] $ANS(CHZ361a).
Length of proof is 7. Level of proof is 5.

----> UNIT CONFLICT at 0.40 sec ----> 4429 [binary,4428.1,82.1] $ANS(CHZ362).
Length of proof is 7. Level of proof is 5.
----> UNIT CONFLICT at 0.41 sec ----> 4613 [binary,4612.1,85.1] $ANS(CHZ352).
Length of proof is 6. Level of proof is 5.

I was left at this point with the target of proving CHZ11.

When I looked back in this notebook to see what had occurred with this target, I realized that I could get a proof by modifying A Most Useful Input File. Besides commenting out the axioms of transitivity and connectivity of betweenness, the important moves were assigning the value 11 to max_weight and deleting perhaps twenty-one ground clauses that are irrelevant to proving CHZ11. That (first) proof, cited near the beginning of Section 3, that gem, was found only after the expenditure of more than 141,384 CPU-seconds. A 24-step proof was obtained in approximately 105 CPU-seconds, with retention of clause (8611). When the irrelevant ground clauses remained in the input, again CHZ11 was proved, but this time in approximately 758 CPU-seconds, with retention of clause (30323). As you would expect, the same 24-step proof was found, of course with different numbering.

A natural, and even pressing, question arises. With inner Pasch in place of outer Pasch, does one have a more useful and, in some important sense, a more powerful axiom system? The next section provides, perhaps, some insight in the context of this question.

7. Inner Pasch and Dependencies, More Experiments

You may have, all throughout the reading of this notebook, suspected that I was secretly after proofs of two dependencies, cited earlier in the context of betweenness, namely, transitivity and connectivity. You may have also conjectured that they have continually eluded me. You are indeed on target. Of course, as noted early in this notebook, I finally learned that transitivity of betweenness is *not* dependent on the axiom system in which outer Pasch is present, but connectivity is dependent. So, in the spirit of a seminar, I invite you to join me as I search for what might enable OTTER to produce the elusive proofs. The path at this point will focus on proving the two dependencies, rather than in the presence of outer Pasch, in the presence of inner Pasch. I will discuss different attacks that may, for some, add to insights for research in areas other than geometry. I already know, and you will learn shortly, of some possibly important items and aspects.

At this point in the expedition, to avoid writing an entire book, I will feature highlights of a long, long journey, leaving for you various challenges. You will be witnessing research as it occurred. My first goal was to prove transitivity of betweenness dependent on the other Tarski axioms, omitting connectivity, which eventually also became a target for proving dependency. The experiments that produced proofs of a number of theorems from Szmielew immediately came into play, those relying on a Tarski system that included inner Pasch, but not outer Pasch, and that avoided the use of both transitivity and connectivity of betweenness. Indeed, I adjoined the conclusion (so to speak) of each of the proved theorems, which included, among various others, as you will see in an input file I will supply, symmetry of betweenness and CHZ361, the following.

-T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd). % Satz 3.61.

To revisit a tiny pedantic point, the negative unit clause (by negating a nonunit clause) that was used to complete a proof was, of course, not that which was included; the inclusion, as shown, was the entire nonunit clause that was negated to seek a proof.

In less than 7 CPU-seconds and the retention of less than 1,100 new conclusions, OTTER presented me with a 4-step proof of the dependence of transitivity of betweenness in the presence of inner Pasch. Just two items were used from the input, other than those resulting from the denial of the theorem, namely, CH4 (symmetry of betweenness) and CHZ361. To be careful, or to yield to paranoia, I verified that the proofs of these two theorems were obtained without the use of either transitivity or connectivity of betweenness. I now offer the input file that was used and the proof found with its use.

A Key Input File Focusing on Inner Pasch and Transitivity

```
assign(max_weight,23).
assign(max_proofs,10).
assign(max_distinct_vars,4).
assign(pick_given_ratio,4).
set(hyper_res).
set(input_sos_first).
clear(order_hyper).
clear(print_kept).
set(para_into).
set(para_from).
set(ur_res).
set(binary_res).
set(unit_deletion).

weight_list(pick_and_purge).
weight(f4(*(0),(0),*(0),*(0)),1).
end_of_list.

list(sos).
T(aaa,aab,aad).
T(aab,aac,aad).
% IFS(hjha,hjhb,hjhc,hjhd,hjha1,hjhb1,hjhc1,hjhd1).
% -E(hjhb,hjhd,hjhb1,hjhd1) | $ANS(CHZ42).
x = x.
% following 20 are translations of first 20 from ch6 of book2 for Tarski
-T(x,y,x) | (x = y).
% -T(x,y,u) | -T(y,z,u) | T(x,y,z).
% -T(x,y,z) | -T(x,y,u) | (x = y) | T(x,z,u) | T(x,u,z).
E(x,y,y,x).
-E(x,y,z,z) | (x = y).
-E(x,y,z,u) | -E(x,y,v,w) | E(z,u,v,w).
% -T(x,v,u) | -T(y,u,z) | T(x,f1(v,x,y,z,u),y).
% -T(x,v,u) | -T(y,u,z) | T(z,v,f1(v,x,y,z,u)).
% A7, inner Pasch, two clauses, replacing preceding two for outer
-T(x,v,u) | -T(y,z,u) | T(v,f1(x,v,u,y,z),y).
-T(x,v,u) | -T(y,z,u) | T(z,f1(x,v,u,y,z),x).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,z,f2(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,y,f3(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(f2(v6,x,y,z,u),v6,f3(v6,x,y,z,u)).
-E(x1,y1,x2,y2) | -E(y1,z1,y2,z2) | -E(x1,u1,x2,u2) | -E(y1,u1,y2,u2) |
 -T(x1,y1,z1) | -T(x2,y2,z2) | (x1 = y1) | E(z1,u1,z2,u2).
T(x,y,f4(x,y,u,v)).
E(y,f4(x,y,u,v),u,v).
-T(c1,c2,c3).
-T(c2,c3,c1).
-T(c3,c1,c2).
-E(x,u,x,v) | -E(y,u,y,v) | -E(z,u,z,v) | (u = v) | T(x,y,z) | T(y,z,x) | T(z,x,y).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | E(u1,y1,u1,f5(x1,y1,z1,x2,z2,u1)).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | T(x2,f5(x1,y1,z1,x2,z2,u1),z2).
end_of_list.

list(sos).
```

```
% Following proved so far, from out5s2q6dd5sss and ss2.
E(x,y,x,y).  % Satz 2.1
-E(xa,xb,xc,xd) | E(xc,xd,xa,xb).  % Satz 2.2
-E(xa,xb,xc,xd) | -E(xc,xd,xe,xf) | E(xa,xb,xe,xf).  % Satz 2.3
-E(xa,xb,xc,xd) | E(xb,xa,xc,xd).  % Satz 2.4
-E(xa,xb,xc,xd) | E(xa,xb,xd,xc).  % Satz 2.5
E(x,x,y,y).  % Satz 2.8
xq = xa | -T(xq,xa,u) | -E(xa,u,xc,xd) | ext(xq,xa,xc,xd) = u.    % Satz 2.12
T(x,y,y).    % Satz 3.1
-T(xa,xb,xc) | T(xc,xb,xa).  % Satz 3.2.
T(xa,xa,xb).  % Satz 3.3
-T(xa,xb,xc) | -T(xb,xa,xc) | xa = xb.  % Satz 3.4.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd).  % Satz 3.61.
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd).  % Satz 3.71
% following proved in temp.beeson.five.point.out5s2q6dd5sss4
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xc,xd).  % Satz 3.52.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd).  % Satz 3.61.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd).  % Satz 3.62.
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd).  % Satz 3.72
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xb,xa1,xb1) | -E(xb,xc,xb1,xc1) | E(xa,xc,xa1,xc1).  % Satz 2.11
end_of_list.

list(sos).
%  T(a2,a3,a4).
%  T(a1,a2,a5).
%  T(a1,a4,a5).
%  T(A2,a6,a4).
end_of_list.

list(passive).
-T(aaa,aab,aac)|$ANS(TR).
T(fa,fb,fd).
T(fb,fc,fd).
f4(ac1,ac2,a4,a5) != ac3 | $ANS(CHZ12).
-T(cba,cbb,cbd) | $ANS(CH372).
-T(ggga,gggc,gggd) | $ANS(CHZ371).
-T(gga,ggb,ggd) | $ANS(CHZ362).
-T(gb,gc,gd) | $ANS(CHZ361).
-T(ga,gb,gd) | $ANS(CHZ361a).
-T(fa,fc,fd) | $ANS(CHZ352).
%  -T(c,b,a) | $ANS(CHZ32).
%   can != cb | $ANS(CHZ34).
%  da != db | $ANS(CHZ34).
%  d != e | $ANS(CHZZZ).
%  -E(a7,a8,a7,a8) | $ANS(CHZ1).
%  -E(b1,b1,b2,b2) | $ANS(CHZ8).
%  -E(b1,b2,b4,b3) | $ANS(CHZ5).
%  -E(b1,b2,b5,b6) | $ANS(CHZ3).
%  -E(b2,b1,b3,b4) | $ANS(CHZ4).
%  -E(b3,b4,b1,b2) | $ANS(CHZ2).
%  -E(ba,bc,ba1,bc1) | $ANS(CHZ11).
%  -T(a1,a1,a2) | $ANS(CH2).
%  -T(a1,a2,a2) | $ANS(CH3).
```

```
% -T(a9,a8,a7) | $ANS(CH4).
% -T(da,db,dc) | $ANS(CHZ351).
% % -T(a1,a3,a5) | $ANS(CH1).
% % -T(a2,a3,a6) | $ANS(CH5A).
% % -T(a2,a6,a3) | $ANS(CH5B).
% % -T(a4,a3,a6) | $ANS(CH5C).
% % -T(a4,a6,a3) | $ANS(CH5D).
end_of_list.
```

A 4-Step Proof of Transitivity of Betweenness in the Presence of Inner Pasch

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Thu Feb  7 09:13:06 2013
The command was "otter".  The process ID is 30324.
----> UNIT CONFLICT at   0.12 sec ----> 785 [binary,784.1,40.1] $ANS(TR).

Length of proof is 4.  Level of proof is 3.

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

1 [] T(aaa,aab,aad).
2 [] T(aab,aac,aad).
30 [] -T(xa,xb,xc)|T(xc,xb,xa).
33 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xb,xc,xd).
40 [] -T(aaa,aab,aac)|$ANS(TR).
186 [binary,30.1,2.1] T(aad,aac,aab).
187 [binary,30.1,1.1] T(aad,aab,aaa).
453 [hyper,187,33,186] T(aac,aab,aaa).
784 [binary,453.1,30.1] T(aaa,aab,aac).
785 [binary,784.1,40.1] $ANS(TR).
```

The input file illustrates the use of lemma adjunction; indeed, when a lemma or theorem is used, adjoining it to list(usable) or list(sos) is in the spirit of learning and can be the difference between success and failure in the next experiment. That file also shows how, say, with Satz 2.11, you must adjoin the nonunit and not be content with the termination of a proof that uses just the negated literal. The proof nicely shows how the earlier theorems proved were used, namely, 3.61 and that for symmetry of betweenness. No other axioms of Tarski were used for this proof; but, of course, they were used in the proofs of the two cited theorems.

The story of my journey, fictitiously titled Traveling through Tarski, next offered three routes to follow: (1) experiments with other approaches to proving the theorem under discussion; (2) experiments to find a proof of transitivity in the presence of outer Pasch; or (3) experiments concerned with connectivity of betweenness in the presence of inner Pasch. I chose the first of the three paths, in part because it is the easiest to follow, as you will see.

The most obvious change to make was to avoid the use of binary resolution, as you have witnessed at various times in this notebook. The same 4-step proof of transitivity was produced. The main difference was the use of clause (541) rather than clause (784) to complete the proof; also, the CPU time was almost cut in half. Although the cited differences in this case are insignificant, when you tackle theorems that prove to be much harder to prove, you may find such citations to be of value.

Expecting that I might fail, I turned to another experiment, another approach, to seeking a proof of transitivity. The reason for my expectation was that I decided to avoid using any theorems I had proved with the use of inner Pasch. In other words, I asked OTTER to prove transitivity from the remaining Tarski axioms, including inner Pasch, but excluding connectivity of betweenness. I allowed binary resolution to be used and chose to rely on Veroff’s hints strategy, as you see in the following input file.

An Input File Relying on Hints

```
assign(max_weight,23).
assign(max_proofs,10).
assign(max_distinct_vars,4).
assign(pick_given_ratio,4).
assign(bsub_hint_wt,1).
set(keep_hint_subsumers).
set(hyper_res).
set(input_sos_first).
clear(order_hyper).
clear(print_kept).
set(para_into).
set(para_from).
set(ur_res).
set(binary_res).
set(unit_deletion).

weight_list(pick_and_purge).
weight(f4(*(0),(0),*(0),*(0)),1).
end_of_list.

list(sos).
T(aaa,aab,aad).
T(aab,aac,aad).
IFS(hjha,hjhb,hjhc,hjhd,hjha1,hjhb1,hjhc1,hjhd1).
x = x.
ac1 != ac2.
T(ac1,ac2,ac3).
T(a7,a8,a9).
T(aa,ab,ac).
T(aa1,ab1,ac1).
T(ba,bb,bc).
T(ba1,bb1,bc1).
T(a7,a8,a9).
E(ba,bb,ba1,bb1).
E(bb,bc,bb1,bc1).
T(a1,a2,a5).
T(a1,a4,a5).
T(a2,a3,a4).
T(A2,a6,a4).
T(a,b,c).
T(fa,fb,fd).
T(fb,fc,fd).
cbb != cbc.
gggb != gggc.
T(fa,fb,fd).
T(fb,fc,fd).
T(a,b,c).
%  following 20 are translations of first 20 from ch6 of book2 for Tarski
```

```
-T(x,y,x) | (x = y).
%  -T(x,y,u) | -T(y,z,u) | T(x,y,z).
%  -T(x,y,z) | -T(x,y,u) | (x = y) | T(x,z,u) | T(x,u,z).
E(x,y,y,x).
-E(x,y,z,z) | (x = y).
-E(x,y,z,u) | -E(x,y,v,w) | E(z,u,v,w).
%  -T(x,v,u) | -T(y,u,z) | T(x,f1(v,x,y,z,u),y).
%  -T(x,v,u) | -T(y,u,z) | T(z,v,f1(v,x,y,z,u)).
%  A7, inner Pasch, two clauses, replacing preceding two for outer
-T(x,v,u) | -T(y,z,u) | T(v,f1(x,v,u,y,z),y).
-T(x,v,u) | -T(y,z,u) | T(z,f1(x,v,u,y,z),x).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,z,f2(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,y,f3(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(f2(v6,x,y,z,u),v6,f3(v6,x,y,z,u)).
-E(x1,y1,x2,y2) | -E(y1,z1,y2,z2) | -E(x1,u1,x2,u2) | -E(y1,u1,y2,u2) |
 -T(x1,y1,z1) | -T(x2,y2,z2) | (x1 = y1) | E(z1,u1,z2,u2).
T(x,y,f4(x,y,u,v)).
E(y,f4(x,y,u,v),u,v).
-T(c1,c2,c3).
-T(c2,c3,c1).
-T(c3,c1,c2).
-E(x,u,x,v) | -E(y,u,y,v) | -E(z,u,z,v) | (u = v) | T(x,y,z) | T(y,z,x) | T(z,x,y).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | E(u1,y1,u1,f5(x1,y1,z1,x2,z2,u1)).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | T(x2,f5(x1,y1,z1,x2,z2,u1),z2).
end_of_list.

list(sos).
% %  Following sorted proofs steps, 65 of them, taken from proofs from sss8 and sss
%  a8=f1(a8,a9,a9,a7,a8).
%  ba1=bb1|E(bc1,ba1,bc,ba).
%  ba1=bb1|E(bc,ba,bc1,ba1).
%  ba=bb|E(bc,ba,bc1,ba1).
%  bb1=ba1|E(bc,ba,bc1,ba1).
%  bc1=ba1|E(bc,ba,bc1,ba1).
%  bc1=bb1|E(ba1,bc1,ba,bc).
%  bc1=bb1|E(ba1,bc1,bc,ba).
%  bc1=bb1|E(bc,ba,bc1,ba1).
%  b=f1(b,c,c,a,b).
%  da=db.
%  da=f1(db,da,dc,da,db).
%  db=da.
%  db=f1(db,da,db,x,db).
%  db=f1(db,da,dc,da,db).
%  d=e.
%  f4(ac1,ac2,a4,a5)=ac3.
%  gb=f1(gb,gc,gc,ga,gb).
%  gc=f1(gc,gb,ga,gd,gc).
%  gc=f1(gc,gd,gd,ga,gc).
%  x=f1(x,x,x,y,x).
%  x=f4(y,x,z,z).
%  -T(x,y,z) | T(z,y,x).
% %  Following proved so far, from out5s2q6dd5sss and ss2.
% %  following proved in temp.beeson.five.point.out5s2q6dd5sss4
```

```
% % The following many clauses are Definition 4.1
% -IFS(xa,xb,xc,xd,za,zb,zc,zd) | T(xa,xb,xc).
% -IFS(xa,xb,xc,xd,za,zb,zc,zd) | T(za,zb,zc).
% -IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xa,xc,za,zc).
% -IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xb,xc,zb,zc).
% -IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xa,xd,za,zd).
% -IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xc,xd,zc,zd).
%  -T(xa,xb,xc) | -T(za,zb,zc) | -E(xa,xc,za,zc) | -E(xb,xc,zb,zc) |
   -E(xa,xd,za,zd) | -E(xc,xd,zc,zd) | IFS(xa,xb,xc,xd,za,zb,zc,zd).
% % Following 4 are definition 4.4 for n=3
% -E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa1,xa2,xb1,xb2).
% -E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa1,xa3,xb1,xb3).
% -E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa2,xa3,xb2,xb3).
% -E(xa1,xa2,xb1,xb2) | -E(xa1,xa3,xb1,xb3) | -E(xa2,xa3,xb2,xb3) | E3(xa1,xa2,xa3,xb1,xb2,xb3).
% jjja != jjjb.
% Col(jjja,jjjb,jjjc).
end_of_list.

list(sos).
end_of_list.

list(passive).
-T(aaa,aab,aac)|$ANS(TR).
end_of_list.

list(hints).
% Following sorted 14 from out5s2q6dd5sss, prove 3.2 and 361
b=f1(b,c,c,a,b).
gb=f1(gb,gc,gc,ga,gb).
gc=f1(gc,gb,ga,gd,gc).
gc=f1(gc,gd,gd,ga,gc).
T(b,f1(x,c,c,a,b),x).
T(c,b,a).
T(gb,f1(x,gc,gc,ga,gb),x).
T(gb,gc,gd).
T(gc,f1(gc,gb,ga,gd,gc),gc).
T(gc,f1(x,gd,gd,ga,gc),x).
T(gc,gb,ga).
T(gd,gc,ga).
T(x,y,y).
x=f4(y,x,z,z).
end_of_list.
```

The use of hints does not replace and is not equivalent to the use of resonators. Indeed, in contrast to a resonator that treats all variables as indistinguishable and, therefore, focuses on equivalence classes of formulas or equations, a hint treats the variables precisely as written and focuses on items that are identical to the hint (which, of course, includes alphabetic variants), subsume the hint, or are subsumed by the hint, depending on the included options. In general, the hints strategy works faster than does the resonance strategy, probably because far fewer deduced conclusions match a given hint than match a given resonator. The option set(keep_hint_subsumers) often is useful as well as the option assign(bsub_hint_wt,2) (or some small value). The hints you find in the input file were taken from earlier unsuccessful runs.

The experiment was, to me, somewhat of a surprise, which you can appreciate as you read the following.

A Proof of Transitivity with Hints and Inner Pasch

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Thu Feb 7 11:51:27 2013
The command was "otter". The process ID is 31511.
----> UNIT CONFLICT at 1.39 sec ----> 8999 [binary,8998.1,45.1] $ANS(TR).

Length of proof is 8. Level of proof is 5.

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

1 [] T(aaa,aab,aad).
2 [] T(aab,aac,aad).
27 [] -T(x,y,x)|x=y.
29 [] -E(x,y,z,z)|x=y.
31 [] -T(x,v,u)| -T(y,z,u)|T(v,f1(x,v,u,y,z),y).
32 [] -T(x,v,u)| -T(y,z,u)|T(z,f1(x,v,u,y,z),x).
37 [] T(x,y,f4(x,y,u,v)).
38 [] E(y,f4(x,y,u,v),u,v).
45 [] -T(aaa,aab,aac)|$ANS(TR).
111 [binary,31.3,27.1] -T(x,y,z)| -T(y,u,z)|y=f1(x,y,z,y,u).
134 [hyper,31,1,2] T(aab,f1(aaa,aab,aad,aab,aac),aab).
407 [binary,38.1,29.1] x=f4(y,x,z,z).
419 [para_from,407.1.2,37.1.3] T(x,y,y).
2974 [binary,134.1,27.1] aab=f1(aaa,aab,aad,aab,aac).
2999 [para_from,2974.1.2,32.3.2,unit_del,1,2] T(aac,aab,aaa).
6015 [hyper,111,2999,419] aab=f1(aac,aab,aaa,aab,aaa).
8998 [para_from,6015.1.2,32.3.2,unit_del,2999,419] T(aaa,aab,aac).
8999 [binary,8998.1,45.1] $ANS(TR).

Six axioms were used, as you see, in addition to the three clauses that arise from negating transitivity. In particular, both clauses for inner Pasch were relied upon.

The next called-for experiment was to repeat the preceding one but with an avoidance of binary resolution. Because of results already cited, you might, as I did, expect that the 8-step proof just presented would be duplicated. It was not, as you now see.

A Second Proof of Transitivity with Hints and Inner Pasch

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Thu Feb 7 11:57:33 2013
The command was "otter". The process ID is 31575.
----> UNIT CONFLICT at 6.13 sec ----> 21269 [binary,21268.1,45.1] $ANS(TR).

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

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

1 [] T(aaa,aab,aad).
2 [] T(aab,aac,aad).
27 [] -T(x,y,x)|x=y.
29 [] -E(x,y,z,z)|x=y.
31 [] -T(x,v,u)| -T(y,z,u)|T(v,f1(x,v,u,y,z),y).

32 [] -T(x,v,u)| -T(y,z,u)|T(z,f1(x,v,u,y,z),x).
37 [] T(x,y,f4(x,y,u,v)).
38 [] E(y,f4(x,y,u,v),u,v).
45 [] -T(aaa,aab,aac)|$ANS(TR).
96 [hyper,31,1,2] T(aab,f1(aaa,aab,aad,aab,aac),aab).
234 [hyper,38,29] x=f4(y,x,z,z).
239 [para_from,234.1.2,37.1.3] T(x,y,y).
1442 [hyper,96,27] aab=f1(aaa,aab,aad,aab,aac).
1465 [para_from,1442.1.2,32.3.2,unit_del,1,2] T(aac,aab,aaa).
1493 [hyper,1465,32,239] T(aab,f1(x,aaa,aaa,aac,aab),x).
21121 [hyper,1493,27] aab=f1(aab,aaa,aaa,aac,aab).
21268 [para_from,21121.1.2,31.3.2,unit_del,239,1465] T(aaa,aab,aac).
21269 [binary,21268.1,45.1] $ANS(TR).

One last experiment with transitivity was made. I tried repeating the preceding experiment with but one change: I replaced inner Pasch with outer Pasch. Of course, no proof was found; after all, as I learned well after this experiment failed, transitivity of betweenness is independent in the Tarski axiom system that employs outer Pasch. You might indeed be puzzled at my inclusion of this result. Well, the experiment illustrates the type of research that is merited, if you do not know all the facts, as I did not at the time. In particular, axiom replacement to produce a related theorem is worth investigation.

8. Winning a Big Prize

In this section, you are about to embark on a long, long trip; a glass of good wine might indeed be in order. The nature of this prize is a proof, a proof of one of the most challenging theorems to prove in Tarskian geometry. The theorem says that in the presence of inner Pasch (in place of outer Pasch) and the other axioms of Tarskian geometry given in Section 1, you can prove that connectivity of betweenness is a dependent axiom. This theorem was one of Art Quaife's four challenge problems. Of course, you are welcome to accept this challenge before reading the material of this section. However, required of me is the historical fact that Beeson and I spent a long, long time before we succeeded with OTTER. To succeed, we relied in part on Szmielew, in part on Beeson alone, in part on me alone, and in part on the joint effort of Beeson and me. Before presenting the first approach that produced a proof, a brief discussion of proof checking versus proof finding is in order.

The work of Szmielew up through Chapter 12 had been thoroughly proof checked in Coq by Narboux et al.: quite a feat. In proof checking, as you may already know, you take the author's proof and show that it is indeed a proof. You may be forced, or your program may be forced, to insert a few steps here and there. In proof finding, you do not use the steps offered by the author, but, instead, present the theorem to be proved by your program, with possibly various so-called helpers (as will be discussed). Although proof checking can present numerous obstacles, in general, researchers assert, proof finding is far more difficult. If success occurs with proof finding, the resulting proof can, more than occasionally, be quite different from the proof of the author in focus. Now to the approach taken by Beeson and me, his idea.

The Beeson approach, from the beginning, was to prove one theorem after another and work our way through Szmielew from Chapter 1 on. As we proved each theorem, we would add that theorem in the next set of experiments, designed to move forward. Of course, as definitions in Szmielew occurred, they would also be adjoined. We relied essentially on the notation used by Szmielew, which, in this section, I will also. Therefore, in place of the function *f1* occurring in earlier sections, I will use the function *ip*. Also, in place of the function *f4*, I will use the function *ext*. The use of both *ip* and *ext* may aid you in keeping track of what is occurring in the approach and in proofs.

One of the early and key decisions to make regarded the type of proof to seek. As you have read earlier, even with a bidirectional or backward proof in hand, the completion of a forward proof can present a monumental obstacle. Recall that in a forward proof, the denial (of the theorem to be proved), which may consist of one or more clauses as in denying or negating a nonunit clause, is not used during the search to draw conclusions and is placed in list(passive). More accurately, some of the units that result from denying a nonunit clause often do participate in the drawing of conclusions, for example, those that are positive. As an illustration, if you were seeking a proof of transitivity of betweenness, the following clause, you would have three unit clauses that result from denying it, from its negation.

```
-T(x,y,u) | -T(y,z,u) | T(x,y,z).
```

They might be T(a,b,d), T(b,c,d), and -T(a,b,c). In a search for a forward proof, ordinarily the third of the cited three would be placed in list(passive); the other two, usually, would be placed in list(sos). So, often, if the goal is to *find* a proof, any proof, the choice that offers the greatest chance of success is that of a bidirectional search. Indeed, the negative unit from a unit-represented theorem or the negative unit clause (or clauses) from the denying of a nonunit clause can play a significant role. For our attempt to win the big prize, we therefore chose to seek a bidirectional proof.

The input file that produced the first proof consists of more than eight hundred lines. Rather than displaying this monster, I will discuss its important aspects, stating where Szmielew contributed, where Beeson alone contributed, and the like. More than a proof of connectivity, you will be treated in this section to other delicacies. Proof finding, and not proof checking, was our study. If all goes as planned, I intend to eventually discuss differences between the Szmielew proof and the OTTER proof or proofs. Yes, I will discuss various proofs, after a while. (Later I will offer A Powerful Input File for the study of the theorem focusing on the dependence of connectivity of betweenness in the presence of inner Pasch. Before I do, I will discuss highlights of an earlier input file, supplied by Beeson, a discussion that will contribute to your understanding of research.)

Beeson presented me with an OTTER input file, a file that did not reason from all of the Tarskian axioms, but just the following.

```
E(x,y,y,x).    % A1 from page 10 of sst equidistance-reflexive
-E(x,y,z,v) | -E(x,y,z2,v2) | E(z,v,z2,v2).   % A2 trans-equidistance
-E(x,y,z,z) | x=y.   % A3 identity-equidist
T(x,y,ext(x,y,w,v)).   % A4, first half segment-construc
E(y,ext(x,y,w,v),w,v).   % A4, second half segment-construc
-E(x,y,x1,y1) | -E(y,z,y1,z1) | -E(x,v,x1,v1) | -E(y,v,y1,v1) |
 -T(x,y,z) | -T(x1,y1,z1) | x=y | E(z,v,z1,v1).
 % A5 five-segment
-T(x,y,x) | x=y.   % A6 identity-between
%  A7, inner Pasch, two clauses.
-T(xa,xp,xc) | -T(xb,xq,xc) | T(xp,ip(xa,xp,xc,xb,xq),xb).
-T(xa,xp,xc) | -T(xb,xq,xc) | T(xq,ip(xa,xp,xc,xb,xq),xa).
-T(alpha,beta,gamma).  %A8,  three lines. lower-dimension
-T(beta,gamma,alpha).
-T(gamma,alpha,beta).
```

His note to me said no more axioms would be needed, probably concluded from a reading of Szmielew. I am not certain, for we did not touch on every item. So, as you can check, some of the axioms were not included. On the other hand, the file I received included twenty-six clauses corresponding to theorems we had proved, mostly by Beeson, theorems from Szmielew covered before she turned to a proof of connectivity of betweenness. Various other sets of clauses were present, some from Szmielew definitions; so you see already how Szmielew and Beeson each contributed to the approach that was taken. But Beeson's contribution went much further; indeed, he included the following clauses, defined points, because of being certain that a diagram must be represented.

```
c1 = ext(a,d,c,d).
d1 = ext(a,c,c,d).
b1 = ext(a,c1,c,b).
b2 = ext(a,d1,d,b).
e = ip(c1,d,b,d1,c).
p = ext(e,c,c,d).
r = ext(d1,c,e,c).
```

```
q = ext(p,r,p,r).
```

So, you ask, how did I contribute?

Although I cannot be positive, I believe the following to be essentially accurate. I made many experiments before the one in focus. From those, I chose so-called intermediate targets, targets that, if proved, would suggest progress was occurring. Those targets (or the negated part after denial), in negated form, were placed in list(passive). It may have been my choice to place all the items from which deductions were to be made in a list(sos), rather than placing some or many in a list(usable), a list consulted for inference-rule completion, as opposed to inference-rule initiation. I may be the one who chose the options and parameter values, the following, to be discussed shortly.

```
set(hyper_res).
set(para_into).
set(para_from).
set(binary_res).
set(ur_res).
set(order_history).
assign(report,5400).
assign(max_mem,840000).
clear(print_kept).
set(input_sos_first).



assign(max_weight,20).
assign(max_distinct_vars,4).
assign(pick_given_ratio,4).
assign(max_proofs,10).
assign(heat,2).
```

A large set of resonators was included, items chosen from proof steps of earlier proofs, a set perhaps supplied in part by me and in part by Beeson; thus, joint effort was seen. Finally, the following two clauses were placed in list(hot); the reason for this choice I cannot recall.

```
list(hot).
-E(x,y,x1,y1) | -E(y,z,y1,z1) | -E(x,v,x1,v1) | -E(y,v,y1,v1) |
 -T(x,y,z) | -T(x1,y1,z1) | x=y | E(z,v,z1,v1).  % A5
AFS(d1,c,r,p,p,c,e,d1).
end_of_list.
```

The assignment of the value 2 to the heat parameter was chosen to have OTTER heavily and recursively consult list(hot). When compared with an assignment of 0, which means the hot list is totally ignored, the assignment of 2 enables a program to make various deductions far sooner than it would otherwise; however, often the latter assignment costs much in CPU time. As it turned out, heat was not used much, as seen by a glance at the output file, and used not at all in the proof that was eventually found after a long, long time.

The "set" commands that the input file relied upon included those for the use of hyperresolution, UR-resolution, paramodulation (both from and into), and, yes, binary resolution. An additional command, set(input_sos_first), was included to cause OTTER to choose, for inference-rule initiation, all the items in the initial set of support before choosing any deduce clause for that purpose. In general, a good move is to include this command when seeking a proof. Three assign commands were also chosen by me, in addition to the cited set commands, namely, that assigning a value of 20 to max_weight, that assigning a value of 4 to pick_given_ratio, and that assigning a value of 4 to max_distinct_vars. The assignment of values is in no way a science; instead, the art of assigning is based on experience and guesses. Here, the idea was to avoid drowning the program in newly kept conclusions and, at the same time, give it enough room so that a proof might be found. Success resulted, and a discussion of the proof is next in order. In that discussion, you will learn which theorems from Szmielew played a role, some of which I leave to you to prove as more challenges.

OTTER succeeded on January 11, 2013, returning to me a proof, of the dependence of connectivity of betweenness (based on inner Pasch), of length 104 (new conclusions) in approximately 17,126 CPU-seconds after retention of clause (429191). Only the following five Tarski axioms were used; of course, others were used in proving the theorems that will be cited from Szmielew.

1 [] E(x,y,y,x). % A1 reflex equidistance
2 [] -E(x,y,z,v)| -E(x,y,z2,v2)|E(z,v,z2,v2). % A2 transitivity equidistance
4 [] T(x,y,ext(x,y,w,v)). % A4, first half segment construction
5 [] E(y,ext(x,y,w,v),w,v). % A4, second half
6 [] -E(x,y,x1,y1)| -E(y,z,y1,z1)| -E(x,v,x1,v1)| -E(y,v,y1,v1)| -T(x,y,z)| -T(x1,y1,z1)|x=y|E(z,v,z1,v1).
% A5 five-segment

Of the twenty-six theorems of Szmielew found in the more-than-800-step input file, twenty, the following, were relied upon, as well as certain definitions.

14 [] -E(xa,xb,xc,xd)|E(xc,xd,xa,xb). % Satz 2.2
15 [] -E(xa,xb,xc,xd)|E(xb,xa,xc,xd). % Satz 2.4
16 [] -E(xa,xb,xc,xd)| -E(xc,xd,xe,xf)|E(xa,xb,xe,xf). % Satz 2.3
17 [] -E(xa,xb,xc,xd)|E(xa,xb,xd,xc). % Satz 2.5
18 [] E(x,x,y,y). % Satz 2.8
19 [] -T(xa,xb,xc)| -T(xa1,xb1,xc1)| -E(xa,xb,xa1,xb1)| -E(xb,xc,xb1,xc1)|E(xa,xc,xa1,xc1). % Satz 2.11
21 [] T(x,y,y). % Satz 3.1
23 [] -T(xa,xb,xc)|T(xc,xb,xa). % Satz 3.2.
24 [] T(xa,xa,xb). % Satz 3.3
25 [] -T(xa,xb,xc)| -T(xb,xa,xc)|xa=xb. % Satz 3.4.
26 [] -T(xa,xb,xd)| -T(xb,xc,xd)|T(xa,xb,xc). % Satz 3.51
27 [] -T(xa,xb,xd)| -T(xb,xc,xd)|T(xa,xc,xd). % Satz 3.52.
28 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xb,xc,xd). % Satz 3.61.
32 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xa,xb,xd). % Satz 3.62.
33 [] -T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xc,xd). % Satz 3.71
34 [] -T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xb,xd). % Satz 3.72
36 [] -T(xa,xb,xc)| -T(xa1,xb1,xc1)| -E(xa,xc,xa1,xc1)| -E(xb,xc,xb1,xc1)|E(xa,xb,xa1,xb1). % Satz 4.3
43 [] -IFS(xa,xb,xc,xd,za,zb,zc,zd)|T(za,zb,zc). % Part of Definition 4.1
(3 lines here; there are 7 lines in the definition)
46 [] -IFS(xa,xb,xc,xd,za,zb,zc,zd)|E(xa,xd,za,zd).
47 [] -IFS(xa,xb,xc,xd,za,zb,zc,zd)|E(xc,xd,zc,zd).
52 [] -E(xa1,xa2,xb1,xb2)| -E(xa1,xa3,xb1,xb3)| -E(xa2,xa3,xb2,xb3)|
E3(xa1,xa2,xa3,xb1,xb2,xb3). % Part of definition 4.4
57 [] -T(xa,xb,xc)| -E3(xa,xb,xc,xa1,xb1,xc1)|T(xa1,xb1,xc1). % Satz 4.6
59 [] Col(xa,xb,xc)| -T(xa,xb,xc). % half of Definition 4.10 (three lines)
60 [] Col(xa,xb,xc)| -T(xb,xc,xa).
61 [] Col(xa,xb,xc)| -T(xc,xa,xb).
76 [] xa=xb| -Col(xa,xb,xc)| -E(xa,xp,xa,xq)| -E(xb,xp,xb,xq)|E(xc,xp,xc,xq). Satz 4.17
78 [] -T(xa,xc,xb)| -E(xa,xc,xa,xc1)| -E(xb,xc,xb,xc1)|xc=xc1. Satz 4.19
87 [] -AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1)|T(xa,xb,xc). % part of Definition 2.10

(I must give you more information, eventually, if you intend to try to prove the theorems, as challenges, from 4.2 on.) The following five lemmas were used, each supplied by Beeson, but which he wished to avoid using as input; he might have simply conjectured, correctly, that they would be important.

96 [] AFS(b,c,d1,c1,b1,c1,d,c). % A lemma: TARG1INT
97 [] E(c1,d1,c,d). % TARGINT or TARG2INT
98 [] IFS(d,e,d1,c,d,e,d1,c1). % TARG3INT
99 [] IFS(c,e,c1,d,c,e,c1,d1). % TARG4INT

100 [] E(e,c,e,c1). % TARG5INT

(As you proceed further in this journey focusing on connectivity, you will revisit these five lemmas, motivated by Beeson.) For the denial of connectivity, the proof relied on the following.

115 [] a!=b. % negation of Satz 5.1, connectivity, the first of five lines
116 [] T(a,b,c).
117 [] T(a,b,d).
118 [] -T(a,c,d)|$ANS(CON).
119 [] -T(a,d,c)|$ANS(CON).

Finally—so many thanks to Beeson, again—five of his defined points were relied upon, the following.

120 [] c1=ext(a,d,c,d). % auxiliary points used to prove Satz 5.1, the first of five lines
121 [] d1=ext(a,c,c,d).
125 [] p=ext(e,c,c,d).
126 [] r=ext(d1,c,e,c).
127 [] q=ext(p,r,p,r).

Three additional points are found in the input file, the following, but were not used in this first proof.

b1 = ext(a,c1,c,b).
b2 = ext(a,d1,d,b).
e = ip(c1,d,b,d1,c).

Of the 104 deduced steps, 45 were obtained with binary resolution, 23 with hyperresolution, 10 with UR-resolution, 25 with paramodulation, and 1 with factoring. The proof is bidirectional; and, exciting it may be, I still do not have a forward proof of connectivity of betweenness in the presence of inner Pasch, although I do have a number of proofs that are not forward. Yes, you are presented with quite a challenge: Find a forward proof. You might wish to wait until you read more about these experiments.

Although I was indeed pleased that after so many, many experiments, we had a proof of connectivity of betweenness, Beeson was not so pleased, which I learned weeks later. Indeed, he informed me that the five lemmas used in the proof had not actually been proved, and therefore we did not yet have a proof. In other words, more work was required, namely, the finding of five proofs, followed by the seeking of a proof of connectivity that did not include, in the input file, the five lemmas, called, respectively, RG1 through RG5.

My approach was not particularly inventive. Indeed, I simply (essentially) took the input file that has just been discussed, that which yielded the 104-step proof, and commented out the five lemmas, and inserted their negations as targets. OTTER found a 40-step proof of the first of the five, TARG1INT, in approximately 473 CPU-seconds with retention of clause (109977). The proof was bidirectional, as was expected. To my annoyance, none of the other target lemmas was proved. Instead, the program produced, in total, eighteen proofs of the first lemma, stopping as it reached the assigned value to max_proofs. You see, one of the side effects that can, and often does, occur when seeking a bidirectional proof is the completion of one proof after another, of the same theorem. Perhaps contributing factors to the ease of finding a bidirectional proof as opposed to a forward proof are, first, such proofs so to speak meet in the middle because of reasoning forward and backward and, second, the level of such proofs is in general lower than that of the corresponding forward proof. On the other hand, when seeking a forward proof, this annoyance is in general avoided. Do you see what action to take to enable OTTER to avoid proving this first lemma again and, instead, seek one or more proofs of the other four target lemmas? Well, pause here if you wish to ponder the question. My approach was the following.

I took the input file that had found a proof (actually, eighteen proofs) of the first lemma and made two changes. I commented back in the positive form of the lemma and commented out its negation. In approximately 730 CPU-seconds, with retention of clause (148036), OTTER presented me with a 42-step proof of the second of the five lemmas. In fact, annoyance repeating itself, the program found eighteen proofs of this second lemma. Rather than immediately proceeding in the quest for proofs of Lemmas 3 through 5, I repeated the experiment that had yielded the proof of the second lemma, but with an assignment of the value 0 to heat, with the goal of finding proofs much faster. A good move indeed: With retention of clause (52976) and the use of approximately 57 CPU-seconds, OTTER completed a 47-step proof of the second lemma.

So, as you would accurately predict, I commented back in the second target and commented out its negation for the next run. The targets were all reached, in order, the fourth, the third, and the fifth. The respective proof lengths are 20, 16, and 20. Thus, consistent with Beeson's position that we had best not use the five lemmas in an input file whose goal was the proof of connectivity, which meant that proofs of each of the five must be in hand, this phase of the research was complete; we had the five proofs.

Before turning to the next big move, the next stop on this lengthy and most intriguing expedition, I pause to answer a question that might naturally arise at this point. How hard is it to find forward proofs of each of the five lemmas and thus avoid the side effect of focusing on bidirectional proofs? Well, earlier today, I did make a run to seek those five, possibly elusive, proofs. Indeed, one of the charming facets of notebook writing says that such experiments offer a break from writing and can proceed while more material is added to a section. so far, the first two (of the five) lemmas have been proved, proved with a forward proof.

So you must wish to know what I did with the proofs of the five lemmas, and wonder about the nature of the next key experiment. Before answering such questions, I feel it incumbent on me to adhere to history. In particular, I believe prompted by Beeson, I ran various experiments to verify a conjecture. Specifically, he and I had succeeded in finding proofs of diverse theorems by resorting to a case analysis. For example, where constants $d1$ and e were present, we had OTTER make two runs, one with $D1 = e$ and one with $D1$!= e. Quite often, both runs succeeded. Now, some researchers would be satisfied with the two proofs and maintain that the theorem in focus had been proved. Not Beeson. He intended that we find a way for OTTER to prove the theorem in a single run. I hypothesized that a possible approach, rather than running two cases, was to adjoin one new clause, namely, $D1 = e \mid D1$!= e, a tautology.

The theorem to prove for the experiment—and also the choice because we were after a proof that avoided the use of the five lemmas in the input—was, again, connectivity of betweenness in the presence of inner Pasch. OTTER succeeded, producing a 113-step proof, relying on thirty axioms and previously proved theorems, on eight defined points, on both clauses for inner Pasch, and, yes, on *two* tautologies. What was also of interest in the cited proof, among the already-proved theorems, was the presence of transitivity of betweenness. We will return to the significance of such an inclusion. The use of tautology adjunction was, for me, nostalgic; indeed, I had written about it as an approach to dealing with cases many decades ago, in the context of proving that subgroups of index 2 are normal. For possible research, Beeson (in effect) posed a problem: He thought, or hoped, that some way must exist for OTTER to avoid being forced to consider cases, a way other than tautology adjunction. And with this so-called interruption aside, the possible use of proofs of the five lemmas, RG1 through RG5, now takes center stage, and you learn that case analysis and tautology adjunction are in fact not needed to prove connectivity.

With the goal of obtaining a proof of connectivity of betweenness that does *not* rely on any of the five lemmas, RG1 through RG5, three moves were in order, as you will see from an input file I give shortly. First, I removed the five lemmas from the input file to be used. Second, I emphasized the use of hints rather than resonators, motivated by the wish to speed things up. Third, I placed in list(hints), in the file to be used, a sorted set of proof steps of proofs of the five lemmas, proofs obtained with cited experiments. After sorting to remove duplicates, I had a set of eighty-seven items to insert into a new list(hints).

A Powerful Input File for the Study of Connectivity in the Presence of Inner Pasch

```
set(hyper_res).
set(para_into).
set(para_from).
set(binary_res).
set(ur_res).
set(order_history).
```

```
assign(report,5400).
assign(max_mem,840000).
clear(print_kept).
%set(very_verbose).
set(input_sos_first).



assign(max_weight,11).
assign(max_distinct_vars,4).
assign(pick_given_ratio,4).
assign(max_proofs,10).
assign(heat,0).
assign(bsub_hint_wt,1).
set(keep_hint_subsumers).

weight_list(pick_and_purge).
end_of_list.



list(sos).
E(x,y,y,x).                         % A1 from page 10 of sst
-E(x,y,z,v) | -E(x,y,z2,v2) | E(z,v,z2,v2).   % A2
-E(x,y,z,z) | x=y.                  % A3
T(x,y,ext(x,y,w,v)).                % A4, first half
E(y,ext(x,y,w,v),w,v).              % A4, second half
-E(x,y,x1,y1) | -E(y,z,y1,z1) | -E(x,v,x1,v1) | -E(y,v,y1,v1) |
 -T(x,y,z) | -T(x1,y1,z1) | x=y | E(z,v,z1,v1).  % A5
-T(x,y,x) | x=y.                    % A6
-T(xa,xp,xc) | -T(xb,xq,xc) | T(xp,ip(xa,xp,xc,xb,xq),xb).
-T(xa,xp,xc) | -T(xb,xq,xc) | T(xq,ip(xa,xp,xc,xb,xq),xa).
-T(alpha,beta,gamma).  %A8,  three lines.
-T(beta,gamma,alpha).
-T(gamma,alpha,beta).
E(x,y,x,y).  % Satz 2.1
-E(xa,xb,xc,xd) | E(xc,xd,xa,xb).  % Satz 2.2
-E(xa,xb,xc,xd) | E(xb,xa,xc,xd).  % Satz 2.4
-E(xa,xb,xc,xd) | -E(xc,xd,xe,xf) | E(xa,xb,xe,xf).  %  Satz 2.3
-E(xa,xb,xc,xd) | E(xa,xb,xd,xc).  % Satz 2.5
E(x,x,y,y).  % Satz 2.8
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xb,xa1,xb1) | -E(xb,xc,xb1,xc1) | E(xa,xc,xa1,xc1).  % Satz 2.11
xq = xa | -T(xq,xa,u) | -E(xa,u,xc,xd) | ext(xq,xa,xc,xd) = u.    % Satz 2.12
T(x,y,y).    % Satz 3.1
-E(u,v,x,x) | u=v.   % Not one of Szmielew's theorems but we proved it.
-T(xa,xb,xc) | T(xc,xb,xa).  % Satz 3.2.
T(xa,xa,xb).  % Satz 3.3
-T(xa,xb,xc) | -T(xb,xa,xc) | xa = xb.  % Satz 3.4.
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xb,xc).  % Satz 3.51.
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xc,xd).  % Satz 3.52.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd).  % Satz 3.61.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd).  % Satz 3.61.
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd).  % Satz 3.71
```

```
alpha != beta.   %related to Satz 3.14;  easily provable if added to sst 3h.in.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd).  % Satz 3.62.
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd).  % Satz 3.71
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd).  % Satz 3.72
-IFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xd,xb1,xd1).  % Satz 4.2
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xc,xa1,xc1) | -E(xb,xc,xb1,xc1) | E(xa,xb,xa1,xb1).  % Satz 4.3

alpha != beta.  % Satz 3.13
beta != gamma.
alpha != gamma.
T(xa,xb,ext(xa,xb,alpha,gamma)).   % Satz 3.14, first half
xb != ext(xa,xb,alpha,gamma).    % Satz 3.14, second half
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | T(xa,xb,xc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | T(za,zb,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xa,xc,za,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xb,xc,zb,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xa,xd,za,zd).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xc,xd,zc,zd).
-T(xa,xb,xc) | -T(za,zb,zc) | -E(xa,xc,za,zc) | -E(xb,xc,zb,zc) | -E(xa,xd,za,zd) |
 -E(xc,xd,zc,zd) | IFS(xa,xb,xc,xd,za,zb,zc,zd).

% Following 4 are definition 4.4 for n=3
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa1,xa2,xb1,xb2).
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa1,xa3,xb1,xb3).
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa2,xa3,xb2,xb3).
-E(xa1,xa2,xb1,xb2) | -E(xa1,xa3,xb1,xb3) | -E(xa2,xa3,xb2,xb3) | E3(xa1,xa2,xa3,xb1,xb2,xb3).
% Following three lines are Satz 4.5
-T(xa,xb,xc) | -E(xa,xc,xa1,xc1) | T(xa1,insert(xa,xb,xa1,xc1),xc1).
-T(xa,xb,xc) | -E(xa,xc,xa1,xc1) | E3(xa,xb,xc,xa1,insert(xa,xb,xa1,xc1),xc1).
insert(xa,xb,xa1,xc1) = ext(ext(xc1,xa1,alpha,gamma),xa1,xa,xb).
-E3(x,y,z,u,v,w) | E3(x,z,y,u,w,v).   % See sst4q.in, not in Szmielew
-T(xa,xb,xc) | -E3(xa,xb,xc,xa1,xb1,xc1) | T(xa1,xb1,xc1).  % Satz 4.6

% following is Definition 4.10
-Col(xa,xb,xc) | T(xa,xb,xc) | T(xb,xc,xa) | T(xc,xa,xb).
Col(xa,xb,xc) | -T(xa,xb,xc).
Col(xa,xb,xc) | -T(xb,xc,xa).
Col(xa,xb,xc) | -T(xc,xa,xb).
% Following are Satz 4.11
-Col(x,y,z) | Col(y,z,x).
-Col(x,y,z) | Col(z,x,y).
-Col(x,y,z) | Col(z,y,x).
-Col(x,y,z) | Col(y,x,z).
-Col(x,y,z) | Col(x,z,y).
% following is Satz 4.12
Col(x,x,y).
% following is Satz 4.13
-Col(xa,xb,xc) | - E3(xa,xb,xc,xa1,xb1,xc1) | Col(xa1,xb1,xc1).
% following is Satz 4.14
-Col(xa,xb,xc) | -E(xa,xb,xa1,xb1) | E3(xa,xb,xc,xa1,xb1,insert5(xa,xb,xc,xa1,xb1)).
% following is Definition 4.15
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | Col(xa,xb,xc).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E3(xa,xb,xc,xa1,xb1,xc1).
```

```
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xa,xd,xa1,xd1).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xd,xb1,xd1).
-Col(xa,xb,xc) | - E3(xa,xb,xc,xa1,xb1,xc1) | - E(xa,xd,xa1,xd1) |
 -E(xb,xd,xb1,xd1) | FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1).
% Following is Satz 4.16
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | xa = xb | E(xc,xd,xc1,xd1).
% Following is Satz 4.17
xa = xb | -Col(xa,xb,xc) | -E(xa,xp,xa,xq) | -E(xb,xp,xb,xq) | E(xc,xp,xc,xq).
% Following is Satz 4.18
xa = xb | -Col(xa,xb,xc) | -E(xa,xc,xa,xc1) | -E(xb,xc,xb,xc1) | xc = xc1.
% Following is Satz 4.19
-T(xa,xc,xb) | -E(xa,xc,xa,xc1) | -E(xb,xc,xb,xc1) | xc = xc1.
% Following defines T5
T5(x,y,z,u,v) | -T(x,y,z) | -T(x,y,u) | -T(x,y,v) | -T(x,z,u) | -T(x,z,v) | -T(x,u,v).
-T5(x,y,z,u,v) | T(x,y,z).
-T5(x,y,z,u,v) | T(x,y,u).
-T5(x,y,z,u,v) | T(x,y,v).
-T5(x,y,z,u,v) | T(x,z,u).
-T5(x,y,z,u,v) | T(x,z,v).
-T5(x,y,z,u,v) | T(x,u,v).
% Following defines AFS
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xb,xa1,xb1) | -E(xb,xc,xb1,xc1) |
 -E(xa,xd,xa1,xd1) | -E(xb,xd,xb1,xd1) | AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | T(xa,xb,xc).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | T(xa1,xb1,xc1).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xa,xb,xa1,xb1).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xc,xb1,xc1).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xd,xb1,xd1).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1).
% Following proved in sst5a.in
% following proved in sst5a2.in
% Following proved in sst5a3.in
end_of_list.

list(passive).
-AFS(b,c,d1,c1,b1,c1,d,c) |  $ANS(TARG1INT).
-E(c1,d1,c,d) |  $ANS(TARG2INT).
-IFS(d,e,d1,c,d,e,d1,c1) |  $ANS(TARG3INT).
-IFS(c,e,c1,d,c,e,c1,d1) |  $ANS(TARG4INT).
-E(e,c,e,c1) |  $ANS(TARG5INT).
%  %  following negs from 66 proof steps.
%  %  following 66 shorter, temp.beeson.sst5a4.of4t25
%  %  %  Following negs from a 79-step proof of what might be the primary target.
%  %  following 10 are later steps in proofs of 38 and F, temp.beeson.sst5a4.out26
end_of_list.

list(sos).
    %prove the theorem in case c = c1, which should be easy.
a != b.
T(a,b,c).
T(a,b,d).
-T(a,c,d) | $ANS(CON1).
-T(a,d,c) | $ANS(CON2).
```

```
c1 = ext(a,d,c,d).
d1 = ext(a,c,c,d).
b1 = ext(a,c1,c,b).
b2 = ext(a,d1,d,b).
e = ip(c1,d,b,d1,c).
p = ext(e,c,c,d).
r = ext(d1,c,e,c).
q = ext(p,r,p,r).
c != b.
end_of_list.

list(hints).
% Following 104 purport to prove trans of betw depend on axms with inner Pasch.
-E(x,y,z,u)|E(z,u,y,x).
E(ext(x,y,z,u),y,z,u).
E(x,y,ext(z,u,x,y),u).
E(x,y,z,u)| -E(z,u,y,x).
E(x,y,z,u)| -E(y,x,u,z).
-E(x,x,y,z)|E(u,u,y,z).
-T(x,y,z)| -T(y,x,u)| -E(y,z,x,u)|E(x,z,y,u).
T(ext(x,y,z,u),y,x).
-T(x,y,z)| -T(y,u,z)| -T(u,x,z)|u=x.
-E(x,y,z,u)| -E(z,y,x,u)|E3(x,z,y,z,x,u).
T(b,c,d1).
-T(c1,x,d1)| -T(c,y,d)| -E(x,d1,y,d)|E(c1,x,c,y).
E(d,c,d,c1).
T(d,e,d1).
E(c1,d,c1,d1).
E(c,d,c,d1).
T(c,e,c1).
E(c,e,e,c1).
$ANS(CON)| -T(b,d,c).
T(a,d,c1).
E(c,d1,c,d).
T(a,c,d1).
E(c,p,c,d).
T(e,c,p).
E(c,r,e,c).
T(d1,c,r).
E(r,q,p,r).
T(p,r,q).
E(ext(x,y,z,u),y,u,z).
E(d1,c,c,d).
E(e,c,r,c).
E(c,d,p,c).
T(p,c,e).
Col(c,d1,b).
E(c,d,d,c1).
E(c1,d1,c1,d).
E(d,c,c,d1).
E3(c,d1,d,c,d,d1).
Col(c,d1,a).
-T(a,d1,d)|$ANS(CON2).
```

-T(c,d,d1)|$ANS(CON).
E(c,p,c,d1).
E(c,r,c,e).
E(c,r,e,c1).
Col(c,r,d1).
E(r,q,r,p).
E(r,c,c,e).
E(d1,c,p,c).
E(r,p,r,q).
-T(c,d1,d)|$ANS(CON).
d1=c|E(r,p,e,d1).
T(d1,e,d).
T(c1,e,c).
Col(c,c1,e).
e=c|T(c1,e,p).
e=c|T(c1,c,p).
c=c1|E(e,d1,e,d).
T(a,b,c1).
T(b,d,c1).
-T(d,c,c1)|$ANS(CON).
Col(a,b,c1).
$ANS(CON)| -T(d1,d,d1).
$ANS(CON)| -E3(x,y,y,c,d,d1).
$ANS(CON)|E(r,p,e,d1).
$ANS(CON)|E(p,r,d1,e).
$ANS(CON)|E(r,q,e,d1).
$ANS(CON)|E(d1,e,p,r).
$ANS(CON)|E(e,d1,e,d).
$ANS(CON)|E(r,q,e,d).
$ANS(CON)|E(e,d,r,q).
$ANS(CON)|E(p,q,d1,d).
$ANS(CON)|d1=e|E(d,c,q,c).
$ANS(CON)|E(p,q,d,d1).
$ANS(CON)| -E3(x,y,y,d,c,c1).
$ANS(CON)| -T(e,d,d1)|E(d,c,q,c).
T(c1,d1,c)|$ANS(CON)|E(d,c,q,c).
e=c| -T(x,c1,p)| -T(e,x,p)|e=x.
e=c| -T(c,c1,p).
e=c|Col(c1,c,p).
-E(c,c,d,d1)|$ANS(CON).
-E(d,d,c,c1)|$ANS(CON).
$ANS(CON)|E(d,c,q,c)|d1=d.
$ANS(CON)|E(d,c,q,c).
$ANS(CON)|E(c,d,c,q).
$ANS(CON)|E(c,p,c,q).
$ANS(CON)|c=r|E(d1,p,d1,q).
e=c| -T(p,c1,c).
$ANS(CON)| -E(x,x,d,d1).
$ANS(CON)| -E(x,x,c,c1).
$ANS(CON)| -E(x,x,p,q).
$ANS(CON)| -E(c1,c,x,x).
$ANS(CON)| -T(e,c,e).
$ANS(CON)| -T(r,c,r).

-T(c1,c,d1)|$ANS(CON).
$ANS(CON)| -T(p,c1,c).
$ANS(CON)|Col(c1,c,p).
$ANS(CON)|E(d1,p,d1,q).
$ANS(CON)|c=d1|E(a,p,a,q).
$ANS(CON)|c=d1|E(b,p,b,q).
$ANS(CON)|E(a,p,a,q).
$ANS(CON)|E(b,p,b,q).
$ANS(CON)|E(c1,p,c1,q).
$ANS(CON)|c1=c.
$ANS(CON)| -T(p,c,c).
% Following 87 sorted from temp.beeson.sst5a4.out40t1k1 -k4,
of five lemmas Beeson wishes to avoid.
$ANS(TARG1INT)| -E(b2,c,b1,c).
$ANS(TARG1INT)| -E(b,b2,b,b1).
$ANS(TARG4INT)| -T(c,e,c1).
$ANS(TARG4INT)| -T(c,ip(c1,d,b,d1,c),c1).
$ANS(TARGINT)| -E(c,d,c1,d1).
$ANS(TARGINT)| -IFS(x,c,y,d,z,c1,u,d1).
b1=b2.
Col(a,b,b1).
Col(e,d1,d).
E(a,b1,a,b2).
E(b1,c1,b,c).
E(b1,c1,c,b).
E(b1,d,b,d1).
E(b2,d1,d,b).
E(b2,d,b,d1).
E(b,b1,b2,b).
E(b,b1,b,b2).
E(b,b2,b1,b).
E(b,b2,b,b1).
-E(b,c1,b1,c)|$ANS(TARG1INT).
E(b,c1,b2,c).
E(b,c,b1,c1).
E(b,d,b2,d1).
E(c1,b1,c,b).
E(c1,d1,c1,d).
E(c1,d1,d1,c).
E(c1,d,c1,d1).
E(c1,d,c1,ext(a,c,c,d)).
E(c1,d,c,d1).
E(c,b2,c1,b).
E(c,d1,c1,d).
E(c,d1,c,d).
E(c,d,c1,d).
E(c,d,c,d1).
E(c,d,c,ext(a,c,c,d)).
E(c,d,d1,c).
e=d1|$ANS(TARG3INT).
E(d1,b2,d,b).
E(d1,c,d1,c1).
E(d1,c,e,c1)|$ANS(TARG3INT).

```
-E(d1,c,e,c1)|$ANS(TARG5INT)|$ANS(TARG3INT).
E(d,c1,c,d).
E(d,c1,d1,c).
E(d,c1,d,c).
E(d,c,d,c1).
-E(d,d1,d,d1)|$ANS(TARG3INT).
E(ext(x,y,z,u),y,z,u).
e=ip(ext(a,d,c,d),d,b,d1,c).
E(x,y,ext(z,u,x,y),u).
E(x,y,z,u)| -E(y,x,u,z).
E(x,y,z,u)| -E(z,u,y,x).
IFS(b,c,b2,d,b2,c1,b,d1).
-IFS(c,e,c1,d,c,e,c1,ext(a,c,c,d))|$ANS(TARG4INT).
IFS(d,e,d1,c,d,e,d1,c1).
T(a,b,b1).
T(a,b,b2).
T(a,b,c1).
T(a,c1,b1).
T(a,c,b2).
T(a,c,d1).
T(a,d1,b2).
T(a,d,c1).
T(b1,c1,a).
T(b1,c1,b).
T(b1,c1,d).
T(b2,c1,b).
T(b2,c,a).
T(b2,c,b).
T(b2,d1,a).
T(b2,d1,c).
T(b,c1,b1).
T(b,c,b2).
T(b,c,d1).
T(b,d,c1).
T(b,d,ext(a,d,x,y)).
T(c1,b,a).
T(c1,d,a).
T(c1,d,b).
T(c,b,a).
T(c,d1,b2).
T(c,ip(c1,d,b,d1,c),c1).
T(d1,c,a).
T(d1,c,b).
T(d,b,a).
T(d,e,d1).
T(ext(a,d,x,y),d,b).
-T(x,y,z)| -T(x,y,u)| -E(y,z,y,u)|E(x,z,x,u).
end_of_list.
```

If a proof would be found with this file, the prediction was that the proof would be longer, perhaps much longer, than the earlier-obtained 104-step proof. For you to see immediately why such is the case, by way of a totally trivial example, the inclusion of steps near the end of a proof in hand in an input file should, and usually does, lead to a quite short proof. So, in a sense conversely, the removal of items from an input file, such as the removal of Lemmas RG1 through RG5, should, if all goes well, lead to finding a longer proof than that found in their presence. Now, with the input file just given, you might discover that one or more of the five lemmas would be deduced and used in a sought-after proof. Although none of the runs that proved one or more of the five lemmas (even after much CPU time was used) led to a proof of connectivity, as so often occurred, use of the corresponding proofs might just get my goal. And indeed OTTER did find the following longer proof, a proof that I will examine in some detail.

A Proof Using Inner Pasch of Length 154 of Connectivity of Betweenness

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Sun Feb  3 09:24:12 2013
The command was "otter".  The process ID is 14677.
----> UNIT CONFLICT at 1267.28 sec ----> 187660 [binary,187659.1,185347.1] $ANS(CON2)|$ANS(CON1).

Length of proof is 154.  Level of proof is 27.

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

1 [] E(x,y,y,x).
2 [] -E(x,y,z,v)| -E(x,y,z2,v2)|E(z,v,z2,v2).
4 [] T(x,y,ext(x,y,w,v)).
5 [] E(y,ext(x,y,w,v),w,v).
6 [] -E(x,y,x1,y1)| -E(y,z,y1,z1)| -E(x,v,x1,v1)| -E(y,v,y1,v1)| -T(x,y,z)| -T(x1,y1,z1)|x=y|E(z,v,z1,v1).
8 [] -T(xa,xp,xc)| -T(xb,xq,xc)|T(xp,ip(xa,xp,xc,xb,xq),xb).
9 [] -T(xa,xp,xc)| -T(xb,xq,xc)|T(xq,ip(xa,xp,xc,xb,xq),xa).
13 [] E(x,y,x,y).
14 [] -E(xa,xb,xc,xd)|E(xc,xd,xa,xb).
15 [] -E(xa,xb,xc,xd)|E(xb,xa,xc,xd).
16 [] -E(xa,xb,xc,xd)| -E(xc,xd,xe,xf)|E(xa,xb,xe,xf).
17 [] -E(xa,xb,xc,xd)|E(xa,xb,xd,xc).
18 [] E(x,x,y,y).
19 [] -T(xa,xb,xc)| -T(xa1,xb1,xc1)| -E(xa,xb,xa1,xb1)| -E(xb,xc,xb1,xc1)|E(xa,xc,xa1,xc1).
21 [] T(x,y,y).
23 [] -T(xa,xb,xc)|T(xc,xb,xa).
24 [] T(xa,xa,xb).
26 [] -T(xa,xb,xd)| -T(xb,xc,xd)|T(xa,xb,xc).
27 [] -T(xa,xb,xd)| -T(xb,xc,xd)|T(xa,xc,xd).
28 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xb,xc,xd).
32 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xa,xb,xd).
33 [] -T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xc,xd).
35 [] -IFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1)|E(xb,xd,xb1,xd1).
36 [] -T(xa,xb,xc)| -T(xa1,xb1,xc1)| -E(xa,xc,xa1,xc1)| -E(xb,xc,xb1,xc1)|E(xa,xb,xa1,xb1).
48 [] -T(xa,xb,xc)| -T(za,zb,zc)| -E(xa,xc,za,zc)| -E(xb,xc,zb,zc)|
 -E(xa,xd,za,zd)| -E(xc,xd,zc,zd)|IFS(xa,xb,xc,xd,za,zb,zc,zd).
52 [] -E(xa1,xa2,xb1,xb2)| -E(xa1,xa3,xb1,xb3)| -E(xa2,xa3,xb2,xb3)|E3(xa1,xa2,xa3,xb1,xb2,xb3).
57 [] -T(xa,xb,xc)| -E3(xa,xb,xc,xa1,xb1,xc1)|T(xa1,xb1,xc1).
59 [] Col(xa,xb,xc)| -T(xa,xb,xc).
60 [] Col(xa,xb,xc)| -T(xb,xc,xa).
61 [] Col(xa,xb,xc)| -T(xc,xa,xb).
76 [] xa=xb| -Col(xa,xb,xc)| -E(xa,xp,xa,xq)| -E(xb,xp,xb,xq)|E(xc,xp,xc,xq).
77 [] xa=xb| -Col(xa,xb,xc)| -E(xa,xc,xa,xc1)| -E(xb,xc,xb,xc1)|xc=xc1.
98 [] a!=b.
```

99 [] T(a,b,c).
100 [] T(a,b,d).
101 [] -T(a,c,d)|$ANS(CON1).
102 [] -T(a,d,c)|$ANS(CON2).
103 [] c1=ext(a,d,c,d).
104 [] d1=ext(a,c,c,d).
105 [] b1=ext(a,c1,c,b).
106 [] b2=ext(a,d1,d,b).
107 [] e=ip(c1,d,b,d1,c).
108 [] p=ext(e,c,c,d).
109 [] r=ext(d1,c,e,c).
110 [] q=ext(p,r,p,r).
304 [binary,2.2,1.1] -E(x,y,z,u)|E(z,u,y,x).
307 [hyper,2,1,5] E(ext(x,y,z,u),y,z,u).
308 [hyper,2,5,1] E(x,y,ext(z,u,x,y),u).
309 [hyper,2,5,13] E(x,y,z,ext(u,z,x,y)).
311 [binary,15.1,14.2] E(x,y,z,u)| -E(z,u,y,x).
313 [hyper,16,5,1] E(x,ext(y,x,z,u),u,z).
314 [binary,17.1,15.2] E(x,y,z,u)| -E(y,x,u,z).
316 [binary,18.1,16.1] -E(x,x,y,z)|E(u,u,y,z).
319 [binary,19.3,13.1] -T(x,y,z)| -T(x,y,u)| -E(y,z,y,u)|E(x,z,x,u).
320 [binary,19.3,1.1] -T(x,y,z)| -T(y,x,u)| -E(y,z,x,u)|E(x,z,y,u).
326 [binary,23.1,4.1] T(ext(x,y,z,u),y,x).
425 [binary,52.1,1.1] -E(x,y,z,u)| -E(z,y,x,u)|E3(x,z,y,z,x,u).
554 [binary,99.1,23.1] T(c,b,a).
616 [binary,100.1,23.1] T(d,b,a).
640 [hyper,28,100,4] T(b,d,ext(a,d,x,y)).
727 [para_from,103.1.2,5.1.2] E(d,c1,c,d).
728 [para_from,103.1.2,4.1.3] T(a,d,c1).
729 [para_from,104.1.2,5.1.2] E(c,d1,c,d).
730 [para_from,104.1.2,4.1.3] T(a,c,d1).
732 [para_from,105.1.2,5.1.2] E(c1,b1,c,b).
733 [para_from,105.1.2,4.1.3] T(a,c1,b1).
735 [para_from,106.1.2,5.1.2] E(d1,b2,d,b).
736 [para_from,106.1.2,4.1.3] T(a,d1,b2).
737 [para_into,107.1.2.1,103.1.1] e=ip(ext(a,d,c,d),d,b,d1,c).
738 [para_from,108.1.2,5.1.2] E(c,p,c,d).
739 [para_from,108.1.2,4.1.3] T(e,c,p).
741 [para_from,109.1.2,5.1.2] E(c,r,e,c).
742 [para_from,109.1.2,4.1.3] T(d1,c,r).
747 [para_from,110.1.2,5.1.2] E(r,q,p,r).
748 [para_from,110.1.2,4.1.3] T(p,r,q).
770 [para_into,307.1.1,106.1.2] E(b2,d1,d,b).
772 [para_into,307.1.1,104.1.2] E(d1,c,c,d).
786 [para_into,308.1.3,109.1.2] E(e,c,r,c).
787 [para_into,308.1.3,108.1.2] E(c,d,p,c).
790 [para_into,308.1.3,104.1.2] E(c,d,d1,c).
791 [para_into,308.1.3,103.1.2] E(c,d,c1,d).
832 [para_into,309.1.4,104.1.2] E(c,d,c,d1).
833 [para_into,309.1.4,103.1.2] E(c,d,d,c1).
889 [para_into,326.1.1,108.1.2] T(p,c,e).
890 [para_into,326.1.1,106.1.2] T(b2,d1,a).
891 [para_into,326.1.1,105.1.2] T(b1,c1,a).

892 [para_into,326.1.1,104.1.2] T(d1,c,a).
893 [para_into,326.1.1,103.1.2] T(c1,d,a).
1046 [hyper,26,326,616] T(ext(a,d,x,y),d,b).
1092 [para_into,640.1.3,103.1.2] T(b,d,c1).
1137 [binary,727.1,311.2] E(d,c,d,c1).
1214 [hyper,32,100,728] T(a,b,c1).
1245 [ur,26,728,102] -T(d,c,c1)|$ANS(CON2).
1248 [binary,729.1,311.2] E(d,c,c,d1).
1280 [binary,730.1,61.2] Col(c,d1,a).
1327 [hyper,28,99,730] T(b,c,d1).
1356 [ur,26,730,101] -T(c,d,d1)|$ANS(CON1).
1388 [para_into,313.1.2,110.1.2] E(r,q,r,p).
1389 [para_into,313.1.2,109.1.2] E(c,r,c,e).
1394 [binary,732.1,314.2] E(b1,c1,b,c).
1592 [hyper,32,730,736] T(a,c,b2).
1595 [hyper,28,730,736] T(c,d1,b2).
1625 [para_from,737.1.2,8.3.2,unit_del,1046] -T(d1,c,b)|T(d,e,d1).
1738 [binary,741.1,314.2] E(r,c,c,e).
1775 [binary,742.1,61.2] Col(c,r,d1).
1853 [binary,747.1,311.2] E(r,p,r,q).
2035 [binary,770.1,311.2] E(b,d,b2,d1).
2260 [hyper,16,772,787] E(d1,c,p,c).
2295 [hyper,16,727,790] E(d,c1,d1,c).
2328 [hyper,16,729,791] E(c,d1,c1,d).
2358 [hyper,52,729,832,1] E3(c,d1,d,c,d,d1).
2374 [hyper,16,738,832] E(c,p,c,d1).
2379 [hyper,2,791,832] E(c1,d,c,d1).
2978 [hyper,27,890,892] T(b2,c,a).
2983 [hyper,26,890,892] T(b2,d1,c).
2985 [hyper,26,892,554] T(d1,c,b).
3097 [hyper,27,893,616] T(c1,b,a).
3100 [hyper,26,891,893] T(b1,c1,d).
3102 [hyper,26,893,616] T(c1,d,b).
3412 [binary,1214.1,59.2] Col(a,b,c1).
3468 [hyper,32,1214,733] T(a,b,b1).
3473 [hyper,28,1214,733] T(b,c1,b1).
3567 [ur,57,21,1245] $ANS(CON2)| -E3(x,y,y,d,c,c1).
3679 [binary,1327.1,61.2] Col(c,d1,b).
4004 [ur,57,21,1356] $ANS(CON1)| -E3(x,y,y,c,d,d1).
4334 [hyper,32,99,1592] T(a,b,b2).
4341 [hyper,28,99,1592] T(b,c,b2).
5016 [binary,2358.1,57.2,unit_del,1356] -T(c,d1,d)|$ANS(CON1).
5068 [hyper,6,2260,1389,1,2374,742,889] d1=c|E(r,p,e,d1).
5314 [hyper,26,2978,554] T(b2,c,b).
5476 [hyper,19,1092,2983,2035,2295] E(b,c1,b2,c).
5876 [hyper,26,891,3097] T(b1,c1,b).
6051 [hyper,19,3100,1327,1394,2379] E(b1,d,b,d1).
6185 [hyper,19,1595,3102,2328,735] E(c,b2,c1,b).
6208 [hyper,9,3102,2985] T(c,ip(c1,d,b,d1,c),c1).
6284 [binary,3468.1,59.2] Col(a,b,b1).
6566 [binary,3567.1,425.3,unit_del,833] $ANS(CON2)| -E(d,d,c,c1).
6648 [binary,4004.1,425.3,unit_del,1248] $ANS(CON1)| -E(c,c,d,d1).
7053 [para_from,5068.1.1,5016.1.2,unit_del,24] $ANS(CON1)|E(r,p,e,d1).

7236 [hyper,19,3473,5314,5476,732] E(b,b1,b2,b).
7562 [para_into,6208.1.2,107.1.2] T(c,e,c1).
7568 [binary,6566.1,316.2] $ANS(CON2)| -E(x,x,c,c1).
7587 [binary,6648.1,316.2] $ANS(CON1)| -E(x,x,d,d1).
7606 [binary,7053.1,314.2] $ANS(CON1)|E(p,r,d1,e).
7638 [hyper,16,1388,7053] $ANS(CON1)|E(r,q,e,d1).
7645 [hyper,2,7053,1853] $ANS(CON1)|E(e,d1,r,q).
7660 [binary,7236.1,17.1] E(b,b1,b,b2).
7790 [binary,7562.1,23.1] T(c1,e,c).
7913 [binary,7568.1,314.1] $ANS(CON2)| -E(x,x,c1,c).
7915 [binary,7568.1,304.2] $ANS(CON2)| -E(c1,c,x,x).
7963 [binary,7587.1,311.1] $ANS(CON1)| -E(d,d1,x,x).
7997 [binary,7606.1,14.1] $ANS(CON1)|E(d1,e,p,r).
8149 [hyper,319,3468,4334,7660] E(a,b1,a,b2).
8201 [binary,7790.1,60.2] Col(c,c1,e).
8249 [hyper,33,7790,739] e=c|T(c1,c,p).
8436 [hyper,77,6284,8149,7660,unit_del,98] b1=b2.
8648 [binary,8249.2,59.2] e=c|Col(c1,c,p).
8745 [para_from,8436.1.1,6051.1.1] E(b2,d,b,d1).
8746 [para_from,8436.1.1,5876.1.1] T(b2,c1,b).
9103 [hyper,48,4341,8746,1,6185,2035,8745] IFS(b,c,b2,d,b2,c1,b,d1).
9345 [binary,9103.1,35.1] E(c,d,c1,d1).
66751 [hyper,2,9345,791] E(c1,d1,c1,d).
66752 [hyper,2,9345,790] E(c1,d1,d1,c).
66757 [ur,36,24,9345,832,7913] -T(c1,c,d1)|$ANS(CON2).
66859 [hyper,76,8201,729,66751] c=c1|E(e,d1,e,d).
66914 [binary,66752.1,304.1] E(d1,c,d1,c1).
67000 [binary,66757.1,23.2] $ANS(CON2)| -T(d1,c,c1).
67148 [para_from,66859.1.1,66757.1.2,unit_del,24] $ANS(CON2)|E(e,d1,e,d).
67237 [binary,67148.1,314.2] $ANS(CON2)|E(d1,e,d,e).
67321 [hyper,2,67148,7645] $ANS(CON2)|E(e,d,r,q)|$ANS(CON1).
99483 [hyper,16,7606,67237] $ANS(CON2)|E(p,r,d,e)|$ANS(CON1).
99495 [ur,36,24,13,67237,7963] $ANS(CON2)| -T(d,d1,e)|$ANS(CON1).
99519 [binary,99495.1,23.2] $ANS(CON2)|$ANS(CON1)| -T(e,d1,d).
176170 [binary,1625.1,2985.1] T(d,e,d1).
176196 [binary,1625.2,23.1,unit_del,2985] T(d1,e,d).
176239 [hyper,48,176170,176170,13,13,1137,66914] IFS(d,e,d1,c,d,e,d1,c1).
176317 [hyper,19,748,176170,99483,7638,factor_simp] E(p,q,d,d1)|$ANS(CON2)|$ANS(CON1).
177084 [hyper,6,7997,67321,2260,786,176196,748,factor_simp] d1=e|E(d,c,q,c)|$ANS(CON1)|$ANS(CON2).
177441 [binary,176239.1,35.1] E(e,c,e,c1).
177560 [ur,16,176317,7587,factor_simp] $ANS(CON2)|$ANS(CON1)| -E(x,x,p,q).
177708 [para_from,177084.1.1,99519.3.2,unit_del,24,factor_simp,factor_simp]
$ANS(CON2)|$ANS(CON1)|E(d,c,q,c).
177804 [binary,177708.1,314.2] $ANS(CON2)|$ANS(CON1)|E(c,d,c,q).
177994 [hyper,16,738,177804] $ANS(CON2)|$ANS(CON1)|E(c,p,c,q).
178086 [hyper,76,1775,177994,1853] $ANS(CON2)|$ANS(CON1)|c=r|E(d1,p,d1,q).
183682 [binary,177441.1,15.1] E(c,e,e,c1).
183798 [hyper,16,741,177441] E(c,r,e,c1).
183841 [binary,183682.1,320.3,unit_del,7562,7568] -T(e,c,e)|$ANS(CON2).
183961 [ur,19,7562,1738,183798,7568] -T(r,c,r)|$ANS(CON2).
184125 [para_into,183841.1.1,8648.1.1,unit_del,24] $ANS(CON2)|Col(c1,c,p).
184128 [para_into,183841.1.1,8249.1.1,unit_del,24] $ANS(CON2)|T(c1,c,p).
184282 [para_into,183961.1.1,178086.1.2,unit_del,24,factor_simp]

```
$ANS(CON2)|$ANS(CON1)|E(d1,p,d1,q).
185347 [ur,36,184128,24,177994,7915,factor_simp,factor_simp]
 $ANS(CON2)| -E(c1,p,c,q)|$ANS(CON1).
186100 [hyper,76,3679,177994,184282,factor_simp,factor_simp]
 $ANS(CON2)|$ANS(CON1)|c=d1|E(b,p,b,q).
186101 [hyper,76,1280,177994,184282,factor_simp,factor_simp]
 $ANS(CON2)|$ANS(CON1)|c=d1|E(a,p,a,q).
186449 [para_from,186100.1.1,67000.2.2,unit_del,24,factor_simp]
 $ANS(CON2)|$ANS(CON1)|E(b,p,b,q).
187134 [para_from,186101.1.1,67000.2.2,unit_del,24,factor_simp]
 $ANS(CON2)|$ANS(CON1)|E(a,p,a,q).
187326 [hyper,76,3412,187134,186449,unit_del,98,factor_simp,factor_simp]
 $ANS(CON2)|$ANS(CON1)|E(c1,p,c1,q).
187515 [hyper,76,184125,187326,177994,unit_del,177560,factor_simp,factor_simp,
 factor_simp,factor_simp,factor_simp] $ANS(CON2)|$ANS(CON1)|c1=c.
187659 [para_from,187515.1.1,187326.3.3,factor_simp,factor_simp]
 $ANS(CON2)|$ANS(CON1)|E(c1,p,c,q).
187660 [binary,187659.1,185347.1] $ANS(CON2)|$ANS(CON1).
```

Before completing the cited bidirectional 154-step proof, the program found, in order, forward proofs of RG1, 2, 4, 3, and 5, of respective lengths 45, 60, 65, 69, and 70. Although none of the five was included in the input, and each was proved, the third and fifth actually are found in the 154-step proof. Somewhat piquant, when I blocked the use of binary resolution and made no other changes, OTTER proved only the first cited three of the lemmas. Also, in the presence of binary resolution but with demodulators to block the retention when deduced of the third and fifth lemmas, the proof of connectivity is still not forthcoming. As in the 104-step proof, transitivity of betweenness is employed in the 154-step proof. In contrast to the 104-step proof that relies on inner Pasch implicitly, in the 154-step proof, both clauses for inner Pasch are relied upon explicitly. For clarity, by relying on 3.51 and 3.61, implicit use of inner Pasch is seen. In other words, the 154-step proof relies explicitly on two additional clauses from among the Tarski axiom system that includes inner Pasch, when compared with the 104-step proof.

For further comparison, among the theorems previously proved, the 154-step proof relies on the use of 2.1, 4.2, part of the definition of 4.1, and 4.18. Whereas the 104-step proof relies on the use of five defined points (by Beeson), the 154-step proof relies on eight defined points. Most important to note, other than Beeson's crucial suggestions and guidance and McCune's incredible automated reasoning program OTTER, is the fact that neither the 104-step nor the 154-step proof would have been found without the direction provided to Beeson and me by the Szmielew book and its approach to proving connectivity. That theorem, had we not followed Szmielew's approach, would have been clearly out of reach for OTTER and for us. To amplify just a bit, when in 2002 the formula *XCB* of equivalential calculus was established to be a single axiom for this area of logic, no proof existed; nothing was extant that could be followed or emulated.

At this point in the experiments I was conducting, my curiosity arose in the context of how much connection exists between transitivity and connectivity of betweenness. In other words, if transitivity is removed from the input file that yielded the 154-step proof, will OTTER still find a proof of the dependence of connectivity? Of course, as you and I know, to tidy things up, proofs of all the theorems used in such a proof would be nice to have in hand, proofs in which transitivity is not used in the input, which is possible since transitivity has been proved dependent. (I do not intend to follow that thorough path at the moment but simply try for a proof based on the input file that was used to obtain the 154-step proof, with transitivity of course deleted. I note, however, that my study of the Tarski axioms given in Section 1, in which outer Pasch is present, resulted in my finding many proofs of Szmielew theorems with an input file in which both transitivity and connectivity of betweenness were absent.)

Again, success was the result. OTTER found a 146-step bidirectional proof in approximately 151 CPU-seconds and with retention of clause (118384). You have an interesting example of using fewer axioms and, yet, obtaining a shorter proof, one of length less than 154. Again, as in the 154-step proof, OTTER's proof relied on eight defined points, the following.

102 [] c1=ext(a,d,c,d).
103 [] d1=ext(a,c,c,d).
104 [] b1=ext(a,c1,c,b).
105 [] b2=ext(a,d1,d,b).
106 [] e=ip(c1,d,b,d1,c).
107 [] p=ext(e,c,c,d).
108 [] r=ext(d1,c,e,c).
109 [] q=ext(p,r,p,r).

Both clauses for inner Pasch were cited, as well as the two for extensionality. As for piquant differences, thirty-one deduced clauses are present in the 146-step proof that are not found in the 154-step proof.

With the finding of a proof of connectivity that does not rely on transitivity, not even at the deduced level, I naturally wondered whether I could block, at the same time, some other theorem that was used from among those relied upon among the input clauses. An appealing target was Szmielew's 4.3, a theorem I had not sought a proof of, but simply accepted it for use; Beeson I am quite sure had proved it.

-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xc,xa1,xc1) | -E(xb,xc,xb1,xc1) |
E(xa,xb,xa1,xb1). % Satz 4.3

My experiment with the goal of proving connectivity in the absence of transitivity and Satz 4.3 yielded no proof. After various other experiments, I chose as target (to avoid using) 4.19, the following.

-T(xa,xc,xb)| -E(xa,xc,xa,xc1)| -E(xb,xc,xb,xc1)|xc=xc1. Satz 4.19

And now you are introduced even more to the vagaries of research. Specifically, after checking, I found that 4.19 was not used in the 154-step proof, among the items that occur before deductions are given. However—and how fortunate that I was hasty and did not check before conducting the experiment—OTTER did find a proof of length 151 that avoids, at the so-called axiomatic level, transitivity and Satz 4.19. But, you are perhaps confused in that 4.19 was not used in the 154-step proof, but I simply took the input file that produced it and commented out 4.19, and, as I noticed later, 4.3 was still commented out from failed attempts. I have no explanation. Instead, the 151-step proof, with more than a chuckle from it, is a proof that avoids, before deduced clauses are presented, transitivity and 4.19 and, surprising to me (because of so many simultaneous experiments), finally avoids 4.3. Yes, a direct attempt at avoiding the use of 4.3, along with avoiding transitivity, in the input yielded nothing, and yet the added avoidance of 4.19 won the game. As expected, I leave to you the solving of the puzzle just described. By the way, among my experiments, I have obtained proofs that rely on forty-three items from among those offered in the input, and no fewer; the 104-step proof relies on forty-eight. Would Alama enjoy such successes, a reduction in the number of axioms and proved theorems needed to reach a desired target? More generally, would those individuals interested in proof shortening find so-called assumption reduction of interest?

With the cited successes in hand, those familiar with my research would accurately predict I would turn to proof shortening. Of course, I turned to the use of ancestor subsumption, the approach McCune devised, motivated by his intent to aid me in my interest in finding shorter proofs. When you instruct OTTER by including set(ancestor_subsume), and wisely include set(back_sub), the program automatically compares two proofs of the same conclusion and, if found, prefers the strictly shorter. I began with the input file that produced the 154-step proof, and I adjoined the two cited commands. Rather than detailing all the experiments, I will present the high points.

When the sequence of experiments ceased yielding shorter and still shorter proofs of connectivity, a sequence that was based on replacing one set of hints by that corresponding to a found shorter proof, my approach was that of relying on demodulation. To start, what you do is take an input file, place weight(junk,1000), for example, in weight_list(pick_and_purge), where 1000 is strictly greater than the value assigned to max_weight, and include a list(demodulators). The plan is to make a series of runs in which each run is based on the preceding with a single new demodulator placed in list(demodulators). The new demodulator is adjoined and chosen because it corresponds to blocking one of the deduced steps of the proof currently in hand and because that blocking enables the program to find a strictly shorter proof than that in hand. For example, assume that you have a proof **P** and that if you place the following demodulator, where *T* is a deduced step in **P**, in list(demodulators), the program will present you with a proof **Q** with the length of **Q** strictly less than the length of **P**.

EQ(T,junk).

As part of the plan, you iterate, focusing on your sequence of runs until no strictly shorter proof results. McCune wrote for me a program, otter-loop, that has me produce a file of demodulators (almost always corresponding to the deduced steps of the proof in hand) to try, an input file, a file to accept a summary of results after blocking the proof steps one at a time, and a message file. The command is the following.

otter-loop filename1 filename2 demodulators filename3, filename4

The filename1 is the input file, 2 is that containing the demodulators to consider, 3 is the file that contains the results of demodulator blocking, and 4 is for the messages (about errors, for example, which can be made). McCune produced various programs that I still use, programs that materially aid my research. Yes, he was a splendid colleague, and brilliant!

You might naturally wish to have one or more clues that explain why this demodulator-blocking can lead to shorter proofs. The following example template will serve nicely. Let **P** be a 40-step proof such that its twentieth step and its tenth together are the parents of the thirtieth. And, to avoid distraction, let us assume that the twentieth is not used elsewhere. It can happen that when the twentieth is blocked (by demodulating it, when deduced, to junk and discarded because of junk being assigned the value 1000), the twenty-second considered with the twelfth also provides the needed parents for the deduction of the thirtieth. Thus, with the removal of the twentieth from consideration, the program can produce a proof **Q** of length 39, which, ignoring parentage, is a subproof of the 40-step proof. Yes, I have often found a shorter proof all of whose steps were among those of a longer proof, but the parentage was different. Much satisfaction, and even amusement, can result from eventually finding a 34-step proof all of whose steps are among those of a 40-step proof that was used to initiate the search for a short proof.

A move that is often profitable has you, once the iterative process focusing on demodulation blocking yields no additional progress, produce another input file based on the file that was used in the beginning. The new file is obtained from the older by now removing all the adjoined demodulators and replacing the hints, or resonators, with the deduced steps of the latest and shortest proof that has been found. You can then begin anew, blocking the steps of this latest proof one at a time. McCune, in his genius, provided me with a variant of otter-loop, a program that enabled me to block proof steps two at a time, three at a time, and more, if I thought it profitable. Sometimes it was profitable.

I did apply the approach just discussed in the preceding paragraphs, beginning with the 154-step proof and its input file. I eventually was presented, by OTTER, with a proof of length 122, the following.

A Shorter Proof of the Dependency of Connectivity

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Fri Mar 22 07:23:22 2013
The command was "otter". The process ID is 31397.
----> UNIT CONFLICT at   3.10 sec ----> 6410 [binary,6409.1,3371.1] $ANS(CON2)|$ANS(CON1).

Length of proof is 122. Level of proof is 27.

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

1 [] E(x,y,y,x).
2 [] -E(x,y,z,v)| -E(x,y,z2,v2)|E(z,v,z2,v2).
4 [] T(x,y,ext(x,y,w,v)).
```

5 [] E(y,ext(x,y,w,v),w,v).
6 [] -E(x,y,x1,y1)| -E(y,z,y1,z1)| -E(x,v,x1,v1)| -E(y,v,y1,v1)| -T(x,y,z)| -T(x1,y1,z1)|x=y|E(z,v,z1,v1).
8 [] -T(xa,xp,xc)| -T(xb,xq,xc)|T(xp,ip(xa,xp,xc,xb,xq),xb).
9 [] -T(xa,xp,xc)| -T(xb,xq,xc)|T(xq,ip(xa,xp,xc,xb,xq),xa).
13 [] E(x,y,x,y).
14 [] -E(xa,xb,xc,xd)|E(xc,xd,xa,xb).
15 [] -E(xa,xb,xc,xd)|E(xb,xa,xc,xd).
16 [] -E(xa,xb,xc,xd)| -E(xc,xd,xe,xf)|E(xa,xb,xe,xf).
17 [] -E(xa,xb,xc,xd)|E(xa,xb,xd,xc).
18 [] E(x,x,y,y).
19 [] -T(xa,xb,xc)| -T(xa1,xb1,xc1)| -E(xa,xb,xa1,xb1)| -E(xb,xc,xb1,xc1)|E(xa,xc,xa1,xc1).
21 [] T(x,y,y).
23 [] -T(xa,xb,xc)|T(xc,xb,xa).
24 [] T(xa,xa,xb).
26 [] -T(xa,xb,xd)| -T(xb,xc,xd)|T(xa,xb,xc).
28 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xb,xc,xd).
32 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xa,xb,xd).
33 [] -T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xc,xd).
35 [] -IFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1)|E(xb,xd,xb1,xd1).
36 [] -T(xa,xb,xc)| -T(xa1,xb1,xc1)| -E(xa,xc,xa1,xc1)| -E(xb,xc,xb1,xc1)|E(xa,xb,xa1,xb1).
48 [] -T(xa,xb,xc)| -T(za,zb,zc)| -E(xa,xc,za,zc)| -E(xb,xc,zb,zc)|
-E(xa,xd,za,zd)| -E(xc,xd,zc,zd)|IFS(xa,xb,xc,xd,za,zb,zc,zd).
52 [] -E(xa1,xa2,xb1,xb2)| -E(xa1,xa3,xb1,xb3)| -E(xa2,xa3,xb2,xb3)|E3(xa1,xa2,xa3,xb1,xb2,xb3).
57 [] -T(xa,xb,xc)| -E3(xa,xb,xc,xa1,xb1,xc1)|T(xa1,xb1,xc1).
59 [] Col(xa,xb,xc)| -T(xa,xb,xc).
60 [] Col(xa,xb,xc)| -T(xb,xc,xa).
61 [] Col(xa,xb,xc)| -T(xc,xa,xb).
65 [] -Col(x,y,z)|Col(y,x,z).
76 [] xa=xb| -Col(xa,xb,xc)| -E(xa,xp,xa,xq)| -E(xb,xp,xb,xq)|E(xc,xp,xc,xq).
77 [] xa=xb| -Col(xa,xb,xc)| -E(xa,xc,xa,xc1)| -E(xb,xc,xb,xc1)|xc=xc1.
98 [] a!=b.
99 [] T(a,b,c).
100 [] T(a,b,d).
101 [] -T(a,c,d)|$ANS(CON1).
102 [] -T(a,d,c)|$ANS(CON2).
103 [] c1=ext(a,d,c,d).
104 [] d1=ext(a,c,c,d).
105 [] b1=ext(a,c1,c,b).
106 [] b2=ext(a,d1,d,b).
107 [] e=ip(c1,d,b,d1,c).
108 [] p=ext(e,c,c,d).
109 [] r=ext(d1,c,e,c).
110 [] q=ext(p,r,p,r).
493 [hyper,28,100,4] T(b,d,ext(a,d,x,y)).
584 [para_from,103.1.2,5.1.2] E(d,c1,c,d).
585 [para_from,103.1.2,4.1.3] T(a,d,c1).
588 [para_from,104.1.2,5.1.2] E(c,d1,c,d).
589 [para_from,104.1.2,4.1.3] T(a,c,d1).
593 [para_from,105.1.2,5.1.2] E(c1,b1,c,b).
594 [para_from,105.1.2,4.1.3] T(a,c1,b1).
598 [para_from,106.1.2,5.1.2] E(d1,b2,d,b).
599 [para_from,106.1.2,4.1.3] T(a,d1,b2).
600 [para_into,107.1.2.1,103.1.1] e=ip(ext(a,d,c,d),d,b,d1,c).

604 [para_from,108.1.2,5.1.2] E(c,p,c,d).
605 [para_from,108.1.2,4.1.3] T(e,c,p).
609 [para_from,109.1.2,5.1.2] E(c,r,e,c).
610 [para_from,109.1.2,4.1.3] T(d1,c,r).
617 [para_from,110.1.2,5.1.2] E(r,q,p,r).
618 [para_from,110.1.2,4.1.3] T(p,r,q).
624 [binary,493.1,23.1] T(ext(a,d,x,y),d,b).
640 [binary,584.1,14.1] E(c,d,d,c1).
660 [hyper,2,584,1] E(c,d,c1,d).
714 [hyper,32,100,585] T(a,b,c1).
739 [ur,26,585,102] -T(d,c,c1)|$ANS(CON2).
743 [binary,588.1,15.1] E(d1,c,c,d).
744 [binary,588.1,14.1] E(c,d,c,d1).
764 [hyper,2,588,1] E(c,d,d1,c).
765 [binary,589.1,61.2] Col(c,d1,a).
807 [hyper,28,99,589] T(b,c,d1).
830 [ur,26,589,101] -T(c,d,d1)|$ANS(CON1).
834 [binary,593.1,15.1] E(b1,c1,c,b).
947 [binary,598.1,17.1] E(d1,b2,b,d).
1012 [hyper,32,589,599] T(a,c,b2).
1015 [hyper,28,589,599] T(c,d1,b2).
1038 [para_from,600.1.2,8.3.2,unit_del,624] -T(d1,c,b)|T(d,e,d1).
1065 [binary,604.1,17.1] E(c,p,d,c).
1090 [hyper,2,604,1] E(c,d,p,c).
1109 [binary,605.1,23.1] T(p,c,e).
1155 [binary,609.1,17.1] E(c,r,c,e).
1179 [hyper,2,609,1] E(e,c,r,c).
1181 [binary,610.1,61.2] Col(c,r,d1).
1375 [para_into,624.1.1,103.1.2] T(c1,d,b).
1377 [binary,640.1,15.1] E(d,c,d,c1).
1449 [hyper,16,588,660] E(c,d1,c1,d).
1496 [hyper,32,714,594] T(a,b,b1).
1499 [hyper,28,714,594] T(b,c1,b1).
1565 [ur,57,21,739] $ANS(CON2)| -E3(x,y,y,d,c,c1).
1641 [hyper,52,588,744,1] E3(c,d1,d,c,d,d1).
1651 [hyper,16,604,744] E(c,p,c,d1).
1654 [hyper,2,660,744] E(c1,d,c,d1).
1686 [binary,807.1,61.2] Col(c,d1,b).
1704 [binary,807.1,23.1] T(d1,c,b).
1798 [ur,57,21,830] $ANS(CON1)| -E3(x,y,y,c,d,d1).
1809 [binary,834.1,17.1] E(b1,c1,b,c).
1858 [hyper,2,947,1] E(b,d,b2,d1).
1905 [hyper,32,99,1012] T(a,b,b2).
1908 [hyper,28,99,1012] T(b,c,b2).
2027 [binary,1038.1,23.2,unit_del,807] T(d,e,d1).
2046 [binary,1038.2,23.1,unit_del,1704] T(d1,e,d).
2095 [hyper,16,743,1090] E(d1,c,p,c).
2382 [hyper,19,1015,1375,1449,598] E(c,b2,c1,b).
2403 [binary,1496.1,59.2] Col(a,b,b1).
2504 [binary,1499.1,23.1] T(b1,c1,b).
2567 [ur,52,1,640,1565] $ANS(CON2)| -E(d,d,c,c1).
2569 [binary,1641.1,57.2,unit_del,830] -T(c,d1,d)|$ANS(CON1).
2684 [hyper,9,1375,1704] T(c,ip(c1,d,b,d1,c),c1).

2713 [ur,52,743,1,1798] $ANS(CON1)| -E(c,c,d,d1).
2720 [binary,1809.1,14.1] E(b,c,b1,c1).
3124 [hyper,6,2095,1155,1,1651,610,1109] d1=c|E(r,p,e,d1).
3225 [hyper,26,2504,1375] T(b1,c1,d).
3286 [ur,16,18,2567] $ANS(CON2)| -E(x,x,c,c1).
3371 [para_into,2684.1.2,107.1.2] T(c,e,c1).
3396 [ur,16,18,2713] $ANS(CON1)| -E(x,x,d,d1).
3419 [hyper,19,1908,2504,2720,2382] E(b,b2,b1,b).
3443 [para_from,3124.1.1,2569.1.2,unit_del,24] $ANS(CON1)|E(r,p,e,d1).
3496 [hyper,19,3225,807,1809,1654] E(b1,d,b,d1).
3569 [binary,3371.1,23.1] T(c1,e,c).
3667 [binary,3419.1,17.1] E(b,b2,b,b1).
3710 [binary,3443.1,17.1] $ANS(CON1)|E(r,p,d1,e).
3713 [binary,3443.1,15.1] $ANS(CON1)|E(p,r,e,d1).
3823 [binary,3569.1,60.2] Col(c,c1,e).
3859 [hyper,33,3569,605] e=c|T(c1,c,p).
3914 [binary,3667.1,14.1] E(b,b1,b,b2).
3974 [hyper,2,3710,1] $ANS(CON1)|E(d1,e,p,r).
4007 [hyper,16,617,3713] $ANS(CON1)|E(r,q,e,d1).
4032 [binary,3859.2,59.2] e=c|Col(c1,c,p).
4158 [hyper,19,1496,1905,13,3914] E(a,b1,a,b2).
4220 [binary,4007.1,14.1] $ANS(CON1)|E(e,d1,r,q).
4355 [hyper,77,2403,4158,3914,unit_del,98] b1=b2.
4416 [hyper,16,3443,4220,factor_simp] $ANS(CON1)|E(r,p,r,q).
4440 [para_from,4355.1.1,3496.1.1] E(b2,d,b,d1).
4443 [para_from,4355.1.1,2504.1.1] T(b2,c1,b).
4558 [hyper,48,1908,4443,1,2382,1858,4440] IFS(b,c,b2,d,b2,c1,b,d1).
4624 [binary,4558.1,35.1] E(c,d,c1,d1).
4687 [hyper,2,4624,764] E(c1,d1,d1,c).
4689 [hyper,2,4624,660] E(c1,d1,c1,d).
4722 [hyper,2,4687,1] E(d1,c,d1,c1).
4741 [hyper,76,3823,588,4689] c=c1|E(e,d1,e,d).
4865 [hyper,48,2027,2027,13,13,1377,4722] IFS(d,e,d1,c,d,e,d1,c1).
4895 [para_from,4741.1.1,102.1.3,unit_del,585] $ANS(CON2)|E(e,d1,e,d).
4972 [binary,4865.1,35.1] E(e,c,e,c1).
4974 [binary,4895.1,17.1] $ANS(CON2)|E(e,d1,d,e).
5025 [hyper,2,4895,4220] $ANS(CON2)|E(e,d,r,q)|$ANS(CON1).
5088 [hyper,2,1179,4972] E(r,c,e,c1).
5091 [ur,19,3371,13,4972,3286] -T(c,e,c)|$ANS(CON2).
5127 [hyper,16,3713,4974] $ANS(CON2)|E(p,r,d,e)|$ANS(CON1).
5132 [ur,36,24,4974,1,3396] $ANS(CON2)| -T(d,d1,e)|$ANS(CON1).
5186 [hyper,6,3974,5025,2095,1179,2046,618,factor_simp] $ANS(CON2)|
$ANS(CON1)|d1=e|E(d,c,q,c).
5219 [ur,19,3371,1155,5088,3286] -T(c,r,c)|$ANS(CON2).
5285 [para_into,5091.1.1,4032.1.2,unit_del,24] $ANS(CON2)|Col(c1,c,p).
5313 [hyper,19,618,2027,5127,4007,factor_simp] $ANS(CON2)|$ANS(CON1)|E(p,q,d,d1).
5435 [para_from,5186.1.1,5132.2.2,unit_del,21,factor_simp,factor_simp]
$ANS(CON2)|$ANS(CON1)|E(d,c,q,c).
5494 [binary,5285.1,65.1] $ANS(CON2)|Col(c,c1,p).
5709 [ur,16,5313,3396,factor_simp] $ANS(CON2)|$ANS(CON1)| -E(x,x,p,q).
5772 [binary,5435.1,17.1] $ANS(CON2)|$ANS(CON1)|E(d,c,c,q).
5949 [hyper,16,1065,5772] $ANS(CON2)|$ANS(CON1)|E(c,p,c,q).
6043 [hyper,76,1181,5949,4416,factor_simp] $ANS(CON2)|$ANS(CON1)|c=r|E(d1,p,d1,q).

6082 [para_from,6043.1.1,5219.1.3,unit_del,21,factor_simp] $ANS(CON2)|$ANS(CON1)|E(d1,p,d1,q).
6105 [hyper,76,1686,5949,6082,factor_simp,factor_simp] $ANS(CON2)|$ANS(CON1)|c=d1|E(b,p,b,q).
6106 [hyper,76,765,5949,6082,factor_simp,factor_simp] $ANS(CON2)|$ANS(CON1)|c=d1|E(a,p,a,q).
6148 [para_from,6105.1.1,2569.1.1,unit_del,24,factor_simp] $ANS(CON1)|$ANS(CON2)|E(b,p,b,q).
6157 [para_from,6106.1.1,2569.1.1,unit_del,24,factor_simp] $ANS(CON1)|$ANS(CON2)|E(a,p,a,q).
6298 [hyper,6,13,13,6157,6148,714,714,unit_del,98,factor_simp,factor_simp]
 $ANS(CON1)|$ANS(CON2)|E(c1,p,c1,q).
6342 [hyper,76,5494,5949,6298,unit_del,5709,factor_simp,factor_simp,
 factor_simp,factor_simp,factor_simp] $ANS(CON1)|$ANS(CON2)|c=c1.
6409 [para_from,6342.1.1,5091.1.3,factor_simp] -T(c,e,c1)|$ANS(CON2)|$ANS(CON1).
6410 [binary,6409.1,3371.1] $ANS(CON2)|$ANS(CON1).

With the input file that led to this proof forthcoming very soon, a few notes about this proof and a bit about how it was finally found are in order. Ancestor subsumption was in use. The output file displays proof lengths, in order, of 43, 51, 52, 125, 125, 122, 122, and 125. Yes, something was up. In view of the cited proof lengths, have you guessed what was happening in the attempt to find this 122-step proof? You have just a couple of seconds before learning of the truth, as Overbeek would say.

By the way, the proof I just included is the first of the two 122-step proofs. The first three proofs are forward, proofs of three of the unwanted Beeson lemmas; the last five are bidirectional, proofs of connectivity. The lengths of the first three proofs are each less than 60, whereas the lengths of the last five each are equal to or exceed 122. The answer to the small mystery about what was happening is that, in contrast to most of my experiments that proved connectivity of betweenness dependent, not all of the five lemmas (unwanted by Beeson) RG1 through RG5 were proved. Indeed, RG2, RG3, and RG5 were proved, but not 1 or 4. My guess as to why they were not proved focuses on the presence of some demodulators in the following input file, the one that yielded the given 122-step proof. A glance at the just-given proof shows that eight points, defined by Beeson, were relied upon, which contrast with just five when the first proof of connectivity was found, a proof relying on all five of the so-called unwanted lemmas. The three points that are used in the 122-step proof, which are not used in the first proof (that of length 104), are those for b1, b2, and e. As for a comparison of the given 122-step proof with (what I am told is) the Szmielew proof or at least thirty-six steps are present in the Szmielew proof that are not among those of the 122-step proof. Clearly satisfying to me is this sharp difference between OTTER's proof and the proof in SST, illustrating some of what can happen with proof finding as opposed to proof checking.

Input File Producing the 122-Step Proof

```
% Tarski-Szmielew's axiom system
% T is Tarski's B, non-strict betweenness
% E is equidistance
% Names for the axioms as in SST.
% Tries to prove Satz 5.1 assuming things proved in sst5a.in and sst5a2.in and sst5a3.in as well as b!=c.
% Contains more subgoals in list(passive). These should suffice to finish off Satz 5.1.


set(hyper_res).
set(para_into).
set(para_from).
set(binary_res).
set(ur_res).
% set(unit_deletion).
set(order_history).
assign(report,5400).
assign(max_seconds,7).
assign(max_mem,840000).
```

```
clear(print_kept).
%set(very_verbose).
set(input_sos_first).
set(ancestor_subsume).
set(back_sub).
%  set(sos_queue).



assign(max_weight,11).
assign(max_distinct_vars,4).
assign(pick_given_ratio,4).
assign(max_proofs,8).
assign(heat,0).
assign(bsub_hint_wt,1).
set(keep_hint_subsumers).

weight_list(pick_and_purge).
weight(junk,1000).
end_of_list.



list(usable).
E(x,y,y,x).                              % A1 from page 10 of sst
-E(x,y,z,v) | -E(x,y,z2,v2) | E(z,v,z2,v2).   % A2
-E(x,y,z,z) | x=y.                       % A3
T(x,y,ext(x,y,w,v)).                     % A4, first half
E(y,ext(x,y,w,v),w,v).                   % A4, second half
-E(x,y,x1,y1) | -E(y,z,y1,z1) | -E(x,v,x1,v1) | -E(y,v,y1,v1) |
-T(x,y,z) | -T(x1,y1,z1) | x=y | E(z,v,z1,v1).  % A5
-T(x,y,x) | x=y.                         % A6
-T(xa,xp,xc) | -T(xb,xq,xc) | T(xp,ip(xa,xp,xc,xb,xq),xb).
-T(xa,xp,xc) | -T(xb,xq,xc) | T(xq,ip(xa,xp,xc,xb,xq),xa).
-T(alpha,beta,gamma).  %A8,  three lines.
-T(beta,gamma,alpha).
-T(gamma,alpha,beta).
% We don't need more of Tarski's axioms than that here.
E(x,y,x,y).  % Satz 2.1
-E(xa,xb,xc,xd) | E(xc,xd,xa,xb).  % Satz 2.2
-E(xa,xb,xc,xd) | E(xb,xa,xc,xd).  % Satz 2.4
-E(xa,xb,xc,xd) | -E(xc,xd,xe,xf) | E(xa,xb,xe,xf).  % Satz 2.3
-E(xa,xb,xc,xd) | E(xa,xb,xd,xc).  % Satz 2.5
E(x,x,y,y).  % Satz 2.8
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xb,xa1,xb1) |
-E(xb,xc,xb1,xc1) | E(xa,xc,xa1,xc1).  % Satz 2.11
xq = xa | -T(xq,xa,u) | -E(xa,u,xc,xd) | ext(xq,xa,xc,xd) = u.    % Satz 2.12
T(x,y,y).    % Satz 3.1
-E(u,v,x,x) | u=v.   % Not one of Szmielew's theorems but we proved it.
-T(xa,xb,xc) | T(xc,xb,xa).  % Satz 3.2.
T(xa,xa,xb).  % Satz 3.3
-T(xa,xb,xc) | -T(xb,xa,xc) | xa = xb.  % Satz 3.4.
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xb,xc).  % Satz 3.51.
```

```
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xc,xd).  % Satz 3.52.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd).  % Satz 3.61.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd).  % Satz 3.61.
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd).  % Satz 3.71
alpha != beta.   %related to Satz 3.14;  easily provable if added to sst 3h.in.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd).  % Satz 3.62.
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd).  % Satz 3.71
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd).  % Satz 3.72
-IFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xd,xb1,xd1).  % Satz 4.2
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xc,xa1,xc1)
| -E(xb,xc,xb1,xc1) | E(xa,xb,xa1,xb1).  % Satz 4.3

alpha != beta.  % Satz 3.13
beta != gamma.
alpha != gamma.
T(xa,xb,ext(xa,xb,alpha,gamma)).   % Satz 3.14, first half
xb != ext(xa,xb,alpha,gamma).    % Satz 3.14, second half
% The following many clauses are Definition 4.1
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | T(xa,xb,xc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | T(za,zb,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xa,xc,za,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xb,xc,zb,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xa,xd,za,zd).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xc,xd,zc,zd).
-T(xa,xb,xc) | -T(za,zb,zc) | -E(xa,xc,za,zc) | -E(xb,xc,zb,zc)
| -E(xa,xd,za,zd) | -E(xc,xd,zc,zd) | IFS(xa,xb,xc,xd,za,zb,zc,zd).

% Following 4 are definition 4.4 for n=3
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa1,xa2,xb1,xb2).
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa1,xa3,xb1,xb3).
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa2,xa3,xb2,xb3).
-E(xa1,xa2,xb1,xb2) | -E(xa1,xa3,xb1,xb3) | -E(xa2,xa3,xb2,xb3)
| E3(xa1,xa2,xa3,xb1,xb2,xb3).
% Following three lines are Satz 4.5
-T(xa,xb,xc) | -E(xa,xc,xa1,xc1) | T(xa1,insert(xa,xb,xa1,xc1),xc1).
-T(xa,xb,xc) | -E(xa,xc,xa1,xc1) | E3(xa,xb,xc,xa1,insert(xa,xb,xa1,xc1),xc1).
insert(xa,xb,xa1,xc1) = ext(ext(xc1,xa1,alpha,gamma),xa1,xa,xb).
-E3(x,y,z,u,v,w) | E3(x,z,y,u,w,v).   % See sst4q.in, not in Szmielew
-T(xa,xb,xc) | -E3(xa,xb,xc,xa1,xb1,xc1) | T(xa1,xb1,xc1).  % Satz 4.6

% following is Definition 4.10
-Col(xa,xb,xc) | T(xa,xb,xc) | T(xb,xc,xa) | T(xc,xa,xb).
Col(xa,xb,xc) | -T(xa,xb,xc).
Col(xa,xb,xc) | -T(xb,xc,xa).
Col(xa,xb,xc) | -T(xc,xa,xb).
% Following are Satz 4.11
-Col(x,y,z) | Col(y,z,x).
-Col(x,y,z) | Col(z,x,y).
-Col(x,y,z) | Col(z,y,x).
-Col(x,y,z) | Col(y,x,z).
-Col(x,y,z) | Col(x,z,y).
% following is Satz 4.12
Col(x,x,y).
```

```
% following is Satz 4.13
-Col(xa,xb,xc) | - E3(xa,xb,xc,xa1,xb1,xc1) | Col(xa1,xb1,xc1).
% following is Satz 4.14
-Col(xa,xb,xc) | -E(xa,xb,xa1,xb1)
| E3(xa,xb,xc,xa1,xb1,insert5(xa,xb,xc,xa1,xb1)).
% following is Definition 4.15
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | Col(xa,xb,xc).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E3(xa,xb,xc,xa1,xb1,xc1).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xa,xd,xa1,xd1).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xd,xb1,xd1).
-Col(xa,xb,xc) | - E3(xa,xb,xc,xa1,xb1,xc1) | - E(xa,xd,xa1,xd1)
 | -E(xb,xd,xb1,xd1) | FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1).
% Following is Satz 4.16
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | xa = xb | E(xc,xd,xc1,xd1).
% Following is Satz 4.17
xa = xb | -Col(xa,xb,xc) | -E(xa,xp,xa,xq) | -E(xb,xp,xb,xq) | E(xc,xp,xc,xq).
% Following is Satz 4.18
xa = xb | -Col(xa,xb,xc) | -E(xa,xc,xa,xc1) | -E(xb,xc,xb,xc1) | xc = xc1.
% Following is Satz 4.19
-T(xa,xc,xb) | -E(xa,xc,xa,xc1) | -E(xb,xc,xb,xc1) | xc = xc1.
% Following defines T5
T5(x,y,z,u,v) | -T(x,y,z) | -T(x,y,u) | -T(x,y,v)
| -T(x,z,u) | -T(x,z,v) | -T(x,u,v).
-T5(x,y,z,u,v) | T(x,y,z).
-T5(x,y,z,u,v) | T(x,y,u).
-T5(x,y,z,u,v) | T(x,y,v).
-T5(x,y,z,u,v) | T(x,z,u).
-T5(x,y,z,u,v) | T(x,z,v).
-T5(x,y,z,u,v) | T(x,u,v).
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xb,xa1,xb1)
| -E(xb,xc,xb1,xc1) | -E(xa,xd,xa1,xd1) | -E(xb,xd,xb1,xd1)
| AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | T(xa,xb,xc).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | T(xa1,xb1,xc1).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xa,xb,xa1,xb1).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xc,xb1,xc1).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xd,xb1,xd1).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1).
% Following proved in sst5a.in
%  T5(a,b,c,d1,b2).
%  T5(a,b,d,c1,b1).
%  E(b,c1,b2,c).
%  E(b,b1,b2,b).
%  b2 = b1.
%  T(c,e,c1).
%  T(d,e,d1).
% following proved in sst5a2.in
%  E(c1,d1,c,d).   % by cases according as b = c or b != c.
%  IFS(d,e,d1,c,d,e,d1,c1).
%  IFS(c,e,c1,d,c,e,c1,d1).
% Following proved in sst5a3.in
%  E(e,c,e,c1).
%  E(e,d,e,d1).
```

```
end_of_list.

list(passive).
-AFS(b,c,d1,c1,b1,c1,d,c) | $ANS(TARG1INT).
-E(c1,d1,c,d) | $ANS(TARG2INT).
-IFS(d,e,d1,c,d,e,d1,c1) | $ANS(TARG3INT).
-IFS(c,e,c1,d,c,e,c1,d1) | $ANS(TARG4INT).
-E(e,c,e,c1) | $ANS(TARG5INT).
% % following 10 are later steps in proofs of 38 and F, temp.beeson.sst5a4.out26
end_of_list.

list(sos).
% c != c1. % we assume this; separately we should
    %prove the theorem in case c = c1, which should be easy.
a != b.
T(a,b,c).
T(a,b,d).
-T(a,c,d) | $ANS(CON1).
-T(a,d,c) | $ANS(CON2).
c1 = ext(a,d,c,d).
d1 = ext(a,c,c,d).
b1 = ext(a,c1,c,b).
b2 = ext(a,d1,d,b).
e = ip(c1,d,b,d1,c).
p = ext(e,c,c,d).
r = ext(d1,c,e,c).
q = ext(p,r,p,r).
% c != b.
% c != c1.
% c != d1.
end_of_list.

list(hot).
end_of_list.

list(hints2).
% following 123 prove con, without c!=b used
T(b,d,ext(a,d,x,y)).
E(d,c1,c,d).
T(a,d,c1).
E(c,d1,c,d).
T(a,c,d1).
E(c1,b1,c,b).
T(a,c1,b1).
E(d1,b2,d,b).
T(a,d1,b2).
e=ip(ext(a,d,c,d),d,b,d1,c).
E(c,p,c,d).
T(e,c,p).
E(c,r,e,c).
T(d1,c,r).
E(r,q,p,r).
T(p,r,q).
```

```
T(ext(a,d,x,y),d,b).
E(c,d,d,c1).
E(c,d,c1,d).
T(a,b,c1).
-T(d,c,c1)|$ANS(CON2).
E(d1,c,c,d).
E(c,d,c,d1).
E(c,d,d1,c).
Col(c,d1,a).
T(b,c,d1).
-T(c,d,d1)|$ANS(CON1).
E(b1,c1,c,b).
E(d1,b2,b,d).
T(a,c,b2).
T(c,d1,b2).
-T(d1,c,b)|T(d,e,d1).
E(c,p,d,c).
E(c,d,p,c).
T(p,c,e).
E(c,r,c,e).
E(e,c,r,c).
Col(c,r,d1).
E(r,q,r,p).
T(c1,d,b).
E(d,c,d,c1).
E(c,d1,c1,d).
T(a,b,b1).
T(b,c1,b1).
$ANS(CON2)| -E3(x,y,y,d,c,c1).
E(c,p,c,d1).
E(c1,d,c,d1).
Col(c,d1,b).
T(d1,c,b).
$ANS(CON1)| -E3(x,y,y,c,d,d1).
E(b1,c1,b,c).
E(b,d,b2,d1).
T(a,b,b2).
T(b,c,b2).
T(d,e,d1).
T(d1,e,d).
E(d1,c,p,c).
E(r,p,r,q).
E(c,b2,c1,b).
Col(a,b,b1).
T(b1,c1,b).
$ANS(CON2)| -E(d,d,c,c1).
T(c,ip(c1,d,b,d1,c),c1).
$ANS(CON1)| -E(c,c,d,d1).
E(b,c,b1,c1).
d1=c|E(r,p,e,d1).
T(b1,c1,d).
$ANS(CON2)| -E(x,x,c,c1).
T(c,e,c1).
```

$ANS(CON1)| -E(x,x,d,d1).
E(b,b2,b1,b).
E(b1,d,b,d1).
$ANS(CON2)| -E(x,x,c1,c).
T(c1,e,c).
E(b,b2,b,b1).
Col(c,c1,e).
e=c|T(c1,c,p).
E(b,b1,b,b2).
e=c|Col(c1,c,p).
E(a,b1,a,b2).
b1=b2.
E(b2,d,b,d1).
T(b2,c1,b).
IFS(b,c,b2,d,b2,c1,b,d1).
E(c,d,c1,d1).
E(c1,d1,d1,c).
E(c1,d1,c1,d).
-T(c1,c,d1)|$ANS(CON2).
E(d1,c,d1,c1).
c=c1|E(e,d1,e,d).
$ANS(CON2)|E(r,p,e,d1).
IFS(d,e,d1,c,d,e,d1,c1).
$ANS(CON2)|E(e,d1,e,d).
$ANS(CON2)|E(r,p,d1,e).
$ANS(CON2)|E(p,r,e,d1).
$ANS(CON2)|E(r,q,e,d1).
$ANS(CON2)|E(e,d1,r,q).
E(e,c,e,c1).
$ANS(CON2)|E(e,d1,d,e).
$ANS(CON2)|E(d1,e,p,r).
$ANS(CON2)|E(e,d,r,q).
E(r,c,e,c1).
-T(c,e,c)|$ANS(CON2).
$ANS(CON2)|E(p,r,d,e).
$ANS(CON2)| -T(d,d1,e)|$ANS(CON1).
$ANS(CON2)|d1=e|E(d,c,q,c).
-T(c,r,c)|$ANS(CON2).
$ANS(CON2)|Col(c1,c,p).
$ANS(CON2)|E(p,q,d,d1).
$ANS(CON2)|$ANS(CON1)|E(d,c,q,c).
$ANS(CON2)|Col(c,c1,p).
$ANS(CON2)| -E(x,x,p,q)|$ANS(CON1).
$ANS(CON2)|$ANS(CON1)|E(d,c,c,q).
$ANS(CON2)|$ANS(CON1)|E(c,p,c,q).
$ANS(CON2)|$ANS(CON1)|c=r|E(d1,p,d1,q).
$ANS(CON2)|$ANS(CON1)|E(d1,p,d1,q).
$ANS(CON2)|$ANS(CON1)|c=d1|E(b,p,b,q).
$ANS(CON2)|$ANS(CON1)|c=d1|E(a,p,a,q).
$ANS(CON2)|$ANS(CON1)|E(b,p,b,q).
$ANS(CON2)|$ANS(CON1)|E(a,p,a,q).
$ANS(CON2)|$ANS(CON1)|E(c1,p,c1,q).
$ANS(CON2)|$ANS(CON1)|c=c1.

-T(c,e,c1)|$ANS(CON2)|$ANS(CON1).
% Following 135 prove conn of betw, may not be using trans of betw, temp.beeson.sst5a4.out40t 36ha
T(b,d,ext(a,d,x,y)).
E(d,c1,c,d).
T(a,d,c1).
E(c,d1,c,d).
T(a,c,d1).
E(c1,b1,c,b).
T(a,c1,b1).
E(d1,b2,d,b).
T(a,d1,b2).
e=ip(ext(a,d,c,d),d,b,d1,c).
E(c,p,c,d).
T(e,c,p).
E(c,r,e,c).
T(d1,c,r).
E(r,q,p,r).
T(p,r,q).
T(ext(a,d,x,y),d,b).
E(d,c1,d,c).
E(c,d,d,c1).
E(c,d,c1,d).
T(a,b,c1).
-T(d,c,c1)|$ANS(CON2).
E(d1,c,c,d).
E(c,d,c,d1).
E(c,d,d1,c).
Col(c,d1,a).
T(b,c,d1).
-T(c,d,d1)|$ANS(CON1).
E(b1,c1,c,b).
E(d1,b2,b,d).
T(a,c,b2).
T(c,d1,b2).
-T(d1,c,b)|T(d,e,d1).
E(c,p,d,c).
E(c,d,p,c).
T(p,c,e).
E(c,r,c,e).
E(r,c,e,c).
E(e,c,r,c).
Col(c,r,d1).
E(r,q,r,p).
T(c1,d,b).
E(d,c,d,c1).
E(c,d1,c1,d).
Col(a,b,c1).
T(a,b,b1).
T(b,c1,b1).
$ANS(CON2)| -E3(x,y,y,d,c,c1).
E3(c,d1,d,c,d,d1).
E(c,p,c,d1).
E(c1,d,c,d1).

Col(c,d1,b).
T(d1,c,b).
$ANS(CON1)| -E3(x,y,y,c,d,d1).
E(b1,c1,b,c).
T(a,b,b2).
T(b,c,b2).
E(d1,c,p,c).
E(r,p,r,q).
E(c,b2,c1,b).
Col(a,b,b1).
T(b1,c1,b).
$ANS(CON2)| -E(d,d,c,c1).
-T(c,d1,d)|$ANS(CON1).
T(c,ip(c1,d,b,d1,c),c1).
$ANS(CON1)| -E(c,c,d,d1).
E(b,c,b1,c1).
d1=c|E(r,p,e,d1).
T(b1,c1,d).
$ANS(CON2)| -E(x,x,c,c1).
T(c,e,c1).
$ANS(CON1)| -E(x,x,d,d1).
E(b,b2,b1,b).
$ANS(CON1)|E(r,p,e,d1).
E(b1,d,b,d1).
$ANS(CON2)| -E(x,x,c1,c).
$ANS(CON2)| -E(c1,c,x,x).
T(c1,e,c).
E(b,b2,b,b1).
$ANS(CON1)|E(r,p,d1,e).
$ANS(CON1)|E(p,r,e,d1).
$ANS(CON1)|E(r,q,e,d1).
$ANS(CON1)|E(e,d1,r,q).
Col(c,c1,e).
e=c|T(c1,c,p).
E(b,b1,b,b2).
e=c|Col(c1,c,p).
E(a,b1,a,b2).
b1=b2.
E(b2,d,b,d1).
T(b2,c1,b).
E(b,d,b2,d1).
IFS(b,c,b2,d,b2,c1,b,d1).
E(c,d,c1,d1).
$ANS(CON1)|E(d1,e,p,r).
E(c1,d1,d1,c).
E(c1,d1,c1,d).
-T(c1,c,d1)|$ANS(CON2).
E(d1,c,d1,c1).
c=c1|E(e,d1,e,d).
$ANS(CON2)| -T(d1,c,c1).
$ANS(CON2)|E(e,d1,e,d).
$ANS(CON2)|E(e,d1,d,e).
$ANS(CON2)|E(e,d,r,q)|$ANS(CON1).

```
$ANS(CON2)|E(p,r,d,e)|$ANS(CON1).
$ANS(CON2)| -T(d,d1,e)|$ANS(CON1).
$ANS(CON2)|$ANS(CON1)| -T(d1,e,d1).
T(d,e,d1).
T(d1,e,d).
IFS(d,e,d1,c,d,e,d1,c1).
E(p,q,d,d1)|$ANS(CON2)|$ANS(CON1).
d1=e|E(d,c,q,c)|$ANS(CON1)|$ANS(CON2).
E(e,c,e,c1).
$ANS(CON2)|$ANS(CON1)| -E(x,x,p,q).
$ANS(CON2)|$ANS(CON1)|E(d,c,q,c).
$ANS(CON2)|$ANS(CON1)|E(d,c,c,q).
E(r,c,e,c1).
-T(c,e,c)|$ANS(CON2).
$ANS(CON2)|Col(c1,c,p).
$ANS(CON2)|T(c1,c,p).
$ANS(CON2)|Col(c,c1,p).
$ANS(CON2)|T(p,c,c1).
$ANS(CON2)| -T(p,d1,c).
$ANS(CON2)|$ANS(CON1)|E(c,p,c,q).
$ANS(CON2)|$ANS(CON1)|c=r|E(d1,p,d1,q).
-T(c,r,c)|$ANS(CON2).
$ANS(CON2)|$ANS(CON1)|E(d1,p,d1,q).
$ANS(CON2)|$ANS(CON1)|c=d1|E(b,p,b,q).
$ANS(CON2)|$ANS(CON1)|c=d1|E(a,p,a,q).
$ANS(CON2)|$ANS(CON1)|E(b,p,b,q).
$ANS(CON2)|$ANS(CON1)|E(a,p,a,q).
$ANS(CON2)|$ANS(CON1)|E(c1,p,c1,q).
$ANS(CON2)|$ANS(CON1)|c=c1.
$ANS(CON2)|$ANS(CON1)| -E(c,p,c1,q).
$ANS(CON2)|$ANS(CON1)|E(c,p,c1,q).
end_of_list.

list(demodulators).
EQ(E(r,q,r,p),junk).
EQ(E(r,c,e,c),junk).
%  EQ(T(b,d,c1),junk).
%  EQ(Col(a,b,c1),junk).
%  EQ(E3(c,d1,d,c,d,d1),junk).
EQ(E(d,c1,d,c),junk).
%%EQ(Col(c,d1,a),junk).
%%EQ(E(r,c,e,c),junk).
%%EQ(Col(a,b,c1),junk).
%%EQ(E3(c,d1,d,c,d,d1),junk).
%  EQ(T(b,d,c1),junk).
%  EQ(E(c,e,e,c1),junk).
%  EQ(T(d1,c,a),junk).
%  EQ(T(b1,c1,a),junk).
%  EQ(E(d,c1,d1,c),junk).
%  EQ(E(c,r,e,c1),junk).
end_of_list.
```

The following three demodulators, found in the just-given file, might indeed prevent OTTER from deducing RG1 and RG4.

EQ(E(r,q,r,p),junk).
EQ(E(r,c,e,c),junk).
EQ(E(d,c1,d,c),junk).

These demodulators were used in the iterative process that eventually culminated in the 122-step proof but that were commented out in the end in order to enable OTTER to offer this cited proof. Or, perhaps, the program would have proved those two Beeson unwanted lemmas, RG1 and RG4, if allowed more latitude in various ways. A scholar would most likely investigate this question; I will not, at least at this time. You see that since both clauses for inner Pasch are used explicitly, blocking either or both from use might lead to an interesting proof.

With the use of ancestor subsumption, as you see, you are likely to find more than one proof of any given target in the output file. Such was not the case with the three Beeson lemmas, but was, as shown, the case with connectivity. Had I not included the use of ancestor subsumption, what do you think would have occurred? I and you are about to learn, for I am at this writing running the appropriate input file. I still expect the program to find more than one proof of connectivity because of seeking, in contrast to the five Beeson lemmas, a bidirectional proof.

A winner, if the prize is given for the use of ancestor subsumption. Specifically, the program found proofs of respective lengths 43, 51, 52, 131, 131, 129, 129, and 129. The world still turns: multiple proofs for a target when the choice is bidirectional; shorter proofs when ancestor subsumption is in use. You may have paused to study the 122-step proof and observed that both transitivity and Satz 4.3 are back, are used to find the proof and are found before the deduced clauses are presented. And what amounts to an aside in that all the votes are not yet in, you might enjoy watching (so to speak) research as it occurs. In particular, I am currently running an experiment that closely resembles that which yielded the 122-step proof; namely, I made one change, that of avoiding the use of binary resolution (in my continued tribute to Overbeek, and out of curiosity). I am also running a second experiment, one in which both binary resolution and ancestor subsumption are not in use.

After much CPU time, neither experiment succeeded. Therefore, apparently, binary resolution is needed under reasonable conditions, depending on the theorem to be proved. Of course, appropriate value assignments and parameter settings might lead to a proof in which binary resolution was not used. I leave that study to others. Instead, I now turn to a topic that may indeed remind you of proof shortening.

The OTTER proofs, as seen throughout this notebook, present formulas or equations (clauses) that are taken from an input file before presenting any of the deduced formulas or equations that make up the proof. Alama, if I understand correctly, is most interested in axiom dependence. Many years ago, McCune also made such a study, focusing (if memory serves) on John Kalman's right group calculus. Kalman's axiom system consists of five members; McCune was able to prove four of them dependent, as I recall, depending on which of the five (as it turns out) is chosen to be proved to be a single axiom. (I extended McCune's successful study, in a paper I wrote on the cramming strategy; three of the five can serve as a single axiom, if I have checked my work correctly.) Perhaps Alama's interest, communicated to me by email, served as a wellspring, causing me to consider the following.

In some of the proofs I had of the dependence of connectivity of betweenness (where inner Pasch was present), forty-five or more elements from the corresponding input file were used to obtain the proof. I became interested in what might be called axiom-and-theorem dependence. In particular, the items that preceded the deduced clauses in many of the studies reported here consist of Tarski axioms, from one of such systems, followed by theorems that had already been proved, some by me, some by Beeson, and some by our collaboration. Those theorems were taken from Szmielew.

I began a series of experiments, focusing on the connectivity theorem, to see whether I could find a proof that relied on fewer than forty-five items; and, yes, I did find such a proof, one that depends on just forty-two. My approach was indeed straightforward: Take an input file that proves the dependence of connectivity, choose an item from the proof that occurs in the input file, comment it out (so it is inaccessible as a so-called axiom), and see what happens. Have you formed an opinion about what might occur? Also, have you made your choice for an item, axiom, or proved theorem to avoid using among that which the input file offers? Well, OTTER might fail to get any proof for connectivity, if, for example, the item being blocked is crucial to all proofs. Or, the resulting proof might be longer, in deduced steps, than the one in hand, which would not be a tragedy in that proof length is not the object at this stage. Or—annoying or disappointing—the program might complete a number of proofs, but each might rely on too many input items, axioms or proved theorems.

Because of what you have read so far, you have chosen, perhaps, transitivity of betweenness to block. That was my choice, having in hand a proof of length 151, a proof that relied on transitivity of betweenness, as a proved theorem. All was right on the ship; indeed, OTTER presented me with a proof that relied on forty-four items from the input file, omitting, as demanded, the use of transitivity. The proof—and here you encounter that oddity again—of length not longer than 151, but, instead, of length 146.

You can imagine my wandering around in the maze of items that could be blocked from use, axioms or proved theorems. Some of the experiments produced no proofs. But, rather than my detailing all the ports of call, a summary of where things now stand is in order. At this time, I have a proof that relies on just forty-two axioms and proved theorems. The proof, of length 151, was completed in approximately 133 CPU-seconds with retention of clause (104910). The proof is, of course, bidirectional. (Someday, perhaps, I will have in hand from some source a forward proof of the dependence of connectivity; one of my readers might send me such.) The proof completes, to my amusement, with the deduction of a clause that conflicts with T(x,y,y). The following three items were used in the proof relying on forty-five input items and absent from the proof relying on forty-two items.

```
-T(xa,xb,xd)| -T(xb,xc,xd)|T(xa,xb,xc).
-T(xa,xb,xc)| -T(xa1,xb1,xc1)| -E(xa,xc,xa1,xc1)| -E(xb,xc,xb1,xc1)|E(xa,xb,xa1,xb1).
-Col(x,y,z)|Col(y,x,z).
```

To obtain this result, I had OTTER avoid the use, in the input, of 3.51 (transitivity), Satz 4.3, Satz 4.19, and the five items listed under Satz 4.11. To remove confusion, not all of the cited items were used to produce the proof relying on forty-five items; some came and went along my trail of experiments. Yes, I have tried repeatedly to find a proof that relies on strictly fewer than forty-two input items, axioms and proved theorems, but to no avail. Some of my experiments led to the finding, after a long, long, long CPU time, of a long, long, long proof, but a proof that did not satisfy the objective. Indeed, one of the proofs has length 246, completed in approximately 47,807 CPU-seconds, with retention of clause (1495163). That experiment had OTTER avoid the use of both T(x,x,y) and T(x,y,y). And, so piquant and somewhat startling, rather than less than forty-five items to be used, forty-nine were used. When I tried to combine goals, seeking a proof that was short and that depended on perhaps fewer than forty-two input items, focusing on the already-presented 122-step proof, the best that I could do after a bit of experimentation was a proof of length 129. In a different experiment, I did find a proof of length 122 that relied on but forty-four input items, in contrast to the forty-five relied upon by the given 122-step proof. The input item that can be dispensed with is the following.

```
-Col(x,y,z)|Col(y,x,z).
```

At this point on this labyrinthian journey, I now turn to a study that is more in the spirit of proof checking, according to Beeson and, I assume, others, in contrast to proof finding.

9. Szmielew Plays a Bigger Role, Deducing Outer Pasch

You may well ask why I would focus on proof checking in view of my earlier remarks. The explanation is, in a sense, sad and yet practical. You see, three theorems remain on my (so-to-speak) private list for discussion in this notebook. Specifically, I would like to focus on the theorem that asserts that outer Pasch can be proved from inner Pasch, that inner Pasch can be proved from outer Pasch, and that connectivity is dependent on the axiom system offered in Section 1, the system that relies on outer Pasch rather than on inner. My attempts so far have led to essentially nothing in the context of each of these three theorems. However, I can now focus on the first of the cited three, a deduction of outer Pasch from inner Pasch.

I am able to do so, at the simplest level, because of an input file sent to me by Beeson. At a deeper level, from what he writes, Gupta merits all the credit. Indeed, hus proof provided the necessary steps, to be used as resonators or as hints, that could be used by OTTER to successfully prove the theorem in focus. Yes, so much was taken from Gupta's proof that the study must be termed proof checking. So, you further ask—even though the theorem has been on my list from the outset—why feature it at all. Simply, you may advise, present an input file and a proof. Rather than following that program, I will be concerned here with two aspects. First, I will again discuss proof shortening, of this theorem. Second, in the Alama spirit touched on in Section 1, I will spend some time on minimizing the number of assumptions needed to prove the theorem.

So that you (and Alama and his possible colleagues) can proceed on your own in the context of proof shortening and assumption reduction, I now present the input file sent to me by email from Beeson. (By the way, for the curious who might wish to have even more motivation for studying assumption reduction, you might glance at various theorems that hold for group theory, but also hold for semigroups.)

An Input File for Proving Outer Pasch from Inner

```
% Tarski-Szmielew's axiom system
% T is Tarski's B, non-strict betweenness
% E is equidistance
% Names for the axioms as in SST.
% Assumes key parts of earlier chapters and proves outer Pasch (Satz 9.6)


set(hyper_res).
set(para_into).
set(para_from).
set(binary_res).
set(ur_res).
% set(unit_deletion).
set(order_history).
assign(report,5400).
assign(max_seconds,10000).
assign(max_mem,840000).
%clear(print_kept).
%set(very_verbose).
set(input_sos_first).
set(ancestor_subsume).
% set(sos_queue).



assign(max_weight,12).
assign(max_distinct_vars,5).
assign(pick_given_ratio,4).
assign(max_proofs,1).


weight_list(pick_and_purge).
weight(-T(c,p,q),-1).
weight(-T(q,c,p),-1).
weight(-T(p,q,c),-1).
weight(-opposite(a,p,q,b)|Col(p,q,d),-1).
```

```
weight(-opposite(a,p,q,b)|T(a,d,b),-1).
weight(-T(p,c,a)| -T(b,d,a)|T(d,t,p),-1).
weight(-T(p,c,a)| -T(b,d,a)|T(c,t,b),-1).
weight(Col(c,p,a),-1).
weight(T(p,c,a),-1).
weight(Col(c,b,q),-1).
weight(-T(x,b,q)|b=q|T(x,b,c),-1).
weight(T(c,q,b),-1).
weight(-T(p,q,b),-1).
weight(-T(a,x,b)| -Col(p,q,x)|T(q,x,p)|T(x,p,q),-1).
weight(-T(a,q,b),-1).
weight(-opposite(a,p,q,b)| -T(p,q,d),-1).
weight(-opposite(a,p,q,b)|T(b,d,a),-1).
weight(T(c,p,a)|sameside(c,p,a),-1).
weight(Col(c,a,p),-1).
weight(Col(b,c,q),-1).
weight(T(c,a,p)|sameside(c,a,p),-1).
weight(T(d,t,p)| -T(a,d,b),-1).
weight(-T(a,d,b)|T(p,t,d),-1).
weight(T(d,t,p)| -opposite(a,p,q,b),-1).
weight(T(p,t,d)| -opposite(a,p,q,b),-1).
weight(-T(q,p,q),-1).
weight(-T(p,q,p),-1).
weight(sameside(c,p,a)|p=c,-1).
weight(-T(q,c,q),-1).
weight(-T(a,q,c),-1).
weight(-T(b,c,p),-1).
weight(sameside(c,p,a),-1).
weight(c!=p,-1).
weight(sameside(c,a,p)|a=c,-1).
weight(-T(b,p,q),-1).
weight(-T(p,b,c),-1).
weight(sameside(c,a,p),-1).
weight(c!=a,-1).
weight(T(c,t,b)| -opposite(a,p,q,b),-1).
weight(-opposite(a,p,q,b)|Col(b,c,t),-1).
weight(-opposite(a,p,q,b)| -opposite(x,b,c,t),-1).
weight(-Col(p,q,b)|sameside(p,q,b),-1).
weight(-Col(p,q,b)|T(q,b,p),-1).
weight(-T(p,c,b),-1).
weight(-T(b,q,b),-1).
weight(-Col(p,q,b)|b!=q,-1).
weight(-Col(p,q,b)|T(p,b,q),-1).
weight(-T(b,q,a),-1).
weight(-T(a,c,b),-1).
weight(b=q| -Col(p,q,b),-1).
weight(-Col(p,q,b),-1).
weight(-Col(q,p,q)|sameside(q,p,q),-1).
weight(sameside(q,p,q),-1).
weight(Col(q,p,q),-1).
weight(Col(p,q,q),-1).
weight(p=q|opposite(c,p,q,b),-1).
weight(opposite(c,p,q,b),-1).
```

```
weight(-Col(p,q,p)|sameside(p,q,p),-1).
weight(sameside(p,q,p),-1).
weight(Col(p,q,p),-1).
weight(opposite(a,p,q,b),-1).
weight(T(p,t,d),-1).
weight(T(d,t,p),-1).
weight(-T(p,q,d),-1).
weight(T(a,d,b),-1).
weight(Col(p,q,d),-1).
weight(-T(b,c,q),-1).
weight(-T(a,b,c),-1).
weight(-opposite(t,b,c,x),-1).
weight(-T(c,a,b),-1).
weight(-T(c,b,q),-1).
weight(-T(b,c,a),-1).
weight(-T(c,p,b),-1).
weight(-T(p,q,t),-1).
weight(sameside(b,c,q),-1).
weight(b!=c,-1).
weight(-Col(b,c,a),-1).
weight(sameside(c,b,q),-1).
weight(sameside(c,b,c),-1).
weight(Col(c,b,c),-1).
weight(Col(b,c,c),-1).
weight(Col(b,c,p)|opposite(p,b,c,a),-1).
weight(opposite(p,b,c,a),-1).
weight(-sameside(p,q,t),-1).
weight(-Col(p,q,t),-1).
weight(-T(t,p,q),-1).
weight(-T(q,t,p),-1).
weight(-T(d,p,q),-1).
weight(-T(q,d,p),-1).
end_of_list.
```

```
list(usable).
E(x,y,y,x).                              % A1 from page 10 of sst
-E(x,y,z,v) | -E(x,y,z2,v2) | E(z,v,z2,v2).   % A2
-E(x,y,z,z) | x=y.                       % A3
T(x,y,ext(x,y,w,v)).                     % A4, first half
E(y,ext(x,y,w,v),w,v).                   % A4, second half
-E(x,y,x1,y1) | -E(y,z,y1,z1) | -E(x,v,x1,v1) | -E(y,v,y1,v1) | -T(x,y,z) |
 -T(x1,y1,z1) | x=y | E(z,v,z1,v1).  % A5
-T(x,y,x) | x=y.                         % A6
% A7, inner Pasch, two clauses.
-T(xa,xp,xc) | -T(xb,xq,xc) | T(xp,ip(xa,xp,xc,xb,xq),xb).
-T(xa,xp,xc) | -T(xb,xq,xc) | T(xq,ip(xa,xp,xc,xb,xq),xa).
-T(alpha,beta,gamma).  %A8,  three lines.
-T(beta,gamma,alpha).
-T(gamma,alpha,beta).
% We don't need more of Tarski's axioms than that here.
E(x,y,x,y).  % Satz 2.1
```

```
-E(xa,xb,xc,xd) | E(xc,xd,xa,xb).  % Satz 2.2
-E(xa,xb,xc,xd) | E(xb,xa,xc,xd).  % Satz 2.4
-E(xa,xb,xc,xd) | -E(xc,xd,xe,xf) | E(xa,xb,xe,xf).  % Satz 2.3
-E(xa,xb,xc,xd) | E(xa,xb,xd,xc).  % Satz 2.5
E(x,x,y,y).  % Satz 2.8
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xb,xa1,xb1) | -E(xb,xc,xb1,xc1) | E(xa,xc,xa1,xc1).  % Satz 2.11
xq = xa | -T(xq,xa,u) | -E(xa,u,xc,xd) | ext(xq,xa,xc,xd) = u.    % Satz 2.12
T(x,y,y).    % Satz 3.1
-T(xa,xb,xc) | T(xc,xb,xa).  % Satz 3.2.
T(xa,xa,xb).  % Satz 3.3
-T(xa,xb,xc) | -T(xb,xa,xc) | xa = xb.  % Satz 3.4.
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xb,xc).  % Satz 3.51.
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xc,xd).  % Satz 3.52.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd).  % Satz 3.61.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd).  % Satz 3.61.
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd).  % Satz 3.71
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd).  % Satz 3.62.
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd).  % Satz 3.71
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd).  % Satz 3.72
-IFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xd,xb1,xd1).  % Satz 4.2
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xc,xa1,xc1) | -E(xb,xc,xb1,xc1) | E(xa,xb,xa1,xb1).  % Satz 4.3

alpha != beta.  % Satz 3.13
beta != gamma.
alpha != gamma.
T(xa,xb,ext(xa,xb,alpha,gamma)).   % Satz 3.14, first half
xb != ext(xa,xb,alpha,gamma).    % Satz 3.14, second half
% The following many clauses are Definition 4.1
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | T(xa,xb,xc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | T(za,zb,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xa,xc,za,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xb,xc,zb,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xa,xd,za,zd).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xc,xd,zc,zd).
-T(xa,xb,xc) | -T(za,zb,zc) | -E(xa,xc,za,zc) | -E(xb,xc,zb,zc) |
 -E(xa,xd,za,zd) | -E(xc,xd,zc,zd) | IFS(xa,xb,xc,xd,za,zb,zc,zd).

% Following 4 are definition 4.4 for n=3
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa1,xa2,xb1,xb2).
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa1,xa3,xb1,xb3).
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa2,xa3,xb2,xb3).
-E(xa1,xa2,xb1,xb2) | -E(xa1,xa3,xb1,xb3) | -E(xa2,xa3,xb2,xb3) | E3(xa1,xa2,xa3,xb1,xb2,xb3).
% Following three lines are Satz 4.5
-T(xa,xb,xc) | -E(xa,xc,xa1,xc1) | T(xa1,insert(xa,xb,xa1,xc1),xc1).
-T(xa,xb,xc) | -E(xa,xc,xa1,xc1) | E3(xa,xb,xc,xa1,insert(xa,xb,xa1,xc1),xc1).
insert(xa,xb,xa1,xc1) = ext(ext(xc1,xa1,alpha,gamma),xa1,xa,xb).
-T(xa,xb,xc) | -E3(xa,xb,xc,xa1,xb1,xc1) | T(xa1,xb1,xc1).  % Satz 4.6

% following is Definition 4.10
-Col(xa,xb,xc) | T(xa,xb,xc) | T(xb,xc,xa) | T(xc,xa,xb).
Col(xa,xb,xc) | -T(xa,xb,xc).
Col(xa,xb,xc) | -T(xb,xc,xa).
Col(xa,xb,xc) | -T(xc,xa,xb).
```

```
% Following are Satz 4.11
-Col(x,y,z) | Col(y,z,x).
-Col(x,y,z) | Col(z,x,y).
-Col(x,y,z) | Col(z,y,x).
-Col(x,y,z) | Col(y,x,z).
-Col(x,y,z) | Col(x,z,y).
% following is Satz 4.12
Col(x,x,y).
% following is Satz 4.13
-Col(xa,xb,xc) | - E3(xa,xb,xc,xa1,xb1,xc1) | Col(xa1,xb1,xc1).
% following is Satz 4.14
-Col(xa,xb,xc) | -E(xa,xb,xa1,xb1) | E3(xa,xb,xc,xa1,xb1,insert5(xa,xb,xc,xa1,xb1)).
% following is Definition 4.15
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | Col(xa,xb,xc).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E3(xa,xb,xc,xa1,xb1,xc1).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xa,xd,xa1,xd1).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xd,xb1,xd1).
-Col(xa,xb,xc) | - E3(xa,xb,xc,xa1,xb1,xc1) | - E(xa,xd,xa1,xd1) |
 -E(xb,xd,xb1,xd1) | FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1).
% Following is Satz 4.16
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | xa = xb | E(xc,xd,xc1,xd1).
% Following is Satz 4.17
xa = xb | -Col(xa,xb,xc) | -E(xa,xp,xa,xq) | -E(xb,xp,xb,xq) | E(xc,xp,xc,xq).
% Following is Satz 4.18
xa = xb | -Col(xa,xb,xc) | -E(xa,xc,xa,xc1) | -E(xb,xc,xb,xc1) | xc = xc1.
% Following is Satz 4.19
-T(xa,xc,xb) | -E(xa,xc,xa,xc1) | -E(xb,xc,xb,xc1) | xc = xc1.
% Following is Satz 5.1
xa = xb | -T(xa,xb,xc) | -T(xa,xb,xd) | T(xa,xc,xd) | T(xa,xd,xc).
% Following is Satz 5.2
xa = xb | -T(xa,xb,xc) | -T(xa,xb,xd)| T(xb,xc,xd) | T(xb,xd,xc).
% Following is Satz 5.3
-T(xa,xb,xd) | -T(xa,xc,xd) | T(xa,xb,xc) | T(xa,xc,xb).
% Following is Definition 5.4
-T(xc,y,xd) | -E(xa,xb,xc,y) | le(xa,xb,xc,xd).
-le(xa,xb,xc,xd) | T(xc,insert(xa,xb,xc,xd),xd).  %ab inserted into cd
-le(xa,xb,xc,xd) | E(xa,xb,xc,insert(xa,xb,xc,xd)).
% Following is Satz 5.5a
-le(xa,xb,xc,xd) | T(xa,xb,insert(xc,xd,xa,xb)).
-le(xa,xb,xc,xd) | E(xa,insert(xc,xd,xa,xb),xc,xd).
% Following is Satz 5.5b
-T(xa,xb,x) | -E(xa,x,xc,xd) | le(xa,xb,xc,xd).
% Following is Satz 5.6
-le(xa,xb,xc,xd) | -E(xa,xb,xa1,xb1) | -E(xc,xd,xc1,xd1) | le(xa1,xb1,xc1,xd1).
% Following is Satz 5.7
le(xa,xb,xa,xb).
% Following is Satz 5.8
-le(xa,xb,xc,xd) | - le(xc,xd,xe,xf) | le(xa,xb,xe,xf).
% Following is Satz 5.9
-le(xa,xb,xc,xd) | -le(xc,xd,xa,xb) | E(xa,xb,xc,xd).
% Following is Satz 5.10
le(xa,xb,xc,xd) | le(xc,xd,xa,xb).
% Following is Satz 5.11
```

```
le(xa,xa,xc,xd).
% Following is Definition 6.1
sameside(xa,xp,xb) | xa=xp | xb = xp | -T(xp,xa,xb).
sameside(xa,xp,xb) | xa=xp | xb = xp | -T(xp,xb,xa).
-sameside(xa,xp,xb) | xa != xp.
-sameside(xa,xp,xb) | xb != xp.
-sameside(xa,xp,xb) | T(xp,xa,xb) | T(xp,xb,xa).
%Following is Satz 6.2
xa=xp | xb=xp | xc = xp | -T(xa,xp,xc) | -T(xb,xp,xc) | sameside(xa,xp,xb).
xa=xp | xb=xp | xc = xp | -T(xa,xp,xc) | T(xb,xp,xc) | -sameside(xa,xp,xb).
% following is Satz 6.3
-sameside(xa,xp,xb) | xa != xp.
-sameside(xa,xp,xb) | xb != xp.
-sameside(xa,xp,xb) | f63(xa,xp,xb) != xp.
-sameside(xa,xp,xb) | T(xa,xp,f63(xa,xp,xc)).
-sameside(xa,xp,xb) | T(xb,xp,f63(xa,xp,xc)).
xa = xp | xb = xp | xc = xp | -T(xa,xp,xc) | -T(xb,xp,xc) | sameside(xa,xp,xb).
% following is Satz 6.4
-sameside(xa,xp,xb) | Col(xa,xp,xb).
-sameside(xa,xp,xb) | -T(xa,xp,xb).
-Col(xa,xp,xb) | T(xa,xp,xb) | sameside(xa,xp,xb).
% following is Satz 6.5
xa=xp | sameside(xa,xp,xa).
% following is Definition 9.1
xp = xq | Col(xp,xq,xa) | Col(xp,xq,xb) | -T(xa,xt,xb) | -Col(xp,xq,xt) | opposite(xa,xp,xq,xb).
-opposite(xa,xp,xq,xb) | -Col(xp,xq,xa).
-opposite(xa,xp,xq,xb) | -Col(xp,xq,xb).
-opposite(xa,xp,xq,xb) | T(xa,il(xa,xb,xp,xq),xb).
-opposite(xa,xp,xq,xb) | Col(xp,xq,il(xa,xb,xp,xq)).
% following is Satz 9.2
-opposite(xa,xp,xq,xb) | opposite(xb,xp,xq,xa).
% following is Satz 9.5
-opposite(xa,xp,xq,xc) | -Col(xp,xq,xr) | -sameside(xa,xr,xb) | opposite(xb,xp,xq,xc).  % Satz 9.5
end_of_list.


list(passive).
%-opposite(c,p,q,b) | $ANS(1).
%-sameside(c,p,a) | $ANS(2).
%-opposite(a,p,q,b) | $ANS(3).
%t != q | $ANS(4).
end_of_list.

list(sos).
T(p,q,c) | -T(p,q,c).  % case 1a or Case 1b.
Col(p,c,q) | -Col(p,q,c).   % case 1 or case 2
Col(b,p,q) | -Col(b,p,q).    % case 2a or case 2b;
d = il(a,b,p,q).
t = ip(p,c,a,b,d).
T(a,c,p).
T(b,q,c).
-T(a,x,b) | -T(p,q,x).
end_of_list.
```

With the given input file, OTTER produces a proof of length 111 in less than 30 CPU-seconds. I note that, should you block the use of binary resolution, as I have just done at this very moment, you may not obtain a proof; I have not so far. However, perhaps that inference rule is in fact unnecessary, if you choose appropriate values to assign to the various parameters, if such exist. The 111-step proof relies on thirty-seven items from the input, displayed (as is the style of OTTER) before the deduced items are presented. Also, rather than UNIT CONFLICT, the proof completes with the deduction of the empty clause.

Yes, the use of this input file is indeed in the spirit of proof checking. I think it safe to say, that, without the provision of steps from Gupta's proof, none of my methods would have ever led to a proof of outer Pasch from inner Pasch. Some might then conclude that OTTER is not nearly as powerful as my notebooks strongly suggest it is. If you lean in that direction, you might, before taking a firm stand, review successes obtained with OTTER in various areas of abstract algebra and logic. In many, many cases, no guidance was provided, from a book or a paper. If you wonder why Tarskian geometry offers such often-impenetrable obstacles, I suggest (as Overbeek has) that the non-Hornness of the axioms is indeed a problem. In particular, when axioms are involved that contain more than one positive literal, the terrain is far rockier.

Of the two areas for study, proof shortening and assumption reduction, because of many years of research, I knew far more about the former than about the latter. Therefore, I turned immediately to finding a proof shorter than length 111. So typical of my studies, I decided to use otter-loop, a program (as noted) written for me by McCune to aid me in proof shortening. I planned to block each of the deduced steps of the 111-step proof to see what would occur. Blocking the last step ordinarily is fruitless in that it often is the final important step of the proof. But, since the 111-step proof required almost 30 CPU-seconds to complete, I needed to make some move that would reduce the time, not wishing to wait 111 times 30 seconds to see whether progress would occur. I turned to the inclusion of a hints list, placing all 111 deduced items in that list as hints, a move that reduced the CPU time to a few seconds to produce a proof.

I made a few modifications and additions to produce the input file I used to seek ever-shorter proofs, the following.

```
assign(max_seconds,2).
set(back_sub).
assign(max_distinct_vars,4).
assign(max_proofs,6).
assign(bsub_hint_wt,-1).
set(keep_hint_subsumers).
weight(junk,1000).

list(demodulators).
end_of_list).
```

The intention is to demodulate each step of the 111-step proof, one at a time, to junk, which will, because of the assignment of 1000 to junk, cause the item, when deduced, to be discarded in that the max_weight is assigned a value far smaller than 1000. The demodulator list is included to so-to-speak catch new demodulators, if and when I decided to adjoin such. With the use of ancestor subsumption, you should always include set(back_sub). Because the move of relying on hints sharply reduced the time to complete a proof, I assigned the small value 2 to max_seconds. I assigned the value 6 to max_proofs in that, when OTTER finds more than one proof of a given theorem, the proof lengths can vary. Further, you see two items focusing on hints, both of which had best be included when using hints; one or both can easily be forgotten.

How convenient to use otter-loop, for which I will always be indebted to McCune. One single instruction and, in this case, 111 jobs are run, one after the other. When one of the runs showed a reduction in proof length, I adjoined the corresponding demodulator, to block the indicated proof step. After iterating, I found the following two demodulators whose use enabled the program to present to me a 90-step proof deducing outer Pasch from inner Pasch.

```
EQ(Col(d,q,p),junk).
EQ(Col(c,b,q),junk).
```

The fourth proof, the following, of the six found with the amended file, has proof length 90, while the other five have proof lengths that vary from 91 to 93.

A 90-Step Proof of Inner Implies Outer Pasch

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Sun Mar 31 08:30:39 2013
The command was "otter". The process ID is 28254.
-----> EMPTY CLAUSE at 1.09 sec ----> 6392 [binary,6327.1,95.2,unit_del,3348,6350] $F.

Length of proof is 90. Level of proof is 19.

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

8 [] -T(xa,xp,xc)| -T(xb,xq,xc)|T(xp,ip(xa,xp,xc,xb,xq),xb).
9 [] -T(xa,xp,xc)| -T(xb,xq,xc)|T(xq,ip(xa,xp,xc,xb,xq),xa).
21 [] T(x,y,y).
22 [] -T(xa,xb,xc)|T(xc,xb,xa).
23 [] T(xa,xa,xb).
24 [] -T(xa,xb,xc)| -T(xb,xa,xc)|xa=xb.
25 [] -T(xa,xb,xd)| -T(xb,xc,xd)|T(xa,xb,xc).
26 [] -T(xa,xb,xd)| -T(xb,xc,xd)|T(xa,xc,xd).
27 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xb,xc,xd).
30 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xa,xb,xd).
32 [] -T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xb,xd).
55 [] -Col(xa,xb,xc)|T(xa,xb,xc)|T(xb,xc,xa)|T(xc,xa,xb).
57 [] Col(xa,xb,xc)| -T(xb,xc,xa).
58 [] Col(xa,xb,xc)| -T(xc,xa,xb).
62 [] -Col(x,y,z)|Col(y,x,z).
63 [] -Col(x,y,z)|Col(x,z,y).
64 [] Col(x,x,y).
78 [] -T(xa,xb,xd)| -T(xa,xc,xd)|T(xa,xb,xc)|T(xa,xc,xb).
95 [] -sameside(xa,xp,xb)|T(xp,xa,xb)|T(xp,xb,xa).
98 [] -sameside(xa,xp,xb)|xa!=xp.
99 [] -sameside(xa,xp,xb)|xb!=xp.
104 [] -sameside(xa,xp,xb)|Col(xa,xp,xb).
106 [] -Col(xa,xp,xb)|T(xa,xp,xb)|sameside(xa,xp,xb).
108 [] xp=xq|Col(xp,xq,xa)|Col(xp,xq,xb)| -T(xa,xt,xb)| -Col(xp,xq,xt)|opposite(xa,xp,xq,xb).
109 [] -opposite(xa,xp,xq,xb)| -Col(xp,xq,xa).
111 [] -opposite(xa,xp,xq,xb)|T(xa,il(xa,xb,xp,xq),xb).
112 [] -opposite(xa,xp,xq,xb)|Col(xp,xq,il(xa,xb,xp,xq)).
114 [] -opposite(xa,xp,xq,xc)| -Col(xp,xq,xr)| -sameside(xa,xr,xb)|opposite(xb,xp,xq,xc).
118 [] d=il(a,b,p,q).
119 [] t=ip(p,c,a,b,d).
120 [] T(a,c,p).
121 [] T(b,q,c).
122 [] -T(a,x,b)| -T(p,q,x).
247 [para_from,118.1.2,112.2.3] -opposite(a,p,q,b)|Col(p,q,d).
248 [para_from,118.1.2,111.2.2] -opposite(a,p,q,b)|T(a,d,b).
253 [para_from,119.1.2,9.3.2] -T(p,c,a)| -T(b,d,a)|T(d,t,p).
261 [binary,120.1,58.2] Col(c,p,a).
279 [binary,120.1,22.1] T(p,c,a).

344 [binary,121.1,22.1] T(c,q,b).
393 [binary,122.1,23.1] -T(p,q,a).
395 [binary,122.1,21.1] -T(p,q,b).
407 [binary,122.2,21.1] -T(a,q,b).
422 [binary,248.2,122.1] -opposite(a,p,q,b)| -T(p,q,d).
449 [binary,248.2,22.1] -opposite(a,p,q,b)|T(b,d,a).
485 [binary,253.2,22.2,unit_del,279] T(d,t,p)| -T(a,d,b).
503 [binary,261.1,106.1] T(c,p,a)|sameside(c,p,a).
579 [binary,344.1,57.2] Col(b,c,q).
658 [binary,393.1,22.2] -T(a,q,p).
668 [ur,30,279,393] -T(p,q,c).
669 [ur,30,23,393] -T(p,q,p).
672 [ur,27,23,393] -T(q,p,q).
673 [ur,26,279,393] -T(c,q,a).
694 [binary,395.1,22.2] -T(b,q,p).
706 [ur,27,344,395] -T(c,p,q).
728 [binary,407.1,22.2] -T(b,q,a).
740 [ur,27,344,407] -T(c,a,q).
747 [binary,422.2,106.2] -opposite(a,p,q,b)| -Col(p,q,d)|sameside(p,q,d).
761 [binary,422.2,22.2] -opposite(a,p,q,b)| -T(d,q,p).
798 [binary,449.2,9.2] -opposite(a,p,q,b)| -T(x,y,a)|T(d,ip(x,y,a,b,d),x).
800 [binary,449.2,8.2] -opposite(a,p,q,b)| -T(x,y,a)|T(y,ip(x,y,a,b,d),b).
832 [binary,485.1,22.1] -T(a,d,b)|T(p,t,d).
854 [binary,832.1,248.2] T(p,t,d)| -opposite(a,p,q,b).
957 [binary,503.1,24.2,unit_del,279] sameside(c,p,a)|p=c.
980 [binary,579.1,106.1] T(b,c,q)|sameside(b,c,q).
1019 [ur,26,120,658] -T(c,q,p).
1049 [ur,27,121,668] -T(b,p,q).
1050 [ur,26,121,668] -T(p,b,c).
1051 [ur,26,21,668] -T(c,q,c).
1057 [binary,669.1,106.2] -Col(p,q,p)|sameside(p,q,p).
1080 [binary,672.1,106.2] -Col(q,p,q)|sameside(q,p,q).
1125 [ur,30,344,673] -T(c,b,a).
1158 [ur,30,121,694] -T(b,c,p).
1248 [ur,30,121,728] -T(b,c,a).
1269 [ur,78,344,673,740] -T(c,a,b).
1292 [binary,747.2,247.2,factor_simp] -opposite(a,p,q,b)|sameside(p,q,d).
1338 [para_into,800.3.2,119.1.2,unit_del,279] -opposite(a,p,q,b)|T(c,t,b).
1387 [para_from,957.2.1,694.1.3,unit_del,121] sameside(c,p,a).
1404 [binary,980.1,22.1] sameside(b,c,q)|T(q,c,b).
1427 [ur,78,344,706,1019] -T(c,p,b).
1489 [binary,1050.1,55.4,unit_del,1158,1427] -Col(b,c,p).
1536 [ur,27,121,1051] -T(b,c,q).
1539 [ur,25,344,1051] -T(q,c,b).
1543 [binary,1057.1,63.2,unit_del,64] sameside(p,q,p).
1548 [binary,1080.1,63.2,unit_del,64] sameside(q,p,q).
1554 [binary,1125.1,95.2,unit_del,1269] -sameside(b,c,a).
1611 [ur,27,23,1158] -T(c,b,c).
1639 [binary,1248.1,106.2,unit_del,1554] -Col(b,c,a).
1722 [binary,1404.1,99.1,unit_del,1539] q!=c.
1814 [binary,1536.1,106.2,unit_del,579] sameside(b,c,q).
1838 [binary,1543.1,104.1] Col(p,q,p).
1841 [binary,1543.1,99.1] p!=q.

1870 [binary,1548.1,104.1] Col(q,p,q).
1893 [ur,32,121,1722,694] -T(q,c,p).
1934 [binary,1814.1,98.1] b!=c.
1998 [binary,1870.1,62.1] Col(p,q,q).
2010 [binary,1893.1,55.3,unit_del,668,706] -Col(p,q,c).
2114 [hyper,108,344,1998,unit_del,1841,2010] Col(p,q,b)|opposite(c,p,q,b).
2587 [binary,1338.2,57.2] -opposite(a,p,q,b)|Col(b,c,t).
2809 [binary,2114.1,55.1,unit_del,395,1049] opposite(c,p,q,b)|T(q,b,p).
3270 [hyper,32,344,2809,unit_del,1019] opposite(c,p,q,b)|q=b.
3306 [para_from,3270.2.1,694.1.2,unit_del,23] opposite(c,p,q,b).
3311 [hyper,114,3306,1838,1387] opposite(a,p,q,b).
3337 [binary,3311.1,2587.1] Col(b,c,t).
3348 [binary,3311.1,1292.1] sameside(p,q,d).
3353 [binary,3311.1,854.2] T(p,t,d).
3356 [binary,3311.1,798.1] -T(x,y,a)|T(d,ip(x,y,a,b,d),x).
3361 [binary,3311.1,761.1] -T(d,q,p).
3411 [binary,3337.1,109.2] -opposite(t,b,c,x).
3735 [para_into,3356.2.2,119.1.2,unit_del,279] T(d,t,p).
4114 [ur,30,3735,3361] -T(d,q,t).
5912 [binary,1611.1,106.2] -Col(c,b,c)|sameside(c,b,c).
6014 [binary,5912.1,63.2,unit_del,64] sameside(c,b,c).
6018 [binary,6014.1,104.1] Col(c,b,c).
6026 [binary,6018.1,62.1] Col(b,c,c).
6039 [hyper,108,279,6026,unit_del,1934,1489,1639] opposite(p,b,c,a).
6133 [hyper,114,6039,579,3348] opposite(d,b,c,a).
6141 [ur,114,6133,579,3411] -sameside(d,q,t).
6143 [binary,6141.1,106.3,unit_del,4114] -Col(d,q,t).
6154 [binary,6143.1,58.1] -T(t,d,q).
6155 [binary,6143.1,57.1] -T(q,t,d).
6280 [ur,27,3353,6154] -T(p,d,q).
6327 [ur,26,3353,6155] -T(q,p,d).
6350 [binary,6280.1,22.2] -T(q,d,p).
6392 [binary,6327.1,95.2,unit_del,3348,6350] $F.

I was not able to find a shorter proof, although I would bet a shorter proof exists.

I then turned to assumption reduction, whose goal is to reduce the number of items, taken from the input, that are used to still find a proof that inner Pasch implies outer. With fewer assumptions, axioms, and proved theorems, I would expect that the program would present to me, when a proof was found, one or more of greater length than 90. However, in contrast to proof shortening where I could run with a few strokes many jobs in sequence, for assumption reduction I had to run one job at a time. Each job would consist of my choosing one of the items used to obtain the inner-outer proof and comment it out. My method for making such choices was simply guessing. If a proof was obtained, whether a bit longer or so, then I would take the amended input file and choose another item to comment out. The plan was indeed simple; the exercising of it was totally up to me, in contrast to what might be termed the program telling me what to do in proof shortening by citing proof steps that, if demodulated, produced a shorter proof. You might try your hand at reducing the number of items used, thirty-seven, and experience a small amount of research and, perhaps, a large amount of excitement as you find that various axioms and proved theorems can indeed be omitted and still yield a proof. If that is your choice, perhaps you might pause here before seeing what happened in my sojourn.

With the knowledge that the 111-step proof relied on thirty-seven input axioms and proved theorems, I felt certain I could reduce that number. With this iterative approach, I eventually had a proof, from OTTER, in which but twenty-eight items were sufficient. Perhaps many guesses among the thirty-seven would have sufficed; I do not know. The following nine were omitted, and still a proof was found.

-T(xa,xb,xc) | -T(xb,xa,xc) | xa = xb.
-Col(x,y,z) | Col(y,z,x).
-Col(x,y,z) | Col(z,y,x).
-Col(x,y,z) | Col(x,z,y).
Col(x,x,y).
-T(xa,xb,xd) | -T(xa,xc,xd) | T(xa,xb,xc) | T(xa,xc,xb).
-sameside(xa,xp,xb) | T(xp,xa,xb) | T(xp,xb,xa).
xa = xp | sameside(xa,xp,xa).
Col(b,p,q) | -Col(b,p,q).

A glance at the 90-step proof shows that it relies on thirty-three axioms and previously proved theorems. After many additional experiments, I learned that Beeson had a proof that inner implies outer that depends on the same twenty-eight.

Among those experiments I found a proof that depends on 28 but differs by the following six from the proof that agreed with Beeson's proof.

-T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xc,xd).
Col(xa,xb,xc)| -T(xa,xb,xc).
-Col(x,y,z)|Col(z,x,y).
xa=xp|xb=xp|xc=xp| -T(xa,xp,xc)|T(xb,xp,xc)| -sameside(xa,xp,xb).
Col(p,c,q)| -Col(p,q,c).
-Col(p,q,c)|Col(q,p,c).

The proof space is clearly quite rich, as you see from the fact that two different sets of twenty-eight axioms and proved theorems, differing by six items, still lead to a proof. The proof lengths that were offered me during this sojourn usually ranged from 101 to 108; however, one experiment led to a 246-step proof after quite a long time, a proof, if memory serves that depends on at least thirty-four items. Eventually, I found an input file that yielded a proof that in turn depended on but twenty-six axioms and previously proved theorems. Perhaps because of a type of redundancy, rather than being different from the thirty-seven by eleven, the file that yielded the proof relying on but twenty-six differed by avoiding the use of the following fourteen.

-T(xa,xb,xc) | -T(xb,xa,xc) | xa = xb.
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xb,xc).
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd).
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd).
Col(xa,xb,xc) | -T(xb,xc,xa).
-Col(x,y,z) | Col(y,z,x).
-Col(x,y,z) | Col(z,y,x).
-Col(x,y,z) | Col(y,x,z).
-Col(x,y,z) | Col(x,z,y).
Col(x,x,y).
-T(xa,xb,xd) | -T(xa,xc,xd) | T(xa,xb,xc) | T(xa,xc,xb).
-sameside(xa,xp,xb) | T(xp,xa,xb) | T(xp,xb,xa).
xa=xp | sameside(xa,xp,xa).
Col(b,p,q) | -Col(b,p,q).

To my surprise, I discovered along the way that in addition to dispensing with previously proved theorems, a proof can be found by omitting clauses that are part of various definitions. For example, if you avoid including in the input the first four of the following seven that were relied upon in a proof using twenty-eight items before making deductions, you can replace the four by the last three of the following seven to get a proof that relies on twenty-seven items.

-T(xa,xb,xc)| -T(xa,xc,xd)|T(xb,xc,xd).
-T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xb,xd).
Col(xa,xb,xc)| -T(xb,xc,xa).
-Col(x,y,z)|Col(y,x,z).

-T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xc,xd).
Col(xa,xb,xc)| -T(xa,xb,xc).
-Col(x,y,z)|Col(z,x,y).

I was also pleased to find that the clause for transitivity of betweenness can be dispensed with, and I enjoyed the discovery of what I think of as redundancy. For example, I have two proofs of inner implies outer, each relying on 26 items, which differ by the exchange of the first (of the following) for the second and conversely.

-sameside(xa,xp,xb)|xa!=xp.
-sameside(xa,xp,xb)| -T(xa,xp,xb).

I have not made a more detailed examination to see which, if any, of the axioms was unneeded. However, the obtaining of a long, long proof when some item is dispensed with does indeed suggest how important that axiom, part of a definition, or previously proved theorem is.

In the next section, with invaluable guidance and suggestions from Beeson, in part based on Szmielew and other sources (that include Coxeter's book titled *Introduction to Geometry*), I can turn to two proofs of great interest. During this part of the trip, you will see how tautologies can be used effectively, and, again, how a diagram proves crucial. I return, in the following section, to the 1959 Tarski axiom set, that given in Section 1, but using a notation consistent with Szmielew. In the first theorem, the goal is to prove connectivity of betweenness dependent on the other axioms of the set I gave first in this notebook. In the second theorem, the goal is to prove inner Pasch from that set, which, as you can see, relies on two clauses for outer Pasch.

10. Two Promised Theorems

In this section, you will learn, thanks to Beeson, how demodulation can be used directly to find a proof. Ordinarily, demodulation is used for canonicalization and simplification. Of course, you have been treated to its use in the context of proof shortening as well. I was surprised at his use of demodulation and its substantial value in proof finding, which is discussed when the theorem that proves inner Pasch from outer Pasch takes center stage.

Here you will see how the hot list usage made the difference, the difference between completing a proof and obtaining no proof at all. As you will learn, I turned to using the hot list strategy in place of using demodulation. Tautologies come into play in this section. Why their presence enables OTTER to find a proof, in contrast to their absence that leads to no proof, I cannot say; I have not analyzed the situation. You will also share my surprise when I found that level saturation carried the day, whereas its replacement with complexity-preference (ratio) did not. The approaches taken here, to prove that inner Pasch is implied by outer Pasch (with the use of the 1959 Tarski axioms given in Section 1), and that, in the presence of those axioms, the axiom of connectivity of betweenness is dependent—although not proved with OTTER, I regret to say—may indeed provide you with additional research ideas.

At one point in the research reported in this notebook, some doubt was cast regarding the dependence of connectivity of betweenness on the remaining axioms offered in Section 1. Nevertheless, based on nothing that occurs to me, I still held out hope that such dependence was a fact. Eventually, at my request, Beeson sent me an input file that provided me with a beginning for my attack on this promised theorem, although its use, as Beeson certainly knew, did not come close to proving what I wished to prove. Its hints2 list contained 655 items, including proof steps from a proof, perhaps found jointly by Beeson and me, of connectivity based on the use of inner Pasch; as noted, I was after a proof in which outer Pasch was used. Fortunately, as you will learn, I concentrated my efforts for quite a while on the theorem that deduces a proof of inner Pasch from outer Pasch in the presence of the 1959 Tarski axiom set. If I hadn't, at least for now in early May 2013, I would be stuck without a proof of the dependence of connectivity.

Beeson, in his usual kindness, sent me an input file to begin my study of proving inner Pasch from outer Pasch; and, if memory serves and moderate scrutiny seems to support, the file does include the axiom of connectivity of betweenness, an axiom that I later avoided using. Of course, he was certain that much was still needed. For but one aspect, the file was oriented toward a case analysis, focusing on a != c and, then, on a = c. Because I ordinarily accept input files from colleagues without reading them fully, I note, so important, that I was unaware that Beeson (in effect) suggested using demodulators to sharply aid OTTER in finding proofs. Clearly, as I learned so many, many experiments later, he had effectively employed demodulators based on an appropriate diagram to make substantial progress. But, being unaware of the role of demodulation, I turned to what I believe, if memory serves, was an assignment from Beeson.

He wished to find a proof that avoided the approach based on case analysis, in particular, based on the two cited cases, a != c and a = c. He had each of the two proofs, based on this case analysis, already. I conducted simultaneously two experiments, one based on the ratio strategy and one based on level saturation; in both, I commented out the cited cases and replaced them with the tautology a != c | a = c. The expectation is that level saturation will take a long time when compared with the use of ratio; after all, with level saturation, no retained clause is skipped because of its weight. Indeed, to initiate inference-rule applications, level saturation has the program consider the items in the order they are retained. By its nature, you thus see that the use of sos(queue), level saturation, has at the same time possible advantages and possible disadvantages.

What occurred was quite unexpected. Specifically, the level-saturation approach found the desired proof that outer Pasch implies inner Pasch in the presence of the 1959 axiom set. With ratio, the program eventually stopped, noting that the sos went empty, and with no proof found. Now how could that occur, you might ask. Well, subsumption may be the key.

In the following trivial example, you see that paramodulation does not behave as, say, binary resolution does in the context of generalizing a parent.

```
a = b.
Q(f(a)).
Q(F(b)).
Q(x).
```

With paramodulation, from the first two given items, you deduce the third. However, if the fourth item is substituted for the second and paramodulation is applied, the conclusion is *not* more general than that yielded from the first two items. Therefore, and I am simply conjecturing, OTTER may have deduced a more general item with the ratio strategy, subsumed a less general item, and been prevented from making a needed deduction.

The following 60-step proof was obtained.

A 60-Step Proof of Inner Pasch from Outer Pasch

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Fri Apr 19 14:00:14 2013
The command was "otter". The process ID is 10322.
----> UNIT CONFLICT at 835.70 sec ----> 19943 [binary,19942.1,19230.1] $F.

Length of proof is 60. Level of proof is 23.

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

2 [] T(x,y,ext(x,y,w,v)).
3 [] E(y,ext(x,y,w,v),w,v).
10 [] -E(xa,xb,xc,xd)|E(xc,xd,xa,xb).
11 [] -E(xa,xb,xc,xd)|E(xb,xa,xc,xd).
15 [] -E(u,v,x,x)|u=v.
16 [] T(x,y,y).
17 [] -T(xa,xb,xc)|T(xc,xb,xa).
18 [] T(xa,xa,xb).
```

21 [] -T(xa,xb,xd)| -T(xb,xc,xd)|T(xa,xc,xd).
22 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xb,xc,xd).
23 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xa,xb,xd).
25 [] -T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xb,xd).
26 [] xa=xb| -T(xa,xb,xc)| -T(xa,xb,xd)|T(xa,xc,xd)|T(xa,xd,xc).
27 [] xa=xb| -T(xa,xb,xc)| -T(xa,xb,xd)|T(xb,xc,xd)|T(xb,xd,xc).
28 [] -T(xa,xb,xd)| -T(xa,xc,xd)|T(xa,xb,xc)|T(xa,xc,xb).
30 [] op(q,b,p,d,c)=j.
31 [] op(j,b,a,d,p)=e.
32 [] op(j,q,a,b,e)=f.
33 [] T(a,p,c).
34 [] T(b,q,c).
35 [] -T(p,x,b)| -T(q,x,a).
36 [] d=ext(a,c,a,c).
40 [] g=op(p,b,q,a,f).
41 [] -T(x,v,u)| -T(y,u,z)|T(x,op(v,x,y,z,u),y).
42 [] -T(x,v,u)| -T(y,u,z)|T(z,v,op(v,x,y,z,u)).
43 [] c!=d|c=d.
44 [] a!=c|a=c.
650 [] -T(xa,xb,xc)|T(xc,xb,xa).
654 [binary,33.1,17.1] T(c,p,a).
657 [hyper,23,33,2] T(a,p,ext(a,c,x,y)).
658 [hyper,22,33,2] T(p,c,ext(a,c,x,y)).
679 [binary,34.1,17.1] T(c,q,b).
704 [binary,35.1,18.1] -T(q,p,a).
705 [binary,35.2,18.1] -T(p,q,b).
706 [binary,35.2,16.1] -T(p,a,b).
708 (heat=1) [binary,706.1,650.2] -T(b,a,p).
709 [para_from,36.1.2,3.1.2] E(c,d,a,c).
710 [para_from,36.1.2,2.1.3] T(a,c,d).
750 [para_into,657.1.3,36.1.2] T(a,p,d).
752 [para_into,658.1.3,36.1.2] T(p,c,d).
753 (heat=1) [binary,752.1,650.1] T(d,c,p).
862 [ur,22,654,704] -T(c,q,p).
863 [ur,21,654,704] -T(q,c,a).
865 (heat=1) [binary,863.1,650.2] -T(a,c,q).
866 [ur,22,679,705] -T(c,p,q).
868 [binary,709.1,11.1] E(d,c,a,c).
1502 [hyper,42,34,752,demod,30] T(d,q,j).
1508 [hyper,41,34,752,demod,30] T(b,j,p).
1576 (heat=1) [binary,1502.1,650.1] T(j,q,d).
1582 (heat=1) [binary,1508.1,650.1] T(p,j,b).
1838 [para_into,863.1.2,44.2.2,unit_del,16] a!=c.
1846 [ur,28,654,862,866] -T(c,q,a).
1853 (heat=1) [binary,1846.1,650.2] -T(a,q,c).
1860 [binary,868.1,10.1] E(a,c,d,c).
2892 [hyper,42,1508,750,demod,31] T(d,j,e).
2901 [hyper,41,1508,750,demod,31] T(b,e,a).
2995 (heat=1) [binary,2901.1,650.1] T(a,e,b).
3076 [ur,21,1508,708] -T(j,a,p).
3568 [binary,1838.1,15.2] -E(a,c,x,x).
3750 [para_into,1860.1.3,43.2.2,unit_del,3568] c!=d.
5222 [hyper,22,1502,2892] T(q,j,e).

```
7188 [ur,25,710,3750,865] -T(c,d,q).
7215 (heat=1) [binary,7188.1,650.2] -T(q,d,c).
8420 [hyper,42,5222,2995,demod,32] T(b,j,f).
8427 [hyper,41,5222,2995,demod,32] T(q,f,a).
10326 [hyper,26,1508,8420] b=j|T(b,p,f)|T(b,f,p).
10669 [binary,8427.1,35.2] -T(p,f,b).
10670 (heat=1) [binary,10669.1,650.2] -T(b,f,p).
12824 [binary,10326.2,17.1,unit_del,10670] b=j|T(f,p,b).
13766 [para_from,12824.1.1,34.1.1] T(j,q,c)|T(f,p,b).
14362 [hyper,27,1576,13766,unit_del,7215] T(f,p,b)|j=q|T(q,c,d).
14624 [para_from,14362.2.1,1582.1.2,unit_del,705] T(f,p,b)|T(q,c,d).
14626 (heat=1) [binary,14624.2,650.1] T(f,p,b)|T(d,c,q).
14753 [hyper,27,753,14626,unit_del,866,862] T(f,p,b)|d=c.
14835 [para_from,14753.2.1,7215.1.2,unit_del,16] T(f,p,b).
14836 (heat=1) [binary,14835.1,650.1] T(b,p,f).
15472 [hyper,42,14836,8427] T(a,p,op(p,b,q,a,f)).
15480 [hyper,41,14836,8427] T(b,op(p,b,q,a,f),q).
16129 [para_into,15472.1.3,40.1.2] T(a,p,g).
16190 [hyper,22,15480,34] T(op(p,b,q,a,f),q,c).
16648 [hyper,26,33,16129] a=p|T(a,c,g)|T(a,g,c).
17230 [para_into,16190.1.1,40.1.2] T(g,q,c).
18265 [para_from,16648.1.1,3076.1.2,unit_del,16] T(a,c,g)|T(a,g,c).
19229 [ur,22,17230,863] -T(g,c,a).
19230 [ur,21,17230,1853] -T(a,g,c).
19231 (heat=1) [binary,19229.1,650.2] -T(a,c,g).
19941 [binary,18265.2,17.1,unit_del,19231] T(c,g,a).
19942 (heat=1) [binary,19941.1,650.1] T(a,g,c).
19943 [binary,19942.1,19230.1] $F.
```

I now give the initial segment of an input file that, with appropriate hints, will yield the just-given 60-step proof.

Initial Segment of an Input File for Proving Inner Pasch

```
% Tarski's 1959 axiom system.
% That consists of axioms A1-A6, outer pasch instead of A7, A8-A11,
% transitivity of betweenness (Satz 3.5, aka A15), and
% connectivity of betweeenness (Satz 5.1, aka A18).
% In this file we try to prove inner Pasch from outer Pasch.

set(hyper_res).
set(para_into).
set(para_from).
set(ur_res).
set(binary_res).
set(unit_deletion).
set(order_history).
assign(report,5400).
% assign(max_seconds, 60000).
assign(max_mem,840000).
clear(print_kept).
set(input_sos_first).
set(sos_queue).
```

```
set(back_sub).
assign(bsub_hint_wt,-1).
set(keep_hint_subsumers).
assign(max_weight,11).
assign(max_distinct_vars,4).
%  assign(pick_given_ratio,2).
assign(max_proofs,4).
assign(neg_weight,8).

list(usable).
x=x.
T(x,y,ext(x,y,w,v)).                  % A4, first half
E(y,ext(x,y,w,v),w,v).                  % A4, second half
E(x,x,y,y).  % Satz 2.8
-T(x,y,x) | x=y.                    % A6
% following two clauses are outer Pasch.
-T(x,v,u) | -T(y,u,z) | T(x,op(v,x,y,z,u),y).
-T(x,v,u) | -T(y,u,z) | T(z,v,op(v,x,y,z,u)).
% following is the transitivity of betweenness, Satz 3.5, Axiom A15
-T(x,y,u) | -T(y,z,u) | T(x,y,z).
E(x,y,x,y).  % Satz 2.1
-E(xa,xb,xc,xd) | E(xc,xd,xa,xb).  % Satz 2.2
-E(xa,xb,xc,xd) | E(xb,xa,xc,xd).  % Satz 2.4
-E(xa,xb,xc,xd) | -E(xc,xd,xe,xf) | E(xa,xb,xe,xf).  % Satz 2.3
-E(xa,xb,xc,xd) | E(xa,xb,xd,xc).  % Satz 2.5
E(x,x,y,y).  % Satz 2.8
-E(u,v,x,x) | u=v.   % Not one of Szmielew's theorems but we proved it.
T(x,y,y).    % Satz 3.1
-T(xa,xb,xc) | T(xc,xb,xa).  % Satz 3.2.
T(xa,xa,xb).  % Satz 3.3
-T(xa,xb,xc) | -T(xb,xa,xc) | xa = xb.  % Satz 3.4.
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xb,xc).  % Satz 3.51.
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xc,xd).  % Satz 3.52.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd).  % Satz 3.61.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd). % Satz 3.62.
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd).  % Satz 3.71
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd).  % Satz 3.72
% Following is Satz 5.1
xa = xb | -T(xa,xb,xc) | -T(xa,xb,xd) | T(xa,xc,xd) | T(xa,xd,xc).
% Following is Satz 5.2
xa = xb | -T(xa,xb,xc) | -T(xa,xb,xd)| T(xb,xc,xd) | T(xb,xd,xc).
% Following is Satz 5.3
-T(xa,xb,xd) | -T(xa,xc,xd) | T(xa,xb,xc) | T(xa,xc,xb).
end_of_list.
list(passive).
end_of_list.
list(demodulators).
ext(a,c,a,c)=d.
op(q,b,p,d,c)=j.
op(j,b,a,d,p)=e.
op(j,q,a,b,e)=f.
end_of_list.
```

```
list(sos). % negated form of inner Pasch
T(a,p,c).
T(b,q,c).
-T(p,x,b) | -T(q,x,a).
% Now we define the diagram for the proof
d = ext(a,c,a,c).
j = op(q,b,p,d,c).
e = op(j,b,a,d,p).
f = op(j,q,a,b,e).
g = op(p,b,q,a,f).
% following two clauses are outer Pasch.
-T(x,v,u) | -T(y,u,z) | T(x,op(v,x,y,z,u),y).
-T(x,v,u) | -T(y,u,z) | T(z,v,op(v,x,y,z,u)).
c!=d | c = d.  % works with this tautology present, but not without!
a != c | a = c.
%  b=j | b != j.  % works with either case, but not with the disjunction.
end_of_list.
```

I have omitted the large set of hints that were used because the key aspect at this point is the list of axioms and theorems that were used. Some of the theorems that are listed and that were used to obtain the given 60-step proof were not proved by me. I leave that for you as a challenge. As you will shortly see, I turned to experiments after finding the 60-step proof, designed to see how many of the items (from the input) that were used in the proof that I could do without, experiments reminiscent of some discussed earlier in this notebook. As for crucial differences between the given initial segment and its correspondent in the input file Beeson sent to me, I included the use of unit_deletion and the use of level saturation with set(sos_queue).

A few lines earlier, I noted that I was fortunate to begin with an emphasis on proving inner Pasch from outer. Indeed, that choice naturally led me (as just noted) to a study similar to a study reported earlier in this notebook, namely, the experimentation with removing various items chosen from the input and used in a proof. You might think of this activity as a cousin to proof shortening. Among the experiments that I conducted, one of them proved, much, much later, to be fortuitous and indeed extremely relevant to the study of the possible dependence of connectivity. However, history and science each and both require that I report that the relevance of an early experiment to the connectivity theorem did not in any way occur to me until I was seeking a so-called direct proof—and finding the venture difficult and hazardous. As I describe my blocking of one after another of input items, you, in contrast to me, may immediately see the relevance to the dependence of connectivity.

From among the possible choices to block, to avoid using, from those theorems in the input file, I chose Satz 5.3, the following.

```
% -T(xa,xb,xd) | -T(xa,xc,xd) | T(xa,xb,xc) | T(xa,xc,xb).  %  Satz 5.3
```

(By the way, for the person who enjoys following the geometry, 5.3 asserts the totally believable, namely, if points b and c are each between points a and d, then either b is between a and c or c is between a and b.) You could naturally guess that such a statement is not needed for proving this important and powerful theorem, that outer Pasch implies inner Pasch. In approximately the same CPU time, around 2800 CPU-seconds, a second 60-step proof was completed. When compared with the 60-step proof you viewed a bit ago, both rely on twenty-eight items from the input file, which I did not expect, believing that the avoidance of 5.3 would lead to the use of but twenty-seven. In place of relying on 5.3, this second 60-step proof relied on 3.51, the following.

```
-T(xa,xb,xd)| -T(xb,xc,xd)|T(xa,xb,xc).  %  Satz 3.51
```

Again, for the person who enjoys pictures, you can draw a line a,b,d, then insert the point c between b and d, and see how reasonable is the fact that b is between a and c. The following two deductions were present in this second 60-step proof but not in the given 60-step proof.

-T(a,p,q).
-T(a,g,q).

I include various comparisons and statistics because they might lead a researcher to a new strategy or a new approach.

Have you been eager to receive another challenge, especially a challenge I cannot as yet meet? Well, I have such for you now. In particular, I took the amended input file, amended by commenting out 5.3, and made a third input file, a file in which I commented out Satz 5.2, the following.

% xa = xb | -T(xa,xb,xc) | -T(xa,xb,xd)| T(xb,xc,xd) | T(xb,xd,xc) % Satz 5.2

An appealing claim this is, namely, if a is not equal to b, and if the point b is between a and c and also between a and d, then either c is between b and d or d is between b and c. With this second blocking, and various experiments—including some conducted weeks after my first one—I have failed to find a proof in which 5.2 is absent. If you find such a proof, an email to me would indeed be of interest. This challenge seems formidable and, just perhaps, cannot be met. (Yes, I did not, as I often do, switch to a new set of hints or resonators; if I had, the new set would be based on the preceding proof or proofs, which might cause OTTER to miss the desired proof that avoids the use of 5.2.)

So, to cope with this annoying failure and, more important, to continue on my quest for finding a proof of inner from outer in which fewer, perhaps far fewer, than twenty-eight items from among axioms, theorems, and definitions were needed, I reinstated the possible use of 5.2 and, instead, commented out 5.1, the following.

% xa = xb | -T(xa,xb,xc) | -T(xa,xb,xd) | T(xa,xc,xd) | T(xa,xd,xc) % Satz 5.1

To me, 5.1 and 5.2 appear to be similar. Do you find them quite alike? More intriguing, especially if you could conjecture what will be presented late in this notebook, does Satz 5.1 remind you of an important property? At this juncture in my journey toward the city of inner Pasch, I was not reminded of any property. Perhaps I was too engrossed in my quest for a proof relying on fewer, maybe far fewer, than twenty-eight items.

OTTER did find a proof, one of length sixty-three, relying on twenty-seven input items—progress—omitting the use of 5.3 and 5.1, and adding the use of 3.51. The 63-step proof contains seventeen deduced items not in the first 60-step proof, and 15 not in the second. I was not surprised at finding a somewhat longer proof, although commenting out items does not guarantee the finding of longer proofs. (To get some idea of my thinking, remember that at this point I still did not understand how valuable this 63-step proof was and would prove to be; I simply continued my assault on assumption reduction.)

The story continued to unfold, one experiment after another, as I blithely experienced satisfaction, indeed unaware of the significance of having found a proof in which Satz 5.1 was not relied upon. The thought of eventually proving connectivity of betweenness dependent essentially drifted into the background. For example, when I commented out (blocked the use of) the following two, 3.62 and 3.72, OTTER produced a pleasing result.

% -T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd). % Satz 3.62.
% -T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd). % Satz 3.72

A 65-step proof was found, relying on twenty-five, compared with twenty-eight, items from the input file. Not used were 5.3, 5.1, 3.62, and 3.72, the latter two of which I had proved many weeks ago (from the 1959 axiom system). But, rather than introducing the use of 3.51 as was the case earlier, 3.71, the following, was relied upon.

-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd). % Satz 3.71

I have not attempted to analyze the nature of these various tradeoffs, but they are, to me, entertaining. The 65-step proof that was presented to me contained eight deduced items not in the 63-step proof and contained twenty-two not present in the first (given) 60-step proof.

As my experiments proceeded, I was making real progress, still getting proofs of inner from outer but with fewer and fewer items from the input. It occurred to me, however, that the use of demodulation interferes with an accurate determination of proof length. Therefore, the proof lengths I had been quoting were a bit misleading. For example, an examination of the earlier-given 60-step proof shows that demodulation was used six times, the demodulators corresponding to the diagram that Beeson supplied to permit OTTER to find that proof. If you count the occurrences of demodulation, noting that at no time in the 60-step proof did a step rely on more than one demodulator, you would cite the proof as being one of length 66. Because I prefer that the program report to me the more accurate proof length, and because I was curious about what would occur if I did not rely on demodulation, I eventually turned to the use of the hot list strategy. And here is a move, or set of moves, that you might find not only interesting but profitable to make in various contexts in diverse studies. Specifically, in place of the demodulator list you find in the Segment of an input file given earlier, I made two changes. I inserted a command, the following, and I included a hot list with the items that provide the diagram.

```
assign(heat,1).
list(hot).
ext(a,c,a,c)=d.
op(q,b,p,d,c)=j.
op(j,b,a,d,p)=e.
op(j,q,a,b,e)=f.
end_of_list.
```

I, of course, commented out the demodulator list and its contents, rather than removing those lines. I also copied the elements in the given hot list to list(usable). They could have been copied to list(sos). With the cited moves, if and when a proof is found, the quoted proof length will reflect accurately the number of deduced items found in the corresponding proof. If memory serves, in almost all the remaining experiments reported here that pertain to assumption-shortening, I relied on the hot list strategy rather than on demodulation.

Instead of detailing the remainder of this journey, I will simply focus on the best of my results, a proof that relies on but nineteen, rather than twenty-eight, items from the input file. As you examine the proof I am about to display, I note that when the hot list strategy is in use, an input item can be cited more than once.

A Proof of Inner Pasch from Outer with a Minimal Input Set

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Mon Apr 29 16:28:37 2013
The command was "otter". The process ID is 7855.
----> UNIT CONFLICT at  36.53 sec ----> 57028 [binary,57027.1,1488.1] $F.

Length of proof is 105.  Level of proof is 27.

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

1 [] T(x,y,ext(x,y,w,v)).
2 [] E(y,ext(x,y,w,v),w,v).
4 [] -T(x,y,u)| -T(y,z,u)|T(x,y,z).
5 [] -E(xa,xb,xc,xd)|E(xc,xd,xa,xb).
7 [] -E(u,v,x,x)|u=v.
8 [] -T(xa,xb,xc)|T(xc,xb,xa).
10 [] -T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xc,xd).
11 [] xa=xb| -T(xa,xb,xc)| -T(xa,xb,xd)|T(xb,xc,xd)|T(xb,xd,xc).
13 [] T(a,p,c).
14 [] T(b,q,c).
15 [] -T(p,x,b)| -T(q,x,a).
```

16 [] g=op(p,b,q,a,f).
17 [] -T(x,v,u)| -T(y,u,z)|T(x,op(v,x,y,z,u),y).
18 [] -T(x,v,u)| -T(y,u,z)|T(z,v,op(v,x,y,z,u)).
19 [] c!=d|c=d.
23 [] op(j,q,a,b,e)=f.
887 [] ext(a,c,a,c)=d.
888 [] op(q,b,p,d,c)=j.
889 [] op(j,b,a,d,p)=e.
890 [] op(j,q,a,b,e)=f.
892 [binary,5.1,2.1] E(x,y,z,ext(u,z,x,y)).
893 (heat=1) [para_into,892.1.4,887.1.1] E(a,c,c,d).
895 [binary,7.1,2.1] x=ext(y,x,z,z).
896 [binary,8.1,4.3] T(x,y,z)| -T(z,y,u)| -T(y,x,u).
897 [binary,8.1,1.1] T(ext(x,y,z,u),y,x).
898 [binary,8.2,4.2] -T(x,y,z)| -T(u,z,x)|T(u,z,y).
900 (heat=1) [para_into,897.1.1,887.1.1] T(d,c,a).
902 [binary,10.2,8.2] -T(x,y,z)|y=z|T(x,z,u)| -T(u,z,y).
911 [binary,13.1,8.1] T(c,p,a).
933 [binary,14.1,8.1] T(c,q,b).
948 [binary,15.1,10.4] -T(q,x,a)| -T(p,y,x)| -T(y,x,b)|y=x.
951 [binary,15.2,10.4] -T(p,x,b)| -T(q,y,x)| -T(y,x,a)|y=x.
961 [binary,17.1,14.1] -T(x,c,y)|T(b,op(q,b,x,y,c),x).
1064 [para_from,23.1.2,16.1.2.5] g=op(p,b,q,a,op(j,q,a,b,e)).
1082 [para_into,893.1.3,19.2.1] E(a,c,d,d)|c!=d.
1095 [binary,896.2,8.2] T(x,y,z)| -T(y,x,u)| -T(u,y,z).
1098 [binary,896.3,8.2] T(x,y,z)| -T(z,y,u)| -T(u,x,y).
1142 [binary,898.2,8.2] -T(x,y,z)|T(u,z,y)| -T(x,z,u).
1146 [hyper,898,13,897] T(ext(a,c,x,y),c,p).
1147 (heat=1) [para_into,1146.1.1,887.1.1] T(d,c,p).
1236 [binary,902.3,8.1] -T(x,y,z)|y=z| -T(u,z,y)|T(u,z,x).
1292 [hyper,896,900,911] T(p,c,d).
1360 [binary,961.1,8.2] T(b,op(q,b,x,y,c),x)| -T(y,c,x).
1361 [binary,961.2,8.1] -T(x,c,y)|T(x,op(q,b,x,y,c),b).
1362 (heat=1) [para_into,1360.1.2,888.1.1,unit_del,1147] T(b,j,p).
1478 [binary,1095.2,911.1] T(p,c,x)| -T(a,c,x).
1486 [para_into,895.1.2,895.1.2] x=x.
1488 [para_from,895.1.2,897.1.1] T(x,x,y).
1489 [para_from,895.1.2,1.1.3] T(x,y,y).
1493 [binary,1098.2,933.1] T(x,q,c)| -T(b,x,q).
1514 [binary,1147.1,1098.2] T(x,c,d)| -T(p,x,c).
1528 [binary,1147.1,11.3] d=c| -T(d,c,x)|T(c,x,p)|T(c,p,x).
1576 [hyper,902,1147,13] c=p|T(d,p,a).
1748 [binary,1292.1,18.2] -T(x,y,c)|T(d,y,op(y,x,p,d,c)).
1855 [binary,1361.2,15.1] -T(p,c,x)| -T(q,op(q,b,p,x,c),a).
1860 (heat=1) [para_into,1855.2.2,888.1.1,unit_del,1292] -T(q,j,a).
2143 [binary,1489.1,1098.2] T(x,y,z)| -T(y,x,y).
2579 [para_from,1576.1.1,900.1.2,factor_simp] T(d,p,a).
2589 (heat=1) [para_into,2579.1.1,887.1.2] T(ext(a,c,a,c),p,a).
2706 [binary,1748.2,8.1] -T(x,y,c)|T(op(y,x,p,d,c),y,d).
2718 (heat=1) [para_into,2706.2.1,888.1.1,unit_del,14] T(j,q,d).
3042 [binary,1860.1,8.2] -T(a,j,q).
3354 [binary,2579.1,8.1] T(a,p,d).
3496 [binary,2589.1,951.3,unit_del,1488] -T(q,ext(a,c,a,c),p)|ext(a,c,a,c)=p.

3950 [binary,2718.1,11.3] j=q| -T(j,q,x)|T(q,x,d)|T(q,d,x).
4183 [binary,3354.1,18.2] -T(x,y,p)|T(d,y,op(y,x,a,d,p)).
4185 [binary,3354.1,17.2] -T(x,y,p)|T(x,op(y,x,a,d,p),a).
4207 (heat=1) [para_into,4183.2.3,889.1.1,unit_del,1362] T(d,j,e).
4502 [para_from,3950.1.1,3042.1.2,unit_del,1489] -T(j,q,x)|T(q,x,d)|T(q,d,x).
4733 [binary,4185.2,8.1] -T(x,y,p)|T(a,op(y,x,a,d,p),x).
4746 (heat=1) [para_into,4733.2.2,889.1.1,unit_del,1362] T(a,e,b).
4944 [hyper,1095,2718,4207] T(q,j,e).
5441 [binary,4746.1,18.2] -T(x,y,e)|T(b,y,op(y,x,a,b,e)).
5443 [binary,4746.1,17.2] -T(x,y,e)|T(x,op(y,x,a,b,e),a).
5469 (heat=1) [para_into,5441.2.3,890.1.1,unit_del,4944] T(b,j,f).
5474 (heat=1) [para_into,5443.2.2,890.1.1,unit_del,4944] T(q,f,a).
5977 [hyper,11,1362,5469] b=j|T(j,p,f)|T(j,f,p).
6094 [binary,5474.1,15.2] -T(p,f,b).
6100 [binary,5474.1,8.1] T(a,f,q).
6469 [binary,5977.2,8.1] b=j|T(j,f,p)|T(f,p,j).
7017 [binary,6469.2,8.1] b=j|T(f,p,j)|T(p,f,j).
7538 [binary,1488.1,948.1] -T(p,x,q)| -T(x,q,b)|x=q.
7545 [binary,1488.1,15.2] -T(p,q,b).
7546 [binary,1488.1,15.1] -T(q,p,a).
7820 [binary,7538.2,933.1] -T(p,c,q)|c=q.
7895 [ur,1098,7545,14] -T(c,p,q).
7940 [binary,7546.1,8.2] -T(a,p,q).
7944 [ur,1098,7546,3354] -T(d,q,p).
7945 [ur,1098,7546,13] -T(c,q,p).
7948 [ur,1095,7546,2579] -T(p,q,d).
8180 [para_from,7820.2.1,1292.1.2,unit_del,7948] -T(p,c,q).
8206 [binary,7895.1,2143.1] -T(p,c,p).
8207 [binary,7895.1,1528.4,unit_del,7945] d=c| -T(d,c,q).
8338 [ur,1142,7944,2718] -T(j,p,q).
8552 [binary,8180.1,1478.1] -T(a,c,q).
8565 [binary,8180.1,8.2] -T(q,c,p).
8673 [para_into,8206.1.1,3496.2.2,unit_del,1146] -T(q,ext(a,c,a,c),p).
8675 (heat=1) [para_into,8673.1.2,887.1.1] -T(q,d,p).
8902 [binary,8338.1,2143.1] -T(p,j,p).
9166 [ur,898,14,8552] -T(a,c,b).
9417 [ur,1095,8902,1362] -T(j,p,b).
26823 [binary,1082.1,7.1] c!=d|a=c.
30755 [para_from,26823.2.1,9166.1.1,unit_del,1488] c!=d.
30826 [ur,1236,1292,30755,8675] -T(q,d,c).
30906 [para_into,30755.1.1,8207.1.2,unit_del,1486] -T(d,c,q).
31185 [binary,30906.1,8.2] -T(q,c,d).
31298 [binary,31185.1,4502.2,unit_del,30826] -T(j,q,c).
49340 [para_into,31298.1.1,7017.1.2,unit_del,14] T(f,p,j)|T(p,f,j).
49966 [hyper,1236,5469,49340,unit_del,6094] T(f,p,j)|j=f.
52685 [para_from,49966.2.1,3042.1.2,unit_del,6100] T(f,p,j).
53043 [hyper,1236,1362,52685] j=p|T(f,p,b).
53809 [para_from,53043.1.1,9417.1.1,unit_del,1488] T(f,p,b).
54690 [binary,53809.1,8.1] T(b,p,f).
55261 [hyper,18,54690,5474] T(a,p,op(p,b,q,a,f)).
55271 [hyper,17,54690,5474] T(b,op(p,b,q,a,f),q).
55420 [para_into,55261.1.3,16.1.2] T(a,p,g).
55421 [binary,55271.1,1493.2] T(op(p,b,q,a,f),q,c).

55585 [hyper,11,13,55420] a=p|T(p,c,g)|T(p,g,c).
55818 [ur,1098,8565,55421] -T(p,c,op(p,b,q,a,f)).
55844 (heat=1) [para_into,55818.1.3.5,890.1.2] -T(p,c,op(p,b,q,a,op(j,q,a,b,e))).
55855 [para_into,55421.1.1,16.1.2] T(g,q,c).
56083 [para_into,55585.2.3,1064.1.1,unit_del,55844] a=p|T(p,g,c).
56391 [ur,1142,55855,30906] -T(g,c,d).
56691 [binary,56083.2,1514.2,unit_del,56391] a=p.
57027 [para_from,56691.1.1,7940.1.1] -T(p,p,q).
57028 [binary,57027.1,1488.1] $F.

You see, in this given proof, items with heat cited, meaning that the hot list strategy played a role in the corresponding deduction. You also see that one input item, the following,is present twice, once because of being in list(sos) and once from being in list(hot).

op(j,q,a,b,e)=f.

The following three items from the input were now used but not used in the first 60-step proof, the third actually being used but with its arguments interchanged.

-T(x,y,u)| -T(y,z,u)|T(x,y,z). $ Transitivity of betweenness
-T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xc,xd). % Satz 3.71
ext(a,c,a,c)=d. % from the Beeson diagram

I like the resulting 105-step proof in part because none of the theorems relied upon, from the input, were unproved by me from the 1959 axiom system. Finally, the following eleven items, taken from the Input Fragment, were used in the given 60-step proof but are not used in a 145-step proof.

-E(xa,xb,xc,xd)|E(xb,xa,xc,xd).
T(x,y,y).
T(xa,xa,xb).
-T(xa,xb,xd)| -T(xb,xc,xd)|T(xa,xc,xd).
-T(xa,xb,xc)| -T(xa,xc,xd)|T(xb,xc,xd).
-T(xa,xb,xc)| -T(xa,xc,xd)|T(xa,xb,xd).
-T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xb,xd).
xa=xb| -T(xa,xb,xc)| -T(xa,xb,xd)|T(xa,xc,xd)|T(xa,xd,xc).
-T(xa,xb,xd)| -T(xa,xc,xd)|T(xa,xb,xc)|T(xa,xc,xb).
d=ext(a,c,a,c).
a!=c|a=c.

I was able to dispense with the tautology a != c | a = c. I was not able to dispense with the tautology c != d | c = d.

If you might enjoy probing further, trying to eliminate more items from the input, searching for a shorter proof, or seeking to satisfy some other aspect, the following input file might serve well; its use will give a 105-step proof of inner from outer, and with connectivity of betweenness commented out.

A Minimized Input File for Proving Inner Pasch from Outer

```
% Tarski's 1959 axiom system.
% That consists of axioms A1-A6, outer pasch instead of A7, A8-A11, transitivity of betweenness
%(Satz 3.5, aka A15), and connectivity of betweeenness (Satz 5.1, aka A18).
% In this file we try to prove inner Pasch from outer Pasch.
% This file works, except for some case distinctions we can't yet remove. See list(sos).


set(hyper_res).
set(para_into).
```

```
set(para_from).
set(ur_res).
set(binary_res).
set(unit_deletion).
set(order_history).
assign(report,5400).
%  assign(max_seconds, 60000).
assign(max_mem,840000).

clear(print_kept).
% set(very_verbose).
set(input_sos_first).
%  set(sos_queue).
set(back_sub).
assign(bsub_hint_wt,-1).
set(keep_hint_subsumers).


assign(max_weight,15).
assign(max_distinct_vars,4).
assign(pick_given_ratio,4).
assign(max_proofs,4).
%  assign(neg_weight,8).
assign(heat,1).

list(sos).
%  x=x.
T(x,y,ext(x,y,w,v)).                    % A4, first half
E(y,ext(x,y,w,v),w,v).                  % A4, second half
%  E(x,x,y,y).  % Satz 2.8
%x = y | ext(u,v,x,y) != v.
%  -T(x,y,x) | x=y.                       % A6
%  -T(xa,xp,xc) | -T(xb,xq,xc) | T(xp,ip(xa,xp,xc,xb,xq),xb).
%A7, first part (inner Pasch)
%  -T(xa,xp,xc) | -T(xb,xq,xc) | T(xq,ip(xa,xp,xc,xb,xq),xa).
%A7, second part (inner Pasch)
% following two clauses are outer Pasch.
%  -T(x,v,u) | -T(y,u,z) | T(x,op(v,x,y,z,u),y).
-T(x,v,u) | -T(y,u,z) | T(z,v,op(v,x,y,z,u)).
% following is the transitivity of betweenness, Satz 3.5, Axiom A15
-T(x,y,u) | -T(y,z,u) | T(x,y,z).

%  E(x,y,x,y).  % Satz 2.1
-E(xa,xb,xc,xd) | E(xc,xd,xa,xb).  % Satz 2.2
%  -E(xa,xb,xc,xd) | E(xb,xa,xc,xd).  % Satz 2.4
-E(xa,xb,xc,xd) | -E(xc,xd,xe,xf) | E(xa,xb,xe,xf).  % Satz 2.3
%  -E(xa,xb,xc,xd) | E(xa,xb,xd,xc).  % Satz 2.5
%  E(x,x,y,y).  % Satz 2.8
-E(u,v,x,x) | u=v.   % Not one of Szmielew's theorems but we proved it.


%  T(x,y,y).    % Satz 3.1
-T(xa,xb,xc) | T(xc,xb,xa).  % Satz 3.2.
```

```
%  T(xa,xa,xb).  % Satz 3.3
%  -T(xa,xb,xc) | -T(xb,xa,xc) | xa = xb.  % Satz 3.4.
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xb,xc). % Satz 3.51.
%  -T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xc,xd). % Satz 3.52.
%  -T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd). % Satz 3.61.
%  -T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd). % Satz 3.62.
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd).  % Satz 3.71
%  -T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd).  % Satz 3.72

%  % Following is Satz 5.1
%  xa = xb | -T(xa,xb,xc) | -T(xa,xb,xd) | T(xa,xc,xd) | T(xa,xd,xc).
% Following is Satz 5.2
xa = xb | -T(xa,xb,xc) | -T(xa,xb,xd)| T(xb,xc,xd) | T(xb,xd,xc).
%  % Following is Satz 5.3
%  -T(xa,xb,xd) | -T(xa,xc,xd) | T(xa,xb,xc) | T(xa,xc,xb).



end_of_list.

list(passive).
%-T(p,c,d) | $ANS(0).
%-T(d,c,p) | $ANS(1).
%-T(a,c,d) | $ANS(2).
%-T(p,c,d) | $ANS(3).
%-T(d,c,p) | $ANS(4).
%-T(c,q,b) | $ANS(5).
%-T(d,q,j) | $ANS(6).
%-T(j,q,d) | $ANS(7).
%-T(p,j,b) | $ANS(8).
%-T(b,j,p) | $ANS(9).
-T(a,c,d) | $ANS(10).
%-T(d,j,e) | $ANS(11).
%-T(a,e,b) | $ANS(12).
%-T(q,j,e) | $ANS(13).
%-T(a,f,q) | $ANS(14).
%-T(q,f,a) | $ANS(15).
%-T(f,j,b) | $ANS(16).
%-T(f,p,b) | $ANS(17).
%-T(q,g,b) | $ANS(18).
%-T(a,p,d) | $ANS(19).
%-T(a,p,g) | $ANS(20).
%c !=op(p,b,q,a,f) | $ANS(21).
%c != g | $ANS(22).

%-T(b,q,d) | $ANS(23).
%-T(b,q,c) | $ANS(24).
%-T(d,c,q) | $ANS(25).
%-T(d,c,b) | $ANS(26).
%-T(d,c,p) | $ANS(27).
%-T(p,b,b) | $ANS(28).
%-T(d,c,a) | $ANS(29).
%-T(a,b,q) | $ANS(30).
```

```
%-T(b,j,f) | $ANS(31).
%-T(f,j,b) | $ANS(32).
%-T(b,j,p) | $ANS(33).
%-T(p,j,b) | $ANS(34).
%-T(b,q,d) | $ANS(35).
%-T(b,q,c) | $ANS(36).

%-T(j,q,d | $ANS(37)).
%-T(b,q,d) | $ANS(38).
%-T(b,q,c) | $ANS(39).
%b = q | $ANS(40).
%-T(q,c,d) | $ANS(41).
%-T(p,c,d) | $ANS(42).
%-T(a,c,q) | $ANS(43).
%-T(p,c,b) | $ANS(44).
%-T(p,c,q) | $ANS(45).
%-T(p,c,d)| $ANS(46).
%-T(d,c,a) | $ANS(47).
%-T(d,c,q) | $ANS(48).
%d=c | $ANS(49).

%-E(a,a,c,d) | $ANS(50).
%-E(a,c,c,d) | $ANS(51).
%-E(c,d,a,a) | $ANS(52).
%-E(c,d,a,c) | $ANS(53).
%-E(a,a,a,c) | $ANS(54).
%a != c | $ANS(55).

end_of_list.

%  list(demodulators).
%  ext(a,c,a,c)=d.
%  op(q,b,p,d,c)=j.
%  op(j,b,a,d,p)=e.
%  op(j,q,a,b,e)=f.
%  end_of_list.

list(sos).  % negated form of inner Pasch
T(a,p,c).
T(b,q,c).
-T(p,x,b) | -T(q,x,a).
% Now we define the diagram for the proof
%  d = ext(a,c,a,c).
%  j = op(q,b,p,d,c).
%  e = op(j,b,a,d,p).
%  f = op(j,q,a,b,e).
g = op(p,b,q,a,f).
% following two clauses are outer Pasch.
-T(x,v,u) | -T(y,u,z) | T(x,op(v,x,y,z,u),y).
-T(x,v,u) | -T(y,u,z) | T(z,v,op(v,x,y,z,u)).
c!=d | c = d.   % works with this tautology present, but not without!

% T(p,f,j) | T(f,p,j).  % This works.
```

```
% a=c. %gives a proof, so we can assume

%  a != c.
%  a != c | a = c.
%  b = j | b != j.   % works with either case, but not with the disjunction.

%q = b.   % That works too, but that's irrelevant.


op(j,b,a,d,p)=e.
op(j,q,a,b,e)=f.
end_of_list.

list(hints2).
%  following 133 prove inner from outer, temp.beeson.outer.inner.new.out8d6q2,
%  relying on many fewer input items than 8a, for ex
E(x,y,z,u)| -E(z,u,y,x).
E(x,ext(y,x,z,u),u,z).
E(c,d,c,a).
-T(x,y,x)|T(z,x,y).
-T(x,y,x)|T(y,x,z).
T(x,y,z)| -T(z,y,u)| -T(y,x,u).
T(ext(x,y,z,u),y,x).
-T(x,y,z)| -T(u,z,x)|T(u,z,y).
T(x,y,z)| -T(y,x,u)| -T(u,y,z).
T(x,y,z)| -T(z,y,u)| -T(u,x,y).
T(d,c,a).
-T(x,y,z)|T(u,z,y)| -T(x,z,u).
-T(x,y,z)|y=z|T(x,z,u)| -T(u,z,y).
-T(x,y,z)|y=z| -T(u,z,y)|T(u,z,x).
T(c,p,a).
T(c,q,b).
-T(x,c,y)|T(b,op(q,b,x,y,c),x).
-T(x,c,y)|T(x,op(q,b,x,y,c),b).
-T(q,x,a)| -T(p,y,x)| -T(y,x,b)|y=x.
-T(q,p,a).
-T(p,x,b)| -T(q,y,x)| -T(y,x,a)|y=x.
-T(p,q,b).
-T(p,a,b).
-T(a,p,q).
-T(b,q,p).
-T(b,a,p).
g=op(p,b,q,a,op(j,q,a,b,e)).
T(d,c,p).
d=c| -T(d,c,x)|T(c,x,p)|T(c,p,x).
T(p,c,d).
-T(p,x,q)| -T(x,q,b)|x=q.
-T(p,q,c).
-T(b,x,a)| -T(x,a,p)|x=a.
-T(a,p,a).
c=p|T(d,p,a).
-T(x,y,c)|T(d,y,op(y,x,p,d,c)).
-T(x,y,c)|T(x,op(y,x,p,d,c),p).
```

T(d,q,j).
T(b,j,p).
T(b,op(q,b,p,d,c),p).
T(p,op(q,b,p,d,c),b).
a!=c.
T(j,q,d).
j=q| -T(j,q,x)|T(q,x,d)|T(q,d,x).
T(p,j,b).
-E(a,c,x,x).
-T(q,j,a).
-T(a,j,q).
-E(x,x,c,a).
-T(q,p,c).
-T(c,p,q).
-T(p,q,x)| -T(q,c,x).
-T(q,c,x)| -T(x,q,p).
-T(q,c,q).
-T(op(q,b,p,d,c),x,p)|T(b,op(q,b,p,d,c),x).
-T(op(q,b,p,d,c),x,b)|T(p,op(q,b,p,d,c),x).
c!=d.
c!=ext(a,c,a,c).
-T(p,c,p).
-T(q,c,b).
-T(b,c,q).
-T(c,p,d).
-T(d,p,c).
T(p,c,ext(a,c,x,y)).
T(x,c,d)| -T(p,x,c).
T(x,q,c)| -T(b,x,q).
T(d,p,a).
T(a,p,d).
T(ext(a,c,a,c),p,a).
-T(d,q,p).
-T(x,y,p)|T(d,y,op(y,x,a,d,p)).
-T(x,y,p)|T(x,op(y,x,a,d,p),a).
T(d,j,e).
T(b,e,a).
T(q,j,e).
T(a,e,b).
-T(x,e,y)|T(y,j,op(j,q,x,y,e)).
-T(x,e,y)|T(q,op(j,q,x,y,e),x).
T(b,j,f).
T(q,f,a).
T(f,j,b).
b=j|T(j,p,f)|T(j,f,p).
b=j|T(j,f,p)|T(f,p,j).
-T(p,f,b).
T(a,f,q).
-T(j,f,b)|j=f.
b=j|T(f,p,j)|T(p,f,j).
-T(j,p,q).
-T(p,j,p).
-T(j,f,b).

-T(f,j,f).
-T(j,f,j).
-T(j,f,op(q,b,p,d,c)).
-T(p,c,q)|c=q.
-T(q,ext(a,c,a,c),p)|ext(a,c,a,c)=p.
-T(p,c,q).
-T(q,c,p).
-T(q,ext(a,c,a,c),p).
-T(q,ext(a,c,a,c),c).
-T(q,d,c).
d=c| -T(d,c,q).
-T(d,c,q).
-T(q,c,d).
-T(op(q,b,p,d,c),q,b)|op(q,b,p,d,c)=q.
-T(op(q,b,p,d,c),q,b).
-T(j,q,b).
-T(c,j,q).
-T(op(q,b,p,d,c),a,p)|op(q,b,p,d,c)=a.
-T(op(q,b,p,d,c),a,p).
-T(j,a,p).
-T(p,a,j).
j=q| -T(b,j,q).
-T(b,j,q).
T(f,p,j)|T(p,f,j).
T(f,p,j)|j=f.
T(f,p,j).
j=p|T(f,p,b).
-T(p,j,f).
-T(p,j,op(j,q,a,b,e)).
T(f,p,b).
T(b,p,f).
T(a,p,op(p,b,q,a,f)).
T(b,op(p,b,q,a,f),q).
T(a,p,g).
T(op(p,b,q,a,f),q,c).
a=p|T(p,c,g)|T(p,g,c).
-T(p,c,op(p,b,q,a,f)).
-T(p,c,op(p,b,q,a,op(j,q,a,b,e))).
T(g,q,c).
a=p|T(p,g,c).
-T(g,c,d).
a=p.
-T(p,p,j).
% following 65 should prove inner from outer, temp.beeson.outer.inner.new.out8d6
T(c,p,a).
T(p,c,ext(a,c,x,y)).
T(c,q,b).
-T(q,p,a).
-T(p,q,b).
-T(p,a,b).
-T(b,a,p).
E(c,d,a,c).
T(a,c,d).

T(d,c,a).
T(p,c,d).
T(d,c,p).
-T(c,q,p).
-T(q,c,a).
-T(c,p,q).
E(d,c,a,c).
T(d,p,a).
T(a,p,d).
T(d,q,j).
T(b,j,p).
T(j,q,d).
T(p,j,b).
-T(q,d,a).
-T(a,d,q).
a!=c.
E(a,c,d,c).
T(d,j,e).
T(b,e,a).
T(a,e,b).
-T(j,a,p).
-E(a,c,x,x).
c!=d.
T(q,j,e).
-T(c,d,q).
-T(q,d,c).
T(b,j,f).
T(q,f,a).
b=j|T(j,p,f)|T(j,f,p).
b=j|T(j,f,p)|T(f,p,j).
-T(p,f,b).
-T(b,f,p).
b=j|T(f,p,j)|T(p,f,j).
-T(j,f,p).
-T(p,f,j).
b=j|T(j,p,f).
T(c,q,j)|T(j,p,f).
T(c,q,j)|T(b,p,f).
T(b,p,f)|T(j,q,c).
T(b,p,f)|j=q|T(q,c,d).
T(b,p,f)|T(q,c,d).
T(b,p,f)|T(d,c,q).
T(b,p,f)|d=c.
T(b,p,f).
T(a,p,op(p,b,q,a,f)).
T(b,op(p,b,q,a,f),q).
T(a,p,g).
T(op(p,b,q,a,f),q,c).
a=p|T(p,c,g)|T(p,g,c).
T(g,q,c).
T(p,c,g)|T(p,g,c).
-T(g,c,a).
-T(a,c,g).

T(p,g,c).
T(p,q,c).
T(c,q,p).
% Following 60 prove inner from outer, temp.beeson.outer.inner.new.out8d, using demod six times.
T(c,p,a).
T(a,p,ext(a,c,x,y)).
T(p,c,ext(a,c,x,y)).
T(c,q,b).
-T(q,p,a).
-T(p,q,b).
-T(p,a,b).
-T(b,a,p).
E(c,d,a,c).
T(a,c,d).
T(a,p,d).
T(p,c,d).
T(d,c,p).
-T(c,q,p).
-T(q,c,a).
-T(a,c,q).
-T(c,p,q).
E(d,c,a,c).
T(d,q,j).
T(b,j,p).
T(j,q,d).
T(p,j,b).
a!=c.
-T(c,q,a).
-T(a,q,c).
E(a,c,d,c).
T(d,j,e).
T(b,e,a).
T(a,e,b).
-T(j,a,p).
-E(a,c,x,x).
c!=d.
T(q,j,e).
-T(c,d,q).
-T(q,d,c).
T(b,j,f).
T(q,f,a).
b=j|T(b,p,f)|T(b,f,p).
-T(p,f,b).
-T(b,f,p).
b=j|T(f,p,b).
T(j,q,c)|T(f,p,b).
T(f,p,b)|j=q|T(q,c,d).
T(f,p,b)|T(q,c,d).
T(f,p,b)|T(d,c,q).
T(f,p,b)|d=c.
T(f,p,b).
T(b,p,f).
T(a,p,op(p,b,q,a,f)).

```
T(b,op(p,b,q,a,f),q).
T(a,p,g).
T(op(p,b,q,a,f),q,c).
a=p|T(a,c,g)|T(a,g,c).
T(g,q,c).
T(a,c,g)|T(a,g,c).
-T(g,c,a).
-T(a,g,c).
-T(a,c,g).
T(c,g,a).
T(a,g,c).
end_of_list.

weight_list(pick_and_purge).
weight(ext(a,c,a,c),1).
weight(op(q,b,p,d,c),1).
weight(op(j,b,a,d,p),1).
weight(op(j,q,a,b,e),1).
end_of_list.

list(hot).
ext(a,c,a,c)=d.
op(q,b,p,d,c)=j.
op(j,b,a,d,p)=e.
op(j,q,a,b,e)=f.
end_of_list.
```

With an examination of the given input file, you might find entertainment when you see how many of the earlier Satz theorems are commented out and, therefore, are not needed. Be careful about using neg_weight, for its use can lead to very, very long clauses in the output. As an aside, I have two minimized proofs that differ only by exchanging the use of 3.51 with transitivity of betweenness.

At this point in my research, I was through with a study of the theorem that asserts the deducibility of inner Pasch from outer Pasch in the presence of the 1959 Tarski system given in Section 1. I turned immediately to a direct attempt to prove, with OTTER, that connectivity of betweenness is dependent on the remaining axioms of the 1959 system. Many experiments did not produce a proof of the theorem now in focus. Yes, I thought I was getting quite close, and I may include the approach I was using. After those many attempts, however, finally I followed a path that you may already have traveled. Specifically, I realized that a proof of the theorem was already in hand, if I was satisfied to have such in which OTTER played only an indirect role. The following argument can be made; and, as noted, you may have made this argument paragraphs ago.

First, I had a proof in hand, from OTTER, that inner Pasch is deducible from outer Pasch in the presence of the 1959 Axiom system. Actually, I had many proofs for this relationship. Second, I had a proof, from OTTER, that connectivity is deducible from an axiom system strongly related to the 1959 system but with outer Pasch (two clauses) replaced by inner Pasch (two clauses). Therefore, provided an important condition is met, I had, by transitivity (in effect), a proof that connectivity is deducible from the 1959 axiom system. Have you written down the crucial condition? Yes, regarding the condition, if one of my proofs of inner Pasch from outer avoids the use of connectivity, then all is in order, with OTTER playing the indirect role of supplying the cited first and second items. Earlier I used the word “fortuitous”, and earlier I hinted that you might, if you had been witnessing my successes with blocking various input items, already have seen what I was totally unaware of at the time. Indeed, one of the proofs I found in my study of the theorem asserting that outer Pasch implies inner Pasch avoids the use of connectivity. Conclusion: I had, from the viewpoint of mathematics in contrast to the viewpoint of automated reasoning, proved that connectivity of betweenness is dependent on the remaining axioms of the Tarski 1959 system, the axioms that included outer Pasch rather than inner Pasch, and, of course, that excluded connectivity of betweenness.

The remaining nontrivial challenge for me was to find, with OTTER (as opposed to relying on the just-given mathematical argument), a proof of the dependence of connectivity, using the 1959 axiom system offered in Section 1. By looking so-to-speak ahead with the aid of many experiments that did not succeed, I note that various obstacles must be overcome. The two proofs that I could try to rely on, inner from outer and the dependence of connectivity from inner Pasch, are each bidirectional. Put another way, if both proofs were forward proofs, then I could begin by placing as hints their proof steps and have a fair chance of success. Indeed, OTTER could begin with the axiom system based on outer Pasch, with connectivity removed and with various theorems proved from that system, deduce inner Pasch (were it a unit clause, for example), and continue on to prove connectivity (were it also a unit clause).

Well, not so hidden in this last sentence, you find the obstacle that concerns the fact that the clauses for inner Pasch, two of them, and that for connectivity are nonunit clauses. You also note that OTTER will not deduce inner Pasch; the proof I have uses its negation in a bidirectional proof. In review, I have a proof that A implies B and a proof that B implies C and, in the A implies B proof, very important, connectivity is not used. But, in contrast to many cases in the past, I have not as yet, with OTTER, been able to find a proof that A implies C. Even with the input file Beeson sent me to use for a start, although I seem to be close, the prize eludes me. Unless magic is found, I must leave to you the challenge of proving, with OTTER or some other automated reasoning program, the dependence of connectivity within the Tarski 1959 axiom system.

I now discuss how close I have come to winning the game and, just perhaps, provide you with an approach you might use to complete the desired proof. I began by choosing as intermediate targets the unit clauses, seventy-seven of them, that are present among the deduced steps of the 122-step proof that deduced, from inner Pasch and axioms and proved theorems, connectivity. I amended the file Beeson sent me with these negated items in list(passive). The approach was one of iteration. In particular—omitting some of the less interesting experiments—I made a run that proved more than half of the seventy-seven intermediate targets; and, for the next run, I took the proof steps of those proofs and adjoined them to the set of support list. More of the seventy-seven intermediate targets were proved, whose proof steps were further adjoined for the next interesting experiment. After not too long, no additional targets were reached. Now, what might you do? I ask in case you wish to participate in my research.

My reasoning, if that is not too flattering—okay, my guess—was to assume that the proof, if found, might resemble the proof of connectivity that relied on inner Pasch. Since my efforts at proving all of the seventy-seven intermediate targets did not succeed, perhaps it was time to focus on some nonunit clauses. I chose from the Beeson hints2 list three of them, the following, and adjoined them in list(sos) for the next experiment.

```
-T(c1,d,b) | -T(d1,c,b) | T(c,e,c1).
-E(c,d,x,y) | E(c,p,x,y).
-T(d1,c,b)  T(d,e,d1).
```

Rather than detailing the unsatisfactory results, I merely note that I did not find the desired proof. Further experiments led essentially nowhere. If doubt exists about the provability, in the context of automated reasoning, of the theorem under study, I note that Gupta in the 1950s provided a proof that is first-order. When I asked Beeson for a new input file, one based (in effect) on Gupta's proof, fine colleague that he is, he supplied such with a warning that some typos might be present, some perhaps from Gupta's work. That file offers in list(passive) targets that correspond to the steps of the Gupta proof. My experiments with the file yields proofs of quite a few of the intermediate targets. However, the proof of connectivity remained out of reach, even after a variety of attacks.

I therefore provide you with what might be termed an unfinished notebook. To finish it in a manner that I prefer would require my supplying an input file and proof, obtainable from it, that connectivity of betweenness is dependent on the remaining 1959 Tarski axioms. The task—the implied challenge—is, perhaps, overwhelming. I suspect that another notebook will be written that follows this one, with more results concerning Tarskian geometry. For now, I pause and await more joint work with Beeson.

An Amazing Approach to Plane Geometry

Larry Wos

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

11. After More Than Thirty-Five Years, Success

Did you enjoy plane geometry in high school? Do you find triangles, rectangles, hexagons, and other such figures esthetically pleasing? Do you recall theorems that asserted the equivalence of two figures with a justification of, say, side-angle-side? Well, whether or not you were or are captivated by this area of mathematics, called plane geometry, you may find most intriguing the material presented in this notebook. Indeed, if you have known and consulted with a brilliant mind that aided you in your research, here you will (possibly again) learn of an assistant, an automated reasoning program, that can also be most effective in the context of finding proofs for deep theorems, theorems taken from geometry. You will find sprinkled throughout data regarding experiments. In poker, success is measured in terms of how often you win and how much you win. In automated reasoning, I measure success in less—shorter proof length, less CPU time, or fewer conclusions retained. Data of this type provides, at least indirectly, important clues to how you might proceed in the future.

You will encounter, for example, situations where the inclusion or omission of a single item can mean the difference between success and failure. Yes, the use of a program such as William McCune's powerful automated reasoning program OTTER can be straightforward or, on the other hand, can be an art. You will learn in these sections how a field, Tarskian geometry, that was virtually impenetrable more than thirty-five years ago, when the Argonne group studied it, finally yielded its treasure to OTTER and to two researchers, Michael Beeson and me—and treasure in abundance.

12. Wellspring and Some Background

The wellspring for this article is my colleague Beeson's intriguing plan for us to use an automated reasoning program to find proofs for all the theorems in the book by Schwabhauser, Szmielew, and Tarski titled *Metamathematische Methoden in der Geometrie* (hereafter referred to as SST). And some background is merited here to explain the origin of that book.

12.1. Geometry as a Testbed

Geometry has been a testbed for automated deduction almost as long as computers have existed. The first experiments were done in the 1950s. In the nineteenth century, geometry was the testbed for the development of the axiomatic method in mathematics, spurred by efforts to prove Euclid's parallel postulate from his other postulates and ultimately the development of non-Euclidean geometry. These efforts culminated in Hilbert's seminal 1899 book *Grundlagen der Geometrie* (*Foundations of Geometry*). In the period 1927-1965, Tarski developed his simple and short axiom system (presented in Section 3.1). In the late 1970s, some experimentation, at Argonne National Laboratory, was conducted aimed at finding proofs from Tarski's axioms; but despite success with proving very simple theorems, several problems were left unsolved. The subject was revisited by Art Quaife, who in his 1992 book *Automated Development of Fundamental Mathematical Theories* reported on the successful solution of some of those challenge problems

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

using an early version of McCune's automated reasoning program OTTER. But several theorems remained that Quaife was not able to prove with OTTER, and he stated them as "challenge problems" in his book.

Quaife's four challenge problems were the following: (1) every line segment has a midpoint; (2) every segment is the base of some isosceles triangle; (3) the outer Pasch axiom (assuming inner Pasch as an axiom); and (4) the first outer connectivity property of betweenness. These are to be proved without relying on any parallel axiom and without even relying on line-circle continuity. They are difficult theorems to prove, the first proofs of which were the heart of H. N. Gupta's Ph.D. thesis under Tarski.

To be more specific, the immediate stimulus for this notebook was the existence of the almost-formal development of many theorems in Tarskian geometry in Part I of SST. Part I is essentially the manuscript developed by Szmielew for her 1965 Berkeley lectures on the foundations of geometry, with "inessential modifications" by Schwabhauser. The manuscript consists of 16 chapters. Quaife's challenge problems occur in the first nine chapters. Gupta's thesis was never published, although it is available from a database of theses; but SST contains the only published version of his proofs.

12.2. Original Plans and a Note about Dependencies

The original goal that Beeson and I had established was to find proofs, not to check proofs, a topic for later in this notebook where the difference is discussed. Beeson says that all the proofs of the theorems through Chapter 12 of the book have been proof checked. Consistent with conversations I had many years ago with Bob Boyer, I warn the uninitiated that checking a proof is a totally different activity from finding a proof for a given theorem. Beeson's plan, though frequently interrupted by other demands, has us prove one theorem after another, then adjoin, for the next study, the theorem in an appropriate list and proceed to the next theorem. As I write this notebook, this plan is being followed and is far from complete.

As in earlier notebooks, you will find here no mention of ruler and compass, no use of actual diagrams—although, with clauses supplied by Beeson, the diagrams (in effect) do play a key role. Instead, you will learn of proofs for theorems in plane geometry that do not rely on your drawing of figures, computing angles, and using items such as a protractor. In early sections, the proofs given here are based on a set of thirteen axioms provided in 1959 by Alfred Tarski for the study of plane geometry. For quite a while, I thought two of these—transitivity and connectivity of betweenness—were dependent. (Indeed, I think it most likely that Tarski himself was unaware of the true facts.) My original plan for this notebook, therefore, called for a discussion and, possibly, proofs of the two dependencies. Thus, much of this notebook proceeds as if the two axioms were dependent. I suggest that as you read the next few sections, you keep in mind that I labored under this erroneous impression, which is born out by my often finding proofs in which both transitivity and connectivity are not employed. When I finally learned of the true situation, I had to abandon the plan. Thus, in Section 4 and following, I will note that transitivity of betweenness is in fact independent when the Tarski 1959 axiom system is in use; and I will, instead, prove two dependencies in another axiom system (closely related to that given in Section 3.1) for Tarskian geometry.

The task of proving that connectivity of betweenness is dependent presents quite a challenge for the system you are about to encounter, a challenge that I did eventually meet, as seen later in this notebook. (At the writing of this notebook, I still have not seen such a proof; perhaps before the notebook is complete, I will have a proof.)

As you browse in this notebook, you will encounter what to me are most piquant phenomena, some that are successes, some that are failures, and some that are puzzles. Indeed, as is sometimes the case when you consult a learned individual, the outcome is hard to predict. For a tiny example, the likelihood of reaching your goal with the use of OTTER can be sharply affected by the values assigned to various parameters. Rather than being a discouragement, however, what you can do with a program such as OTTER will, for many, be astounding. Now, a bit of history is in order.

13. Some History and Some Challenges

I, with colleagues at Argonne National Laboratory, some thirty-five years or more ago sought proofs of elementary theorems with the use of the Tarski axiom system that I present shortly, in a notation that is acceptable to the automated reasoning program OTTER. Our attempts were, in the main, highly unsuccessful, to say the least. I will not present a detailed analysis of why we were so incompetent—although I will hazard some guesses in that regard—but, rather, show you how in 2012 and 2013, with my colleague Beeson and, of course, McCune's automated reasoning program OTTER, proofs of very difficult theorems (to prove) were obtained.

At least in the beginning, I will focus on the use of the axioms our group at Argonne National Laboratory used those decades ago. A slight variation of those axioms will come into play when I progress to my current studies with Beeson; however, because of the cited study of more than thirty-five years ago with its accompanying lack of success, I have chosen for the early part of this notebook to rely on the axioms used so long ago. By doing so, I can follow one of the principles of science: the importance of showing how progress occurred over many, many years. To immediately dispel a natural conjecture, I emphasize that the breakthrough is not mainly a result of far, far more powerful computers; of course, the added power is indeed valuable.

To provide you with a taste of the strangeness—I believe that is a good assessment—of what you will encounter, I offer a few intuitively appealing theorems to consider; indeed, you might treat each of them as a challenge. The axioms that Tarski gives rely on two predicates (relations): T(x,y,z), which may be thought of as asserting that "*y* is between *x* and *z*" (not necessarily strictly between), and E(x1,y1,x2,y2), which may be thought of as asserting that "the distance from *x1* to *y1* is the same as the distance from *x2* to *y2*".

Intuitively, anybody so-to-speak knows what it means to say that the point *B* is between the points *A* and *C*. Indeed, immediately, when you hear such, you think of a line that begins at *A*, passes through *B*, and ends at *C*. One of the theorems you could be asked to prove, relying on the Tarski system and whichever inference rules you had in mind—of course, the clause representation is what is used throughout this notebook—asserts that, for all *x*, *y*, and *z*, if *y* is between *x* and *z*, then *y* is between *z* and *x*. This theorem refers to the symmetry of betweenness. Now, if you (in effect) ask your niece or nephew about this result, you may be told something like "how silly, it's obvious". It certainly is obvious if you depend on a picture. If you are constrained to the use of the Tarski axioms, however, you might find the implied task, of proving this theorem focusing on symmetry of betweenness, far from trivial. You might also be surprised—but perhaps the cited strangeness gives you a warning—to find that axioms focusing on equidistance are utilized. Later in this notebook, I will present approaches to proving the symmetry of betweenness, as well as proving other theorems. Therefore, if you would enjoy seeking your own proof, read only to the end of this section, when you will have in hand the 1959 Tarski axiom system.

Instead, I offer you a simper theorem to consider, once you have in hand the Tarski system that I will shortly supply. You are asked to prove that *y* is between *x* and *y*, for all *x* and *y*. Yes, Tarski does not mean strictly between. A bit harder, I offer you the challenge of proving that *x* is between *x* and *y*, for all *x\fR and y*. The challenge of proving the symmetry of betweenness is clearly more daunting than either of these other two challenges. Again, you are advised to read no further than the completion of this section if you wish to devise your own approach to proving one of the three cited theorems. Now, if you wish to attempt to prove a theorem I have at this point in the notebook not yet even considered for study (but I may focus on it in a later section), I suggest you try to prove that the diagonals of a rectangle bisect each other, relying, of course, on the Tarski system.

You might naturally wonder how those three challenges were met, if they were, by our automated reasoning group those more than thirty-five years ago, or more recently. As you know, recovering the precise details of decades ago is far from simple, and often the effort produces questionable information. However, I can draw on Chapter 6 of a book I wrote in 1987, a book offering "33 Basic Research Problems". My intention at that time, if memory serves, was to increase interest in the field of automated reasoning and present problems that, if solved, would add to the power of reasoning programs. Chapter 6 is devoted to test problems and experimentation, problems that can be used to begin an evaluation of a solution to one of the research problems offered in the cited book. Some of the test problems focus on Tarskian geometry. If you browse in the book, you will find that, in place of the predicate (relation) *T* (used here) for betweenness, the predicate *B* was used, and also that instead of the predicate *E* (used here), the predicate *L* was used.

13.1. Five-Point Theorem

Test Problem 10 asks you to prove the five-point theorem. In particular, you are told that point *A*3 is between points *A*2 and *A*4 and that each of *A*2 and *A*4 is between *A*1 and *A*5. As you may have surmised, the theorem asks you to prove that *A*3 is between *A*1 and *A*5. As for axioms, you are required to use the Tarski axiom set, which consists of the following twenty (so-to-speak) clauses, not counting axioms concerning the substitutivity of equality (as well as other obvious properties of equality), on which I will comment further.

```
identity axiom for betweenness:
-T(x,y,x) | (x = y).
transitivity axiom of betweenness:
-T(x,y,u) | -T(y,z,u) | T(x,y,z).
connectivity axiom for betweenness:
-T(x,y,z) | -T(x,y,u) | (x = y) | T(x,z,u) | T(x,u,z).
reflexivity axiom for equidistance:
E(x,y,y,x).
identity axiom for equidistance:
-E(x,y,z,z) | (x = y).
transitivity axiom for equidistance:
-E(x,y,z,u) | -E(x,y,v,w) | E(z,u,v,w).
Outer Pasch's axiom (2 clauses):
-T(x,v,u) | -T(y,u,z) | T(x,f1(v,x,y,z,u),y).
-T(x,v,u) | -T(y,u,z) | T(z,v,f1(v,x,y,z,u)).
Euclid's axiom (three clauses):
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,z,f2(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,y,f3(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(f2(v6,x,y,z,u),v6,f3(v6,x,y,z,u)).
five-segment axiom:
-E(x1,y1,x2,y2) | -E(y1,z1,y2,z2) | -E(x1,u1,x2,u2) | -E(y1,u1,y2,u2) | -T(x1,y1,z1) | -T(x2,y2,z2) |
  (x1 = y1) | E(z1,u1,z2,u2).
axiom of segment construction (two clauses):
T(x,y,f4(x,y,u,v)).
E(y,f4(x,y,u,v),u,v).
lower dimension axiom (three clauses):
-T(c1,c2,c3).
-T(c2,c3,c1).
-T(c3,c1,c2).
upper dimension axiom:
-E(x,u,x,v) | -E(y,u,y,v) | -E(z,u,z,v) | (u = v) | T(x,y,z) | T(y,z,x) | T(z,x,y).
weakened continuity axiom (two clauses):
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | E(u1,y1,u1,f5(x1,y1,z1,x2,z2,u1)).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | T(x2,f5(x1,y1,z1,x2,z2,u1),z2).
```

The cited book does not offer a proof of the five-point theorem. Indeed, I feel it safe to say that we could not have dented the proof in 1987. If you consult the book, however, you will find in Chapter 6 two proofs relying on the Tarski system. The first proof establishes that, for all *x* and *y*, *x* is between *x* and *y*. The second proof shows that betweenness is indeed symmetric. As the first proof proceeds, you learn that, for all *x* and *y*, *y* is between *x* and *y*. In each of the two proofs—and here you gain much insight into how we attacked Tarskian geometry—the same axiom is cited before deductions are presented. That axiom, the following, is for the substitutivity of equality in the context of the predicate for betweenness. (As in preceding notebooks, "-" denotes logical **not**, and " | " denotes logical **or**.)

```
u != v | -T(x,y,u) | T(x,y,v).
```

If you consult Chapter 6, you will immediately ask why paramodulation was not being used. After all, that inference rule (which generalizes equality substitution) was published in the late 1960s. Further confusion may result when you learn that, if you supply the various axioms focusing on equality properties in the context of the Tarski 13-axiom system, you must adjoin thirty-five additional (so-to-speak) clauses. In the treatment of plane geometry featured in this notebook, paramodulation plays a vital role. And you now have a powerful clue that explains why we, our group, were thwarted those many, many years ago by Tarski; indeed, not using paramodulation was, I fear, more than an oversight. I suggest that, without its use, a challenge offered in 1990 by Art Quaife would not have been met, a challenge that asks for a proof of the dependence of connectivity of betweenness when you use a Tarskian axiom system that relies on inner Pasch. The successful meeting of that challenge is perhaps the high point of what you will read here.

Just as I began this notebook, I told my colleague Ross Overbeek of the joint effort focusing on Tarski being conducted by Beeson and me. He was most pleased. When I told him of the five-point theorem, he suggested the following theorem for study, a theorem to be called the four-point theorem. (As noted, the five-point theorem says that, if A3 is between A2 and A4 and if each of A2 and A4 is between A1 and A5, then A3 is provably between A1 and A5.) For the four-point theorem, if the points *A*2 and *A*3 are each between the points *A*1 and *A*4, then either *A*2 is between *A*1 and *A*3 or the point *A*3 is between *A*1 and *A*2. Overbeek was unable to use OTTER to prove the theorem he suggested; but shortly thereafter, he was able to use the program to obtain a proof of the five-point theorem. At this time in my writing of this notebook, I had as yet not found a proof for the four-point theorem; however, as it turned out, much later in my study of Tarskian geomety, the proof was obtained.

13.2. Proof Shortening, Assumption Reduction, and Other Topics to Come

Two other topics will appear in this notebook. The first concerns proof shortening, an activity that interested various mathematicians and logicians that include Hilbert, C. A. Meredith, A. N. Prior, and D. Ulrich. If, in particular, you find a proof in a book or paper or have found a proof with a program or with your own mind and if you wish to find a more elegant proof, where elegance is concerned with proof length (number of deduced steps), here you will learn how OTTER and various methodologies can often get you your wish. Somewhat similar to proof shortening is what might be termed axiom-and-theorem reduction or assumption reduction. Specifically, in an OTTER proof, you find, before any deduced item is presented, all of the items it used to obtain a proof, items taken from the input. Possibly because of Jesse Alama's success with finding dependent axioms among those offered for some area or for some theorem, which he communicated to me by e-mail, you will find studies aimed at reducing, for a given theorem and proof of that theorem, the number of items (taken from the input) needed to obtain a proof. Those items (from the Szmielew book) ordinarily in this study of Tarski include axioms she supplied and theorems already proved and, typically, adjoined to an input file. (Near the end of the writing of this notebook, I learned by e-mail from Alama that he also studies the reduction of items needed to obtain a proof of some given theorem.)

Well, the time for your independent and uninfluenced consideration of the challenge theorems is up, or you may wish to wait a bit longer before turning to the next section. Indeed, I will, in the next section, discuss possible approaches to proving the four theorems cited in this section. Specifically, to be in focus are $T(x,y,y)$, $T(x,x,y)$, the four-point theorem suggested by Overbeek, and the symmetry of betweenness theorem. These approaches apply not only to the Tarski treatment of geometry but also to areas of algebra, to other fields of mathematics, and to various areas of logic. As this notebook proceeds, focusing on ever-more-difficult theorems, studied jointly by Beeson and me, the methodologies will be utilized again and again. Near the end of this notebook, you will find what might be most intriguing, namely, a theorem whose proof is given from the viewpoint of mathematics and yet resists various attempts to obtain a proof with OTTER even though a first-order proof does exist.

14. Approaches for Proving Theorems and Various Proofs Obtained

I hope your curiosity is mounting, not only in the context of how to use an automated reasoning program to attack the four challenge theorems (focusing on betweenness), but also in the context of how the proofs involve notions that pertain to equidistance. Even if you have but a small interest in geometry, or perhaps none, you might as a puzzle solver find what follows quite entertaining. As you read through this section, I will take you on a labyrinthian journey whose goal, in addition to finding various proofs and illustrating diverse approaches, is the discovery of an approach that is an alternative to the Beeson plan. Specifically, although his intriguing plan of proving one theorem after another and, when successful, adding the theorem to the input file in search of the next proof—which is in the spirit of lemma adjunction—clearly merits intense consideration and is often quite effective, how satisfying it would be to discover an input file whose use would prove, in one run, many, many theorems of the Szmielew book. For a foretaste of what you will find in this section, I note that you will encounter a number of treatments of a single theorem, the five-point theorem. The diverse treatments illustrate many, many different aspects, and they may provide you with alternatives useful in contexts unrelated to that in focus in this notebook.

You might naturally wonder about various approaches to proving theorems in Tarskian geometry. Or, you might have no strong interest in geometry of any type and still wonder about methodologies for proving theorems, especially proving deep theorems. In this section, you will learn of various techniques, accompanied in some cases by commentary focusing on the obstacles to be encountered, the disadvantages, and the advantages of various methodologies.

Before you choose the approach you will use, from among those to be discussed in this section, you are asked to choose which type of proof you wish your program to attempt to complete. I have in mind three types of proof: forward proof, backward proof, and bidirectional proof. In a forward proof, the program relies on the axioms and lemmas that are placed in the input file and uses the denial of the theorem, or part of that denial, only to signal proof completion. With OTTER, when seeking a forward proof, the denial or part of the denial is placed in list(passive). Elements of list(passive) do not participate in the reasoning; they are used to detect unit conflict, for a determination of the completion of a proof, and for subsumption.

Overbeek, upon reading this notebook, correctly pointed out that the concept of forward proof is more complicated than it might at first appear. For a simple example of seeking a forward proof, if the goal is to prove T(x,y,y)—for all *x* and for all *y*, *y* is between *x* and *y*—you place in list(passive) -T(a,b,b). If you intend to use just the Tarski 13-axiom system, supplying no additional lemmas, the program will reason from those axioms with the goal of deducing something that contradicts -T(a,b,b), ordinarily, T(x,y,y). With OTTER, you must place some of the Tarski axioms in list(sos) to enable the program to get started. You could place all of the axioms in list(sos). Forward proofs offer a pleasing characteristic of showing you which deduced lemmas were used in the proof in case you wish to seek a proof that avoids any or all of such.

For a second example of seeking a forward proof, an example that illustrates Overbeek's concern, I offer the theorem that says betweenness is symmetric: For all *x*, *y*, and *z*, if *y* is between *x* and *z*, then *y* is between *z* and *x*. The denial of this theorem can be captured with two clauses, T(a,b,c) and -T(c,b,a). (You see this theorem considered with the input file given shortly, but with the use of three different constants.) If I were seeking a forward proof of this symmetry, I would place T(a,b,c) in list(sos) and place -T(c,b,a) in list(passive). In other words, only part of the denial is placed in list(passive). If both unit clauses, corresponding to the denial, were placed in list(passive), a proof would not be forthcoming. OTTER, given this theorem to prove under such conditions, would complete a forward proof by deducing T(c,b,a).

For a third example, taken from group theory, commutativity can be proved for groups in which the square of *x*, for every *x*, is the identity *e*. Typically, to seek a forward proof, I would place in list(sos) f(x,x) = e, where *f* denotes product, and place f(a,b) != f(b,a) in list(passive). OTTER would ordinarily deduce f(x,y) = f(y,x), commutativity.

In contrast, a backward proof reasons from the denial and from the deductions that result, never reasoning from sets of axioms alone. With OTTER, I would place for a backward proof the denial in list(sos) and place all of the axioms in list(usable). Placing all of the axioms in list(usable) and also placing any lemmas that might be used and that are input prevent any forward reasoning from occurring.

And, as you correctly conclude, for a bidirectional proof, you enable the program to reason forward from some or all of the axioms and (input) lemmas and backward from the denial of the theorem. With OTTER, you place the denial of the theorem in list(sos) and place at least one of the axioms (or input lemmas) in that list also. Evidence supports the position that success is most likely—and frequently in less

CPU time—if you seek a bidirectional proof. For an explanation regarding the increased likelihood of reaching your goal when you employ a bidirectional search, I note that such proofs typically are of a lower level, where input items are at level 0, and a deduced item is of level one greater than the maximum of the levels of the parents used.

If you would enjoy a research problem that I suspect is not easy to solve, you might study the problem of so-called converting a bidirectional proof to a forward proof, constrained (of course) by the inference rules in use. Too often, with OTTER, I find a bidirectional proof for a specific theorem but cannot find a forward proof. The fault does not rest with OTTER. Especially when paramodulation is used in a proof, you or an automated reasoning program may repeatedly fail to complete a forward proof even though a bidirectional or backward proof is in hand. With paramodulation—and here is, I think, a clue—you usually do not permit paramodulating *from* or *into* variables. The sought-after proof may require the program to relax this constraint, permitting one or both types of paramodulation. (Overbeek says he believes he could write a postprocessor to convert bidirectional proofs into forward proofs, using instantiation heavily; instantiation is almost never offered by an automated reasoning program as an inference rule.)

The approaches to be discussed in this notebook are applicable to any of the three types of proof. You will see that I typically prefer a forward proof, but I do give an example of a simple theorem for which I have obtained only a backward proof at this time. One of the simplest and most direct approaches to proof finding asks you to place all of the problem description in list(sos), for OTTER, so that the program can reason from any subset of the items that are found in the input file. You add no lemmas and give no guidelines or hints to influence the program's search. Of course, no surprise, this simple approach sometimes fails to prove the theorem(s) you intend to prove, which is what did indeed occur with a theorem called Z34 (discussed later in this section). With OTTER, you can seek proofs of more than one theorem at a time, as you see in the following input file that focuses on some early theorems from the Szmielew book (that Beeson and I are closely studying), theorems that are in the main intuitively obvious.

Now, before you visit the first of the various input files I offer, you may well enjoy a list of theorems and definitions pertinent to this study of Tarskian geometry. In this list of theorems, you will find some that I have not yet proved. Indeed, some of the theorems are quite difficult to prove, even if you include (besides a set of axioms) previously proved theorems, included as so-to-speak lemmas. You might nevertheless enjoy proceeding on your own.

Theorems and Definitions

```
E(x,y,y,x). % A1 from page 10 of sst
E(x,y,x,y). % Satz 2.1
-E(xa,xb,xc,xd) | E(xc,xd,xa,xb). % Satz 2.2
-E(xa,xb,xc,xd) | E(xb,xa,xc,xd). % Satz 2.4
-E(xa,xb,xc,xd) | -E(xc,xd,xe,xf) | E(xa,xb,xe,xf). %Satz 2.3
-E(xa,xb,xc,xd) | E(xa,xb,xd,xc). % Satz 2.5
E(x,x,y,y). % Satz 2.8
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xb,xa1,xb1) |
-E(xb,xc,xb1,xc1) | E(xa,xc,xa1,xc1). % Satz 2.11
xq = xa | -T(xq,xa,u) | -E(xa,u,xc,xd) | ext(xq,xa,xc,xd) = u. % Satz 2.12
T(x,y,y). % Satz 3.1, also CH3
-E(u,v,x,x) | u=v. % Not one of Szmielew's theorems but we proved it.
-T(xa,xb,xc) | T(xc,xb,xa). % Satz 3.2, also CH4
T(xa,xa,xb). % Satz 2.3, also CH2
-T(xa,xb,xc) | -T(xb,xa,xc) | xa = xb. % Satz 3.4.
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xb,xc). % Satz 3.51.
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xc,xd). % Satz 3.52.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd). % Satz 3.61.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd). % Satz 3.61.
alpha != beta. %related to Satz 3.14; easily provable if added to sst3h.in.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd). % Satz 3.62.
```

```
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd). % Satz 3.71
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd). % Satz 3.72
-IFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xd,xb1,xd1). % Satz 4.2
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xc,xa1,xc1)
| -E(xb,xc,xb1,xc1) | E(xa,xb,xa1,xb1). % Satz 4.3

alpha != beta. % Satz 3.13
beta != gamma.
alpha != gamma.
T(xa,xb,ext(xa,xb,alpha,gamma)). % Satz 3.14, first half
xb != ext(xa,xb,alpha,gamma). % Satz 3.14, second half
% The following many clauses are Definition 4.1
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | T(xa,xb,xc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | T(za,zb,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xa,xc,za,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xb,xc,zb,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xa,xd,za,zd).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xc,xd,zc,zd).
-T(xa,xb,xc) | -T(za,zb,zc) | -E(xa,xc,za,zc) | -E(xb,xc,zb,zc)
| -E(xa,xd,za,zd) | -E(xc,xd,zc,zd) | IFS(xa,xb,xc,xd,za,zb,zc,zd).

% Following 4 are definition 4.4 for n=3
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa1,xa2,xb1,xb2).
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa1,xa3,xb1,xb3).
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa2,xa3,xb2,xb3).
-E(xa1,xa2,xb1,xb2) | -E(xa1,xa3,xb1,xb3) | -E(xa2,xa3,xb2,xb3)
| E3(xa1,xa2,xa3,xb1,xb2,xb3).
% Following three lines are Satz 4.5
-T(xa,xb,xc) | -E(xa,xc,xa1,xc1) | T(xa1,insert(xa,xb,xa1,xc1),xc1).
-T(xa,xb,xc) | -E(xa,xc,xa1,xc1) | E3(xa,xb,xc,xa1,insert(xa,xb,xa1,xc1),xc1).
insert(xa,xb,xa1,xc1) = ext(ext(xc1,xa1,alpha,gamma),xa1,xa,xb).
-E3(x,y,z,u,v,w) | E3(x,z,y,u,w,v). % See sst4q.in, not in Szmielew
-T(xa,xb,xc) | -E3(xa,xb,xc,xa1,xb1,xc1) | T(xa1,xb1,xc1). % Satz 4.6

% following is Definition 4.10
-Col(xa,xb,xc) | T(xa,xb,xc) | T(xb,xc,xa) | T(xc,xa,xb).
Col(xa,xb,xc) | -T(xa,xb,xc).
Col(xa,xb,xc) | -T(xb,xc,xa).
Col(xa,xb,xc) | -T(xc,xa,xb).
% Following are Satz 4.11
-Col(x,y,z) | Col(y,z,x).
-Col(x,y,z) | Col(z,x,y).
-Col(x,y,z) | Col(z,y,x).
-Col(x,y,z) | Col(y,x,z).
-Col(x,y,z) | Col(x,z,y).
Col(x,x,y). % Satz 4.12
-Col(xa,xb,xc) | - E3(xa,xb,xc,xa1,xb1,xc1) | Col(xa1,xb1,xc1). % Satz 4.13
-Col(xa,xb,xc) | -E(xa,xb,xa1,xb1)
| E3(xa,xb,xc,xa1,xb1,insert5(xa,xb,xc,xa1,xb1)). % Satz 4.14
% following is Definition 4.15
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | Col(xa,xb,xc).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E3(xa,xb,xc,xa1,xb1,xc1).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xa,xd,xa1,xd1).
```

```
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xd,xb1,xd1).
-Col(xa,xb,xc) | - E3(xa,xb,xc,xa1,xb1,xc1) | - E(xa,xd,xa1,xd1)
| -E(xb,xd,xb1,xd1) | FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | xa = xb | E(xc,xd,xc1,xd1). % Satz 4.16
xa = xb | -Col(xa,xb,xc) | -E(xa,xp,xa,xq) | -E(xb,xp,xb,xq) | E(xc,xp,xc,xq). % Satz 4.17
xa = xb | -Col(xa,xb,xc) | -E(xa,xc,xa,xc1) | -E(xb,xc,xb,xc1) | xc = xc1. % Satz 4.18
-T(xa,xc,xb) | -E(xa,xc,xa,xc1) | -E(xb,xc,xb,xc1) | xc = xc1. % Satz 4.19
% inner Pasch, two clauses.
-T(xa,xp,xc) | -T(xb,xq,xc) | T(xp,ip(xa,xp,xc,xb,xq),xb).
-T(xa,xp,xc) | -T(xb,xq,xc) | T(xq,ip(xa,xp,xc,xb,xq),xa).
```

In part because of the earlier discussion of forward proofs, access to the positive form of the different theorems, found in the just-given list, may prove most useful when diverse denials come into play. Indeed, for example, when I seek a forward proof of a theorem that takes the form of a nonunit clause, you will see that I often place part of the denial in list(passive) and part in list(sos). Such is the case, in the input file I give shortly, when the theorem that asserts betweenness is symmetric comes into focus, CH4 (also called Satz 3.2). With the various input files, I illustrate approaches that often succeed; I do not seek to find the most effective approach for any given theorem. You will find comments concerning the effects of avoiding the use of diverse inference rules; however, I am not studying this aspect in depth in this notebook (but such a study might prove interesting). For example, with the following input file, the avoidance of binary resolution does not prevent the program from finding the five proofs being sought; in contrast, the avoidance of paramodulation prevents OTTER from proving three of the five. The five theorems in focus with this input file are Satz 2.1, 2.2, 3.1 (CH3), 3.2 (CH4), and 3.3 (CH2).

Input File 1 Illustrating a Simple Approach

```
set(hyper_res).
clear(order_hyper).
set(para_into).
set(para_from).
set(binary_res).
set(ur_res).
set(order_history).
assign(report,5400).
% assign(max_seconds,4).
assign(max_mem,840000).
clear(print_kept).
set(input_sos_first).
set(back_sub).
assign(max_weight,22).
assign(max_distinct_vars,4).
assign(pick_given_ratio,4).
assign(max_proofs,6).
assign(heat,0).

list(sos).
x = x.
% following 20 are translations of first 20 from ch6 of 1987 book for Tarski
-T(x,y,x) | (x = y).
-T(x,y,u) | -T(y,z,u) | T(x,y,z).
-T(x,y,z) | -T(x,y,u) | (x = y) | T(x,z,u) | T(x,u,z).
E(x,y,y,x).
-E(x,y,z,z) | (x = y).
```

```
-E(x,y,z,u) | -E(x,y,v,w) | E(z,u,v,w).
-T(x,v,u) | -T(y,u,z) | T(x,f1(v,x,y,z,u),y).
-T(x,v,u) | -T(y,u,z) | T(z,v,f1(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,z,f2(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,y,f3(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(f2(v6,x,y,z,u),v6,f3(v6,x,y,z,u)).
-E(x1,y1,x2,y2) | -E(y1,z1,y2,z2) | -E(x1,u1,x2,u2) | -E(y1,u1,y2,u2) | -T(x1,y1,z1) | -T(x2,y2,z2) |
  (x1 = y1) | E(z1,u1,z2,u2).
T(x,y,f4(x,y,u,v)).
E(y,f4(x,y,u,v),u,v).
% -T(c1,c2,c3).
% -T(c2,c3,c1).
% -T(c3,c1,c2).
-E(x,u,x,v) | -E(y,u,y,v) | -E(z,u,z,v) | (u = v) | T(x,y,z) | T(y,z,x) | T(z,x,y).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | E(u1,y1,u1,f5(x1,y1,z1,x2,z2,u1)).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | T(x2,f5(x1,y1,z1,x2,z2,u1),z2).
end_of_list.

list(sos).
E(b1,b2,b3,b4).
T(a7,a8,a9).
end_of_list.

list(passive).
-E(a7,a8,a7,a8) | $ANS(CHZ1).
-E(b3,b4,b1,b2) | $ANS(CHZ2).
-T(a1,a1,a2) | $ANS(CH2).
-T(a1,a2,a2) | $ANS(CH3).
-T(a9,a8,a7) | $ANS(CH4).
end_of_list.
```

The following negations, or part of each, of the five theorems to be proved are found in list(passive).

```
-E(a7,a8,a7,a8) | $ANS(CHZ1).
-E(b3,b4,b1,b2) | $ANS(CHZ2).
-T(a1,a1,a2) | $ANS(CH2).
-T(a1,a2,a2) | $ANS(CH3).
-T(a9,a8,a7) | $ANS(CH4).
```

(If you are still seeking your own proofs of theorems cited in Section 1, you might wish to pause here to avoid encountering clues.) The first theorem to prove, Z1, asserts the profound fact that the distance from *x* to *y* is equal to the distance from *x* to *y* for all *x* and *y*. The second theorem, Z2, asserts a type of symmetry, namely, if the distance from *x* to *y* is equal to the distance from *u* to *V* for any four points, then the distance from *u* to *V* is equal to the distance from *x* to *y*. For that proof, in contrast to Z1, you must look to an element in the second occurrence of list(sos), as well as consider the cited denial found in list(passive), in order to view the negation of the theorem in its entirety. Again, I am certain you are not impressed, nor would you be impressed by the proofs of these two theorems that you would obtain if you submitted the given input file to OTTER.

The third, fourth, and fifth theorems, as you no doubt have noted, are three theorems cited in the first section of this notebook. Although the proof of the fourth theorem, CH3, requires but two deductions, as is the case for the proof of the second theorem (Z2), the proof OTTER supplied for CH3 relies on axioms concerning equidistance as well as betweenness. And you now begin to taste the complexity and intrigue offered by Tarski with his axiom system. Yes, although I have just provided a clue, I have not yet given proofs, suspecting that you may still be seeking your own proofs. If you run OTTER with the given input file, you will find yet another item of interest in the context of CH3 versus Z1 and Z2. Indeed, whereas hyperresolution was the only inference rule used in the proofs for Z1 and Z2, with CH3 binary resolution and paramodulation were used. For the proof of CH3, axioms involving equidistance, as noted, were indeed used.

In contrast to CH3, whose proof has length 2, the proof of CH2 has length 5. That 5-step proof relies on more input axioms than does the proof for CH3, some focusing on betweenness, as expected, and some focusing on equidistance. Binary resolution is used twice, paramodulation is used twice, and factoring (quite seldom used) is used once. (Factoring is an inference rule that relies on a single premiss or parent.) For CH4—which is expressed as a nonuit clause and hence its denial produces two unit clauses, one of which was placed in list(passive)—the symmetry of betweenness, again a proof of length 5 was obtained, relying on two applications of binary resolution, one of hyperresolution, and two of paramodulation. And, of course, axioms focusing on equidistance play a key role. Interesting at least to me, in that proof, T(x,y,y) was deduced. What a mind had Tarski!

As part of this discussion of a simple approach to proving theorems, now is the time for identifying which of the thirteen Tarski axioms are dependent, a fact conveyed to me by Beeson. (As noted in Section 1, only the second of the two following axioms is dependent, but I did not learn this for quite a while.)

transitivity axiom of betweenness
-T(x,y,u) | -T(y,z,u) | T(x,y,z).
connectivity axiom for betweenness
-T(x,y,z) | -T(x,y,u) | (x = y) | T(x,z,u) | T(x,u,z).

To prove the dependence of either of the axioms is far from easy—especially since the first is not dependent—but I had intended to consider the corresponding tasks later in this notebook. I introduce the topic of axiom dependence in part because a natural run or experiment asks for proofs of the five theorems in focus with an input file that omits the two (thought-to-be) dependent axioms. In this case a shortcut is available, as you see from the following proof of CH4.

A 5-Step Proof of CH4

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Wed Nov 14 10:16:12 2012
The command was "otter". The process ID is 16942.
----> UNIT CONFLICT at 0.11 sec ----> 381 [binary,380.1,25.1] $ANS(CH4).

Length of proof is 5. Level of proof is 4.

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

2 [] -T(x,y,x)|x=y.
6 [] -E(x,y,z,z)|x=y.
8 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
9 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
14 [] T(x,y,f4(x,y,u,v)).
15 [] E(y,f4(x,y,u,v),u,v).
20 [] T(a7,a8,a9).
25 [] -T(a9,a8,a7)|$ANS(CH4).
31 [binary,8.3,2.1] -T(x,y,z)| -T(x,z,u)|x=f1(y,x,x,u,z).
34 [binary,15.1,6.1] x=f4(y,x,z,z).
132 [para_from,34.1.2,14.1.3] T(x,y,y).
200 [hyper,31,132,20] a7=f1(a8,a7,a7,a9,a8).
380 [para_from,200.1.2,9.3.3,unit_del,132,20] T(a9,a8,a7).
```

381 [binary,380.1,25.1] $ANS(CH4).

An inspection of this proof shows that neither of the two dependent axioms is relied upon. Further, such is the case for the other four proofs that are yielded by using the given input file. (Since this notebook is just that, a notebook, I ask you to think with me as I labored under the impression that transitivity of betweenness is dependent, which it is not in the given 13-axiom system of Tarski.)

Before turning to another approach or methodology, one additional element merits discussion. In particular, when I discussed the use of binary resolution with Overbeek, he asserted that it was not needed, not at all. I said that I believed I had proofs obtained from a related Tarski system that appeared to be out of reach if binary resolution was blocked from use. Overbeek scoffed. So, naturally, I at this point in the notebook made a run with the given input file with but one change, namely, the removal (commenting out) of binary resolution. I will almost immediately supply the proof obtained for CH4, but first note that, for example, the last line of the cited 5-step proof (not counted among the five) cites binary resolution in a manner that shows the proof to be a forward proof because of listing an input clause. Often, I do not include such a line because it is typically the empty clause. (Of course, when the theorem has the form of a nonunit clause, as is the case for the symmetry of betweenness, then the last line, before binary resolution with no actual clause being displayed, cannot be the theorem to prove.) For your possible entertainment, I now give the proof that was obtained with the use of binary resolution blocked.

A 6-Step Proof of CH4 That Avoids Binary Resolution

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Wed Nov 14 16:26:38 2012
The command was "otter". The process ID is 20917.
----> UNIT CONFLICT at 0.62 sec ----> 616 [binary,615.1,25.1] $ANS(CH4).

Length of proof is 6. Level of proof is 5.

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

2 [] -T(x,y,x)|x=y.
6 [] -E(x,y,z,z)|x=y.
8 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
9 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
14 [] T(x,y,f4(x,y,u,v)).
15 [] E(y,f4(x,y,u,v),u,v).
20 [] T(a7,a8,a9).
25 [] -T(a9,a8,a7)|$ANS(CH4).
31 [hyper,6,15] x=f4(y,x,z,z).
97 [para_from,31.1.2,14.1.3] T(x,y,y).
115 [hyper,9,20,97] T(a9,a8,f1(a8,a7,x,a9,a9)).
118 [hyper,8,20,97] T(a7,f1(a8,a7,x,a9,a9),x).
581 [hyper,2,118] a7=f1(a8,a7,a7,a9,a9).
615 [para_from,581.1.2,115.1.3] T(a9,a8,a7).
616 [binary,615.1,25.1] $ANS(CH4).
```

Overbeek has not yet examined this last proof. When and if he does, he will be most pleased. Nevertheless, I hope that you and I, and perhaps Overbeek, will learn later in this notebook that when far more difficult theorems will be in focus, binary resolution is indeed needed; evidence will be given in this context. I strongly suspect that those many, many years ago, in addition to failing to use paramodulation, our automated reasoning group at Argonne National Laboratory did not use binary resolution in our studies of Tarskian geometry. If so, these two failings, and the presence of so many nonunit clauses of a non-Horn nature, in part explain, it seems, why we made so little progress in the mid-1980s and earlier. I have in hand two input files, *A* and *B*, that are identical except for the fact that *A* excludes the use of binary resolution and *B* includes it. With *A*, OTTER finds a proof of interest, a proof of a theorem that I call Z11, which is Satz 2.11 found in the list of theorems and Definitions given earlier; in contrast, with *B*, that proof is not found. On the one hand, you might think that *B* would be the choice because of having access to one additional inference rule, binary resolution. On the other hand, so it appears, its availability may simply drown the program in unneeded conclusions.

Of the numerous paths this notebook might take, some based on current experiments and some on earlier runs, you probably would prefer the path that focuses on a second methodology. The methodology is far from complicated, consisting merely of the inclusion of a so-called fuller axiom set. For example, in the study of group theory, one axiom system relies on associativity of, say, multiplication, the existence of a left identity element e with $ex = x$, and, with respect to e, a left inverse that asserts that, for every x, there exists a y with $yx = e$. McCune often conducted studies in group theory with this axiom system. For such studies, I preferred the axiom system that offers a two-sided identity e, with $ex = xe = x$, and a two-sided inverse, for all x a y exists with $yx = xy = e$. (In answer to your possible curiosity, I believe McCune's preference was based on esthetics; mine, I strongly suspect, was based on my introduction to group theory at the University of Chicago and my doctoral study of groups.) Yes, as many of you know, the system that McCune preferred consists of an independent set of axioms; the one I prefer includes two dependent axioms, that of right identity and that of right inverse. The axiom system with dependent items can be thought of as a fuller axiom system.

You might wonder about the use of the fuller Tarski axiom system versus an axiom system that does not include the two (thought-to-be) dependent items. To explore this question, I conducted two experiments. In the first, I used a Tarski system that avoids the presence of the two cited items and the use of binary resolution. In the second, I took that just-described input file but amended it by including the two cited items. The following proof was found with the second of the two input files, but that proof was not found with the first.

A Proof Relying on Thought-to-Be Dependent Axioms

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Thu Nov 15 07:17:37 2012
The command was "otter". The process ID is 28025.
----> UNIT CONFLICT at 0.02 sec ----> 399 [binary,398.1,49.1] $ANS(CHZ34).

Length of proof is 2. Level of proof is 2.

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

2 [] -T(x,y,x)|x=y.
3 [] -T(x,y,u)| -T(y,z,u)|T(x,y,z).
41 [] T(da,db,dc).
42 [] T(db,da,dc).
49 [] da!=db|$ANS(CHZ34).
263 [hyper,3,41,42] T(da,db,da).
398 [hyper,2,263] da=db.
399 [binary,398.1,49.1] $ANS(CHZ34).
```

The proof uses as an axiom, the transitivity of betweenness, obtaining a 2-step proof. The theorem that is proved, Z34 (Satz 3.4), simply says that if you have three points such that the second is between the first and third and the first is between the second and third, then the first and second points are the same point. Hardly profound! You might immediately deduce that the axiom of transitivity is indeed powerful and, further, conjecture that its proof, from an 11-axiom set of independent axioms, might be difficult to obtain.

Indeed, at the moment my attempts to complete a proof of this theorem, from the independent Tarski 11-axiom system, focusing on transitivity have all failed. (Of course, as you learn later in this notebook, none could succeed since transitivity of betweenness is indeed independent of the 11-axiom system.) More important from the viewpoint of this section, you see that the approach of using a fuller set of axioms can offer much. You might also conjecture that the proof of Z34 that is theoretically obtainable from the 11-axiom system would have length greater than 2.

So you have evidence of the value of including axioms that are dependent when you are searching for various proofs. But, as you might immediately suggest, a disadvantage must exist also. Indeed, because of the presence of additional axioms to use, the CPU time and the number of clauses to be retained on the way to some proof will likely increase. The proof of Z11, a deeper theorem, when the two (thought-to-be) dependent axioms were omitted required 5,737 CPU-seconds and the retention of 415,387 clauses; when the two so-called dependent axioms were included in the input, the proof of Z11 required 19,783 CPU-seconds to complete and the retention of 992,081 clauses. As a reminder, in clause form, Z11, as a theorem, is the following.

-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xb,xa1,xb1) | -E(xb,xc,xb1,xc1) | E(xa,xc,xa1,xc1).

So, in addition to esthetic notions, in view of the preceding comments, perhaps McCune preferred to employ sets of axioms free of dependencies for efficiency considerations, of course recognizing the dangers.

Perhaps one of the most natural ways to proceed when seeking a proof, especially if you suspect the proof will be difficult to obtain, is to (in effect) emulate work in published papers and books. Similarly, you might emulate the approach taken by some mathematician you have observed as that person begins the study of a purported theorem. Specifically, such an approach often consists of conjecturing about how a proof might proceed, where the conjecture focuses on a number of lemmas or lesser theorems that, when suitably used together with some additional steps, reaches the desired conclusion. With the conjecture in hand, the conjecturer turns to proving the lemmas and lesser theorems that, in effect, provide an outline of a hoped-for proof.

For example, consider the case in which the goal is to prove the five-point theorem: that is, if you know that $A3$ is between $A2$ and $A4$, and you know that each of $A2$ and $A4$ is between $A1$ and $A5$, then you are to prove that $A3$ is between $A1$ and $A5$. A reasonable conjecture, one made by Overbeek in fact, asserts that the following three lemmas, if proved, would or could play a key role in obtaining the sought-after proof. First, you might prove that for all x and for all y, y is between x and y. As for this lemma or lesser theorem, you see that the notion of betweenness is not requiring strictly between. Also, I am (in effect) leading the witness in that this lemma is easier to prove than is the next possibly useful lemma. Second, you are to prove that for all x and for all y, x is between x and y. Each of the cited two lemmas is clearly true, intuitively. So, too, is the third lemma to prove, namely, a lemma about symmetry of betweenness: For all x, y and z, if y is between x and z, then y is between z and x. And yes, the third cited lemma is harder to prove than either of its two predecessors.

In the approach under discussion, once you have proved the three lemmas, various choices exist. You could simply adjoin the statement of each of the three to the set of Tarski axioms; after all, as a reminder, the goal originally was to prove the five-point theorem from the thirteen Tarski axioms. (I do not know whether Overbeek stopped to prove the three lemmas; he did use them, successfully, to seek a proof of the five-point theorem, adjoining to the usable list the statement of each.) Instead, you could adjoin to those axioms all the proof steps of the proofs of the three lemmas, of course observing the warnings about proofs of nonunit lemmas. Since this notebook is about automated reasoning as well as about geometry, you might wish me to pause at this point to focus on the two given choices and provide some detail about what you might do precisely.

Well, one clear choice is to adjoin each of the three statements to the input to be used and place each in list(sos). To ensure that each will receive immediate and close consideration, you would also include the command set(input_sos_first). Or, in the context of the second cited choice, you could amend list(sos) for the next experiment with all the proof steps of the three proofs, of course omitting the last step that simply shows, with binary resolution, that success has occurred with no mention of a formula or equation. You must be careful, especially with regard to adjoining the proof steps. In particular, consider the case in which you are seeking a proof of symmetry of betweenness, and you succeed. The proof might list, say, T(a,b,c), before listing any deduced clause, and it might list -T(c,b,a). For this illustration, I will assume that -T(c,b,a) is placed in list(passive) with the intention of producing a forward proof. The proof you might obtain could have as its last step T(c,b,a), the last step before presenting the contradiction by way of employing binary resolution. This step, taken together with -T(c,b,a), is a contradiction. You would not include T(c,b,a) as a so-called lemma, because of its being a proof step—in fact, the conclusion of a proof of the symmetry of betweenness. The inclusion of that clause clearly is not justified. Instead, you must keep in mind that the theorem to be proved takes the form of a nonunit clause—in the case of symmetry of betweenness, -T(x,y,z | T(z,y,x)—a clause that, when proved, can be adjoined to the input you plan to use in your attempt to reach your goal, for example, a proof of the five-point theorem.

Overbeek, of course, had in his input file, in addition to the thirteen Tarski axioms, three clauses, the following.

T(x,y,y).
T(x,x,y)
-T(x,y,z | T(z,y,x).

To be patently clear, if he had proved symmetry of betweenness, he would not have taken the last line of the proof, whether a forward proof, a backward proof, or a bidirectional proof. If, instead of so-called lemma adjunction you were to rely on resonators, a different story could be told.

Resonators are equations or formulas that do not take on a **true** or a **false** value. They are patterns that are conjectured to be of interest, to be used to direct a program's reasoning. The idea is that if a deduced item matches (where variables are in effect ignored) a resonator, then that deduced item should be focused on very soon, because it is conjectured to merit attention. Whereas the adjunction of proof steps or the last line of proofs, on the way to reaching the goal, emulates what is done in unaided research, to a far lesser extent the use of resonators does not emulate a person. However, the use of resonators can be astoundingly powerful. As for the warning about lemmas that take the form of nonunit clauses, so-to-speak **if-then** statements, you can use the last lines of their proofs as is, use intermediate proof steps, and use (what can be thought of as) parts of either as resonators.

For example, consider the case in which you have just proved symmetry of betweenness (represented with the following clause), believing it to be an important lemma on the way to proving, say, the five-point theorem.

-T(x,y,z) | T(z,y,x).

You could include as resonators either of the literals of this clause without the worry of soundness. Of course, you could follow the resonator approach and also, at the same time, follow the (preceding) lemma adjunction approach. So, for greater clarity, you could be told about a possible proof, or you could guess at how a proof might proceed; and then you could try to prove as lemmas or lesser theorems the elements of the so-called outline. In succeeding runs or experiments or attempts, you could, often profitably (ignoring the last line that cites the binary resolution for unit conflict), use the results of success with each lemma and/or the proof steps thereof as resonators and as adjoined lemmas. You must be careful, as discussed, regarding the form taken by lemmas and lesser theorems, when proved, keeping in mind those that take the form of unit clauses and those that take the form on nonunit clauses.

For a fuller example, in the context of resonators, you could, and probably would, take all the proof steps of the three proofs of the three cited lemmas, to be used as resonators, assign to each a small value, and place this set in weight_list(pick_and_purge). Indeed, let *S* be a proof step in one of the three proofs. You would then find, in the amended input file, something like the following.

weight(S,2).

Of course, with the just-described approaches you run the risk that your outline for a proof is not useful, that the lemmas and lesser theorems you prove provide little assistance in completing the sought-after proof. Further, with (possibly) many additional items to reason from, items that turn out to be of little or no use, your program can focus on each of them and deduce so much irrelevant information that it virtually drowns. Similarly, when you include resonators that prove to be of essentially no value, their use can lead the program down one path after another, paths that are fruitless. Indeed, with small values, high priorities for directing the reasoning, assigned to such resonators, equations or formulas that match a resonator of no value will be chosen quickly as the focus of attention and lead the program away from key information.

Instead of considering the methodologies just discussed, you might wish to simply dive right in and submit to OTTER the thirteen Tarski axioms and the theorem to be proved. What would happen, you ask, if you instructed OTTER to prove the five-point theorem without supplying any lemmas or lesser theorems, in other words, without conjecturing about the nature of the proof? Well, I did just that, and OTTER returned the following.

A Proof of the Five-Point Theorem Based on Tarski's Thirteen Axioms Alone

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Wed Nov 14 09:33:18 2012
The command was "otter". The process ID is 16380.
----> UNIT CONFLICT at 280.70 sec ----> 130710 [binary,130709.1,25.1] $ANS(CH1).

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

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

2 [] -T(x,y,x)|x=y.
4 [] -E(x,y,z,z)|x=y.
6 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
7 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
12 [] T(x,y,f4(x,y,u,v)).
13 [] E(y,f4(x,y,u,v),u,v).
18 [] T(a2,a3,a4).
19 [] T(a1,a2,a5).
20 [] T(a1,a4,a5).
25 [] -T(a1,a3,a5)|$ANS(CH1).
37 [binary,6.3,2.1] -T(x,y,z)| -T(x,z,u)|x=f1(y,x,x,u,z).
39 [binary,13.1,4.1] x=f4(y,x,z,z).
95 [binary,20.1,7.2] -T(x,y,a4)|T(a5,y,f1(y,x,a1,a5,a4)).
189 [para_from,39.1.2,12.1.3] T(x,y,y).
195 [binary,189.1,7.2] -T(x,y,z)|T(z,y,f1(y,x,u,z,z)).
315 [hyper,37,189,18] a2=f1(a3,a2,a2,a4,a3).
328 [para_from,37.3.2,7.3.3,factor_simp,factor_simp] -T(x,y,z)| -T(x,z,u)|T(u,y,x).
693 [para_from,315.1.2,7.3.3,unit_del,189,18] T(a4,a3,a2).
739 [hyper,7,693,19] T(a5,a3,f1(a3,a4,a1,a5,a2)).
746 [hyper,6,693,19] T(a4,f1(a3,a4,a1,a5,a2),a1).
13908 [para_into,95.2.3,37.3.2,unit_del,20,factor_simp] -T(a1,x,a4)|T(a5,x,a1).
50002 [para_into,195.2.3,37.3.2,unit_del,189,factor_simp] -T(x,y,z)|T(z,y,x).
50295 [binary,50002.2,13908.1] -T(a4,x,a1)|T(a5,x,a1).
56288 [binary,50295.1,746.1] T(a5,f1(a3,a4,a1,a5,a2),a1).
130709 [hyper,328,739,56288] T(a1,a3,a5).
130710 [binary,130709.1,25.1] $ANS(CH1).
```

You quickly deduce from the given proof that rather than simply supplying appropriate clauses, I did make some decisions. In particular, I chose which inference rules to use. Indeed, binary resolution was used six times to deduce appropriate conclusions; hyperresolution was used four times; paramodulation was used five times. Of course, I am not counting the citing of binary resolution in the line that establishes UNIT CONFLICT or in the last line that presents no clause other than the empty clause. More information concerning my choices can be gleaned from a further examination of the given proof. First, since clause (25), which is part of the denial of the conclusion of the five-point theorem, was not used as a parent from any deduction of a new conclusion, you might correctly surmise that that clause was placed in list(passive), blocking its use in making deductions. Second, since clause (25) was cited in the UNIT CONFLICT line and it appears before any deduced item, you can correctly conclude that the given proof is a forward proof. The presence of various applications of paramodulation and of binary resolution causes me to strongly hazard that, those many, many years ago, our group used neither in our attempts with Tarski. Also, the abundance of many nonunit deduced clauses suggests to me that, long ago, nonunit clauses often presented a problem for us.

Especially if you wish to glean much about automated reasoning, and perhaps about proof in general, you might wonder about the need for all four cited inference rules. After all, Overbeek (in effect) asserts that binary resolution best not be used. You might also wonder about the effect of placing clause (25) in list(sos) to seek a bidirectional proof. Although you cannot tell from the given proof, I note that all but the denial clause were placed in list(sos); therefore, you might wonder what would occur if list(usable) were used. Then you might be curious about the results if some obvious lemmas were included in the input, rather than relying solely on the Tarski system. Further, you might ask about relying on the Tarski system with the two thought-to-be dependent axioms removed. Answers to questions of this type might eventually lead to important insights about the geometry, about Tarski, and about automated reasoning, perhaps in the context of strategy. Let us supply some answers, in part based on experiments.

I begin with the item focusing on the need for all four inference rules: binary resolution, hyperresolution, paramodulation, and UR-resolution. Because equality is present in the Tarski axioms, that suggests that paramodulation merits use; because nonunit clauses containing one positive literal are present, that naturally suggests hyperresolution should be used; because unit clauses are often so valuable, UR-resolution is suggested; and the obvious candidate for removal is binary resolution. By commenting out binary resolution from the input file that was used to obtain the earlier-cited 15-step proof, you can see what will happen with the following input file.

A Second Input File, a File Proving the Five-Point Theorem

```
set(hyper_res).
clear(order_hyper).
set(para_into).
set(para_from).
set(binary_res).
set(ur_res).
set(order_history).
assign(report,5400).
assign(max_mem,840000).
clear(print_kept).
set(input_sos_first).
set(back_sub).
assign(max_weight,22).
assign(max_distinct_vars,4).
assign(pick_given_ratio,4).
assign(max_proofs,6).
assign(heat,0).

list(sos).
x = x.
% following 20 are translations of first 20 from ch6 of book2 for Tarski
```

```
-T(x,y,x) | (x = y).
% -T(x,y,u) | -T(y,z,u) | T(x,y,z).
% -T(x,y,z) | -T(x,y,u) | (x = y) | T(x,z,u) | T(x,u,z).
E(x,y,y,x).
-E(x,y,z,z) | (x = y).
-E(x,y,z,u) | -E(x,y,v,w) | E(z,u,v,w).
-T(x,v,u) | -T(y,u,z) | T(x,f1(v,x,y,z,u),y).
-T(x,v,u) | -T(y,u,z) | T(z,v,f1(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,z,f2(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,y,f3(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(f2(v6,x,y,z,u),v6,f3(v6,x,y,z,u)).
-E(x1,y1,x2,y2) | -E(y1,z1,y2,z2) | -E(x1,u1,x2,u2) | -E(y1,u1,y2,u2) |
  -T(x1,y1,z1) | -T(x2,y2,z2) | (x1 = y1) | E(z1,u1,z2,u2).
T(x,y,f4(x,y,u,v)).
E(y,f4(x,y,u,v),u,v).
-E(x,u,x,v) | -E(y,u,y,v) | -E(z,u,z,v) | (u = v) | T(x,y,z) | T(y,z,x) | T(z,x,y).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | E(u1,y1,u1,f5(x1,y1,z1,x2,z2,u1)).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | T(x2,f5(x1,y1,z1,x2,z2,u1),z2).
end_of_list.

list(sos).
E(b1,b2,b3,b4).
T(a2,a3,a4).
T(a1,a2,a5).
T(a1,a4,a5).
T(a7,a8,a9).
T(A2,a6,a4).
end_of_list.

list(passive).
-E(a7,a8,a7,a8) | $ANS(CHZ1).
-E(b3,b4,b1,b2) | $ANS(CHZ2).
-T(a1,a3,a5) | $ANS(CH1).
-T(a5,a3,a1) | $ANS(CH11).
-T(a1,a1,a2) | $ANS(CH2).
-T(a1,a2,a2) | $ANS(CH3).
-T(a9,a8,a7) | $ANS(CH4).
-T(a2,a3,a6) | $ANS(CH5A).
-T(a2,a6,a3) | $ANS(CH5B).
end_of_list.
```

A glance at the input file shows that two of Tarski's axioms, that for transitivity of betweenness and that for connectivity of betweenness, have been commented out (by using a percent sign). Certain subquestions, so to speak, arise. You could ask about the results to be obtained if binary resolution is not used. You could ask about the results that would be obtained with an input file that, in addition to avoiding the use of binary resolution, permitted the use of the two clauses commented out in the just-given input file. You could ask about the results of permitting the use of binary resolution and the use of the two commented-out clauses. I conducted these very experiments. They complete an obvious set of four, where the fourth involves blocking the use of two thought-to-be dependent clauses while permitting the use of binary resolution.

After many hours, only one of the three experiments produced a proof of the five-point theorem. The experiment that succeeded relied on binary resolution and had access to the clauses for transitivity of betweenness and connectivity of betweenness. I will shortly discuss the proofs that were found, each early.

But first, I note that as I waited impatiently, not long after submitting the three experiments, I examined what was happening. And here is what I found (after I first label the experiments for possible convenience), in the context of proofs relevant to the use of lemmas, if that is a choice to be considered.

The experiment described earlier that yielded the 15-step proof is called EXP1; it relies on the use of binary resolution and excludes access to the clauses for both transitivity and connectivity of betweenness. EXP2 blocks the use of binary resolution and is otherwise identical to EXP1. EXP3 Blocks binary resolution, but it permits the use of both transitivity and connectivity of betweenness. EXP4 allows the use of binary resolution and also allows the program to reason from both transitivity and connectivity of betweenness. Lemma 1 asserts that T(x,y,y); Lemma 2 asserts that T(x,x,y); Lemma 3 asserts the symmetry of betweenness.

My typical approach to experimentation is to include, in addition to the main goal, other goals that are conjectured to be more easily reached. These lesser goals, if and when proved, often provide useful clauses for later experiments, especially when the main goal has not yet been reached. (By focusing on related experiments and the data that results, you can glean some understanding and appreciation of choices that can be made.) In all four experiments, Lemmas 1, 2, and 3 (as well as others of lesser interest) were included as targets. In all four experiments, the three lemmas were proved, but not in an identical manner. A closer examination of EXP1 yields the following interesting results.

In the 5-step proof of Lemma 2, you find (as will be seen shortly) the deduction of Lemma 1, T(x,y,y).

A 5-Step Proof of Lemma 2 with Experiment 1

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Wed Nov 14 09:33:18 2012
The command was "otter". The process ID is 16380.
----> UNIT CONFLICT at 0.05 sec ----> 330 [binary,329.1,27.1] $ANS(CH2).

Length of proof is 5. Level of proof is 3.

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

2 [] -T(x,y,x)|x=y.
4 [] -E(x,y,z,z)|x=y.
6 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
7 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
12 [] T(x,y,f4(x,y,u,v)).
13 [] E(y,f4(x,y,u,v),u,v).
27 [] -T(a1,a1,a2)|$ANS(CH2).
37 [binary,6.3,2.1] -T(x,y,z)| -T(x,z,u)|x=f1(y,x,x,u,z).
39 [binary,13.1,4.1] x=f4(y,x,z,z).
189 [para_from,39.1.2,12.1.3] T(x,y,y).
328 [para_from,37.3.2,7.3.3,factor_simp,factor_simp] -T(x,y,z)| -T(x,z,u)|T(u,y,x).
329 [factor,328.1.2,unit_del,189] T(x,x,y).
```

And here is a proof of Lemma 1 that was found in Experiment 1 before the 5-step proof just given, a proof that, when compared with the 5-step proof, offers perhaps surprising similarity.

A 2-Step Proof of Lemma 1 from Experiment 1

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
```

```
Wed Nov 14 09:33:18 2012
The command was "otter". The process ID is 16380.
----> UNIT CONFLICT at 0.03 sec ----> 190 [binary,189.1,28.1] $ANS(CH3).

Length of proof is 2. Level of proof is 2.

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

4 [] -E(x,y,z,z)|x=y.
12 [] T(x,y,f4(x,y,u,v)).
13 [] E(y,f4(x,y,u,v),u,v).
28 [] -T(a1,a2,a2)|$ANS(CH3).
39 [binary,13.1,4.1] x=f4(y,x,z,z).
189 [para_from,39.1.2,12.1.3] T(x,y,y).
```

Then, I offer you a proof of Lemma 3 from Experiment 1, a proof that so-to-speak borrows from the proofs it found while proving Lemmas 1 and 2.

A 5-Step Proof of Lemma 3 from Experiment 1

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Wed Nov 14 09:33:18 2012
The command was "otter". The process ID is 16380.
----> UNIT CONFLICT at 0.08 sec ----> 485 [binary,484.1,29.1] $ANS(CH4).

Length of proof is 5. Level of proof is 4.

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

2 [] -T(x,y,x)|x=y.
4 [] -E(x,y,z,z)|x=y.
6 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
7 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
12 [] T(x,y,f4(x,y,u,v)).
13 [] E(y,f4(x,y,u,v),u,v).
21 [] T(a7,a8,a9).
29 [] -T(a9,a8,a7)|$ANS(CH4).
37 [binary,6.3,2.1] -T(x,y,z)| -T(x,z,u)|x=f1(y,x,x,u,z).
39 [binary,13.1,4.1] x=f4(y,x,z,z).
189 [para_from,39.1.2,12.1.3] T(x,y,y).
312 [hyper,37,189,21] a7=f1(a8,a7,a7,a9,a8).
484 [para_from,312.1.2,7.3.3,unit_del,189,21] T(a9,a8,a7).
```

As noted before and exemplified by the proof just presented, the last line of a proof cannot necessarily simply be adjoined as a lemma in future investigations. In the case of Lemma 3, if your program supplies a proof of the type just viewed, you can adjoin a clause for symmetry of betweenness (as a lemma) in further experimentation. After all, the lemma or lesser theorem that was proved is a nonunit clause whose negation or denial had you place two unit clauses in the input file designed to prove Lemma 3.

If you are new to the use of OTTER and perhaps to the use of any automated reasoning program, you have strong evidence, if you review the results of Experiment 1, of the patience you must sometimes exercise. In particular, the five-point theorem (the main goal) was also proved in the experiment, but only after a (at least subjectively) long wait. Indeed, as you see from the data, apparently—and that is the correct term—nothing was occurring for more than 4 CPU-minutes. Of course, OTTER was continuously drawing conclusions. Sometimes, days of CPU time elapse between proofs that are found in a single run.

Next in order is a glance at the other three experiments, beginning with EXP2, which differs from EXP1 only in the blocking of the use of binary resolution.

As with EXP1, EXP2 quickly yielded proofs of Lemmas 1, 2, and 3, in that order. The proofs were somewhat different from those cited earlier from EXP1, since the use of binary resolution was blocked. Again, as you will shortly see, the proof of Lemma 2 contains a proof of Lemma 1.

A 6-Step Proof of Lemma 2 from Experiment 2

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Sat Nov 24 14:28:22 2012
The command was "otter". The process ID is 28354.
----> UNIT CONFLICT at 0.60 sec ----> 964 [binary,963.1,27.1] $ANS(CH2).

Length of proof is 6. Level of proof is 5.

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

2 [] -T(x,y,x)|x=y.
4 [] -E(x,y,z,z)|x=y.
6 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
7 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
12 [] T(x,y,f4(x,y,u,v)).
13 [] E(y,f4(x,y,u,v),u,v).
27 [] -T(a1,a1,a2)|$ANS(CH2).
37 [hyper,4,13] x=f4(y,x,z,z).
138 [para_from,37.1.2,12.1.3] T(x,y,y).
179 [hyper,7,138,138] T(x,x,f1(x,y,z,x,x)).
190 [hyper,6,138,138] T(x,f1(y,x,z,y,y),z).
960 [hyper,2,190] x=f1(y,x,x,y,y).
963 [para_from,960.1.2,179.1.3] T(x,x,y).
```

The initial segment of this proof closely resembles the initial segment of the 5-step proof of Lemma 2 obtained with EXP1 with the exception of stating that hyperresolution was used in place of binary resolution. Indeed, the two clauses, parents, that are used for the first step yield the same conclusion whether hyperesolution or binary resolution is used. Whereas hyperresolution is used in this 6-step proof, it is not used, in the beginning, in the 5-step proof yielded with EXP1.

As for Lemma 3, instead of a 5-step proof, EXP2 yielded a 6-step proof, the following.

A 6-Step Proof of Lemma 3 from Experiment 2

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Sat Nov 24 14:28:22 2012
The command was "otter". The process ID is 28354.
----> UNIT CONFLICT at 0.82 sec ----> 1414 [binary,1413.1,29.1] $ANS(CH4).

Length of proof is 6. Level of proof is 5.

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

2 [] -T(x,y,x)|x=y.
4 [] -E(x,y,z,z)|x=y.
6 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
7 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
12 [] T(x,y,f4(x,y,u,v)).
13 [] E(y,f4(x,y,u,v),u,v).
21 [] T(a7,a8,a9).
29 [] -T(a9,a8,a7)|$ANS(CH4).
37 [hyper,4,13] x=f4(y,x,z,z).
138 [para_from,37.1.2,12.1.3] T(x,y,y).
181 [hyper,7,21,138] T(a9,a8,f1(a8,a7,x,a9,a9)).
192 [hyper,6,21,138] T(a7,f1(a8,a7,x,a9,a9),x).
1405 [hyper,2,192] a7=f1(a8,a7,a7,a9,a9).
1413 [para_from,1405.1.2,181.1.3] T(a9,a8,a7).

You see that the avoidance of using binary resolution, at least for proofs of Lemmas 1, 2, and 3, has relatively little effect, almost none in CPU time. However, whereas EXP1 yielded a proof of the five-point theorem, EXP2 did not, even after more than 20 CPU-hours when the program ran out of the alloted memory, slightly less than 1,000 megabytes. Does this data suggest that the use of binary resolution is crucial? Perhaps, as in EXP3 the addition of two input clauses, for transitivity and connectivity of betweenness, will enable the program to prove the five-point theorem.

In EXP3, as noted, again binary resolution was blocked, but two clauses not allowed to participate in EXP2 were allowed to participate in EXP3. In particular, as you have predicted, the two correspond to transitivity and connectivity of betweenness. I thought that, just possibly, EXP3 would produce results a bit unlike those of EXP2. Such was not the case. Indeed, ignoring the clause numbers, all is the same. The presence of the additional clauses did not result in a proof of the five-point theorem either. So, on the surface and to the possible annoyance of Overbeek, evidence was accruing to support the position that binary resolution is needed if a proof of the five-point theorem is to be found without including various lemmas.

In EXP4, the program not only was permitted the use of binary resolution but also was given access to those two elusive clauses, transitivity and connectivity of betweenness. The program quickly proved Lemmas 1, 2, and 3; and the proofs looked like those found in EXP1. With regard to the five-point theorem, however, in contrast to EXP1, you will see in the following proof that far more computer time was needed. In other words, if you had been sitting near a terminal and watching for a proof of the five-point theorem, having in mind the few CPU-minutes required in EXP1, you would most likely have experienced much impatience.

A 13-Step Proof of the Five-Point Theorem from Experiment 4

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Sat Nov 24 15:54:36 2012
The command was "otter". The process ID is 29017.
----> UNIT CONFLICT at 23457.24 sec ----> 617695 [binary,617694.1,27.1] $ANS(CH1).

Length of proof is 13. Level of proof is 7.

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

2 [] -T(x,y,x)|x=y.
6 [] -E(x,y,z,z)|x=y.
8 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
9 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).

14 [] T(x,y,f4(x,y,u,v)).
15 [] E(y,f4(x,y,u,v),u,v).
20 [] T(a2,a3,a4).
21 [] T(a1,a2,a5).
22 [] T(a1,a4,a5).
27 [] -T(a1,a3,a5)|$ANS(CH1).
40 [binary,8.3,2.1] -T(x,y,z)| -T(x,z,u)|x=f1(y,x,x,u,z).
43 [binary,15.1,6.1] x=f4(y,x,z,z).
83 [binary,21.1,9.2] -T(x,y,a2)|T(a5,y,f1(y,x,a1,a5,a2)).
124 [hyper,9,20,22] T(a5,a3,f1(a3,a2,a1,a5,a4)).
126 [hyper,8,20,22] T(a2,f1(a3,a2,a1,a5,a4),a1).
285 [para_from,43.1.2,14.1.3] T(x,y,y).
292 [binary,285.1,9.2] -T(x,y,z)|T(z,y,f1(y,x,u,z,z)).
430 [para_from,40.3.2,9.3.3,factor_simp,factor_simp] -T(x,y,z)| -T(x,z,u)|T(u,y,x).
5855 [para_into,83.2.3,40.3.2,unit_del,21,factor_simp] -T(a1,x,a2)|T(a5,x,a1).
61112 [para_into,292.2.3,40.3.2,unit_del,285,factor_simp] -T(x,y,z)|T(z,y,x).
61533 [binary,61112.2,5855.1] -T(a2,x,a1)|T(a5,x,a1).
109892 [binary,61533.1,126.1] T(a5,f1(a3,a2,a1,a5,a4),a1).
617694 [hyper,430,124,109892] T(a1,a3,a5).

Of course, you would expect, as I did, that the two elusive clauses play a role in this 13-step proof. No, they do not. Why, then, is the proof so much different from the 15-step proof found in EXP1? I cannot say anything of substance. I leave the analysis to you.

And now, perhaps, you have concluded that binary resolution not only is useful but, perhaps, is needed to prove the five-point theorem. But wait: Overbeek's patience has run out. Indeed, he almost shouts, directing the researcher to the use of lemmas, for example, Lemmas 1, 2, and 3. Well, rather than conducting the experiment, perhaps called EXP5, I simply browsed among my numerous experiments for that which he would approve of. And I found an input file whose max_weight was assigned the value of 15 rather than 22 as in preceding data; and its output file, I found, does support Overbeek's conviction that binary resolution is not needed for the proof of the five-point theorem. You will see, in the following proof, taken from the experiment of weeks ago, that binary resolution was not used. To my surprise, however, neither Lemma 1 nor Lemma 2 was used in the proof; Lemma 3 was used.

A 10-Step Proof of the Five-Point Theorem without the Use of Binary Resolution

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on steamroller,
Sat Nov 17 10:43:06 2012
The command was "otter". The process ID is 32251.
----> UNIT CONFLICT at 32868.65 sec ----> 297169 [binary,297168.1,28.1] $ANS(CH1).

Length of proof is 10. Level of proof is 8.

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

1 [] -T(x,y,x)|x=y.
7 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
8 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
23 [] -T(x,y,z)|T(z,y,x).
24 [] T(a2,a3,a4).
25 [] T(a1,a2,a5).
26 [] T(a1,a4,a5).
28 [] -T(a1,a3,a5)|$ANS(CH1).

81 [hyper,26,8,24] T(a5,a3,f1(a3,a2,a1,a5,a4)).
86 [hyper,26,7,24] T(a2,f1(a3,a2,a1,a5,a4),a1).
602 [hyper,86,23] T(a1,f1(a3,a2,a1,a5,a4),a2).
5246 [hyper,602,7,25] T(a1,f1(f1(a3,a2,a1,a5,a4),a1,a1,a5,a2),a1).
101468 [hyper,5246,1] a1=f1(f1(a3,a2,a1,a5,a4),a1,a1,a5,a2).
101473 [para_from,101468.1.2,8.3.3,unit_del,602,25] T(a5,f1(a3,a2,a1,a5,a4),a1).
101476 [hyper,101473,8,81] T(a1,a3,f1(a3,a5,a5,a1,f1(a3,a2,a1,a5,a4))).
101479 [hyper,101473,7,81] T(a5,f1(a3,a5,a5,a1,f1(a3,a2,a1,a5,a4)),a5).
297163 [hyper,101479,1] a5=f1(a3,a5,a5,a1,f1(a3,a2,a1,a5,a4)).
297168 [para_from,297163.1.2,101476.1.3] T(a1,a3,a5).

You now have in hand evidence of how sturdy OTTER is. How piquant that Lemma 3, symmetry of betweenness, is used as a parent in but one deduction! From a global viewpoint, you have a charming example of the methodology for finding proofs that is based on lemma adjunction. This 10-step proof is a forward proof; but, as seen before, its last line does not provide you with a lemma to use in further studies. Part of the denial of the five-point theorem is used to draw conclusions. I leave to you the investigation that focuses on variants of Experiments 2 and 3, for example, where the research focuses on assigning various values to the diverse parameters offered by McCune's program.

Other experiments with the five-point theorem offer food for thought. For a first thought, both the 15-step and the 13-step proof of the five-point theorem are forward proofs, neither reasoning from the denial of the conclusion. A natural experiment has you seek a bidirectional proof; after all, as observed much earlier in this notebook, bidirectional proofs often complete in less CPU time, sometimes far less. A powerful example of what can occur when seeking a bidirectional proof was found by including Lemmas 1, 2, and 3 and a lemma or theorem that Overbeek used in his study of the five-point theorem. In particular, he included, as I did in the following, his four-point theorem, discussed early in this notebook: If a3 and a6 are, respectively, between a2 and a4, then either a3 is between a2 and a6 or a6 is between a2 and a3. Yes, your objection is valid: as yet I have not seen from anybody a proof of the four-point theorem, proved by using the Tarski 13-axiom system or smaller. Nevertheless, I now present the first bidirectional proof (of the five-point theorem), of the ten proofs it found, as an illustration of what can occur.

A 42-Step Bidirectional Proof of the Five-Point Theorem

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Thu Nov 29 15:54:09 2012
The command was "otter". The process ID is 28156.
----> UNIT CONFLICT at 28.31 sec ----> 47093 [binary,47092.1,46664.1] $ANS(CH1).

Length of proof is 42. Level of proof is 11.

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

1 [] -T(x,y,x)|x=y.
2 [] -T(x,y,u)| -T(y,z,u)|T(x,y,z).
3 [] -T(x,y,z)| -T(x,y,u)|x=y|T(x,z,u)|T(x,u,z).
7 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
8 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
21 [] T(x,y,y).
22 [] T(x,x,y).
23 [] -T(x,y,z)|T(z,y,x).
24 [] -T(x,u,z)| -T(x,v,z)|T(x,u,v)|T(x,v,u).
25 [] T(a2,a3,a4).
26 [] T(a1,a2,a5).

27 [] T(a1,a4,a5).
28 [] -T(a1,a3,a5)|$ANS(CH1).
33 [hyper,25,23] T(a4,a3,a2).
52 [hyper,26,23] T(a5,a2,a1).
66 [hyper,26,7,22] T(x,f1(x,x,a1,a5,a2),a1).
72 [hyper,27,23] T(a5,a4,a1).
82 [hyper,27,8,25] T(a5,a3,f1(a3,a2,a1,a5,a4)).
87 [hyper,27,7,25] T(a2,f1(a3,a2,a1,a5,a4),a1).
88 [hyper,27,7,22] T(x,f1(x,x,a1,a5,a4),a1).
93 [ur,28,23] $ANS(CH1)| -T(a5,a3,a1).
130 [hyper,52,8,33] T(a1,a3,f1(a3,a4,a5,a1,a2)).
135 [hyper,52,7,33] T(a4,f1(a3,a4,a5,a1,a2),a5).
141 [hyper,72,24,52] T(a5,a2,a4)|T(a5,a4,a2).
153 [hyper,72,8,25] T(a1,a3,f1(a3,a2,a5,a1,a4)).
158 [hyper,72,7,25] T(a2,f1(a3,a2,a5,a1,a4),a5).
228 [hyper,141,2,25] T(a5,a4,a2)|T(a5,a2,a3).
426 [hyper,228,2,33] T(a5,a2,a3)|T(a5,a4,a3).
820 [hyper,426,23] T(a5,a4,a3)|T(a3,a2,a5).
2697 [hyper,66,2,21] T(x,a1,f1(a1,a1,a1,a5,a2)).
2698 [hyper,66,1] a1=f1(a1,a1,a1,a5,a2).
2798 [para_from,2698.1.1,93.2.3] $ANS(CH1)| -T(a5,a3,f1(a1,a1,a1,a5,a2)).
2808 [para_from,2698.1.1,52.1.3] T(a5,a2,f1(a1,a1,a1,a5,a2)).
3140 [ur,82,2,93] -T(a3,a1,f1(a3,a2,a1,a5,a4))|$ANS(CH1).
3186 [hyper,87,23] T(a1,f1(a3,a2,a1,a5,a4),a2).
3265 [hyper,88,1] a1=f1(a1,a1,a1,a5,a4).
3386 [para_from,3265.1.1,93.2.3] $ANS(CH1)| -T(a5,a3,f1(a1,a1,a1,a5,a4)).
3396 [para_from,3265.1.1,72.1.3] T(a5,a4,f1(a1,a1,a1,a5,a4)).
4597 [ur,130,2,28] -T(a3,a5,f1(a3,a4,a5,a1,a2))|$ANS(CH1).
4724 [hyper,135,23] T(a5,f1(a3,a4,a5,a1,a2),a4).
7586 [hyper,2697,1] f1(a1,a1,a1,a5,a2)=a1.
9173 [ur,3186,2,3140] -T(a3,a1,a2)|$ANS(CH1).
9210 [ur,9173,2,26] $ANS(CH1)| -T(a3,a1,a5).
9241 [ur,9210,23] $ANS(CH1)| -T(a5,a1,a3).
9307 [para_into,9241.2.2,7586.1.2] $ANS(CH1)| -T(a5,f1(a1,a1,a1,a5,a2),a3).
9310 [para_into,9241.2.2,3265.1.1] $ANS(CH1)| -T(a5,f1(a1,a1,a1,a5,a4),a3).
14411 [hyper,3396,3,820,unit_del,3386,9310] a5=a4|T(a3,a2,a5)|$ANS(CH1).
14413 [hyper,3396,3,426,unit_del,3386,9310] a5=a4|T(a5,a2,a3)|$ANS(CH1).
16727 [hyper,14413,3,2808,unit_del,9307,2798] a5=a4|$ANS(CH1)|a5=a2.
17799 [para_from,16727.2.1,28.1.3] -T(a1,a3,a2)|$ANS(CH1)|a5=a4.
44934 [ur,4597,2,21] $ANS(CH1)| -T(a5,f1(a3,a4,a5,a1,a2),a5).
45902 [para_into,4724.1.3,17799.2.2,unit_del,44934] -T(a1,a3,a2)|$ANS(CH1).
45912 [para_into,4724.1.3,14411.1.2,unit_del,44934] T(a3,a2,a5)|$ANS(CH1).
46664 [ur,45902,2,153] $ANS(CH1)| -T(a3,a2,f1(a3,a2,a5,a1,a4)).
47092 [hyper,45912,2,158] $ANS(CH1)|T(a3,a2,f1(a3,a2,a5,a1,a4)).
47093 [binary,47092.1,46664.1] $ANS(CH1).

This proof, if shown to Overbeek, would please him in part. Indeed, binary resolution is not used to deduce any steps, appearing only in the UNIT CONFLICT step and in the final step that asserts (in effect) a deduction of the empty clause. Of the forty-two steps, eight were with UR-resolution, twenty-five with hyperresolution, and nine with paramodulation. You well might wish to know what might occur if the clause for the four-point theorem is deleted; you must wait at this time to see if I later succeed in proving the Overbeek four-point theorem.

To meet your wish, I needed only to comment out the unwanted clause. Again OTTER produced a proof, actually ten proofs, the first one of length 30 rather than 42, and a proof that relies on the same elements of the input file except, of course, the clause for the four-point theorem. Yes, I was surprised, expecting that with one additional hypothesis, the four-point theorem clause, most likely a shorter proof would have been found. That was not the case. The 30-step proof contains twenty steps not in the 42-step proof. Because you might be curious about the nature of this 30-step proof, I now present it.

A 30-Step Bidirectional Proof of the Five-Point Theorem

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Thu Nov 29 15:48:56 2012
The command was "otter". The process ID is 28103.
----> UNIT CONFLICT at 4545.44 sec ----> 318935 [binary,318934.1,278209.1] $ANS(CH1).

Length of proof is 30. Level of proof is 9.

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

1 [] -T(x,y,x)|x=y.
2 [] -T(x,y,u)| -T(y,z,u)|T(x,y,z).
3 [] -T(x,y,z)| -T(x,y,u)|x=y|T(x,z,u)|T(x,u,z).
7 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
8 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
21 [] T(x,y,y).
22 [] T(x,x,y).
23 [] -T(x,y,z)|T(z,y,x).
24 [] T(a2,a3,a4).
25 [] T(a1,a2,a5).
26 [] T(a1,a4,a5).
27 [] -T(a1,a3,a5)|$ANS(CH1).
32 [hyper,24,23] T(a4,a3,a2).
46 [hyper,24,7,22] T(x,f1(x,x,a2,a4,a3),a2).
51 [hyper,25,23] T(a5,a2,a1).
70 [hyper,26,23] T(a5,a4,a1).
80 [hyper,26,8,24] T(a5,a3,f1(a3,a2,a1,a5,a4)).
85 [hyper,26,7,24] T(a2,f1(a3,a2,a1,a5,a4),a1).
128 [hyper,51,8,32] T(a1,a3,f1(a3,a4,a5,a1,a2)).
133 [hyper,51,7,32] T(a4,f1(a3,a4,a5,a1,a2),a5).
265 [hyper,46,2,21] T(x,a2,f1(a2,a2,a2,a4,a3)).
673 [hyper,80,23] T(f1(a3,a2,a1,a5,a4),a3,a5).
727 [hyper,85,23] T(a1,f1(a3,a2,a1,a5,a4),a2).
1482 [ur,128,2,27] -T(a3,a5,f1(a3,a4,a5,a1,a2))|$ANS(CH1).
1609 [hyper,133,23] T(a5,f1(a3,a4,a5,a1,a2),a4).
3037 [hyper,265,7,21] T(x,f1(a2,x,y,f1(a2,a2,a2,a4,a3),a2),y).
3047 [hyper,265,1] f1(a2,a2,a2,a4,a3)=a2.
5191 [hyper,727,7,25] T(a1,f1(f1(a3,a2,a1,a5,a4),a1,a1,a5,a2),a1).
9594 [ur,1609,2,1482] -T(a3,a5,a4)|$ANS(CH1).
9635 [ur,9594,2,70] $ANS(CH1)| -T(a3,a5,a1).
9664 [ur,9635,23] $ANS(CH1)| -T(a1,a5,a3).
274135 [para_into,3037.1.2.4,3047.1.1] T(x,f1(a2,x,y,a2,a2),y).
274547 [hyper,274135,2,21] T(x,y,f1(a2,y,y,a2,a2)).
275483 [hyper,274547,2,21] T(x,f1(a2,y,y,a2,a2),y).
```

276936 [ur,274547,2,27] -T(a1,a3,f1(a2,a5,a5,a2,a2))|$ANS(CH1).
278209 [hyper,275483,2,673] T(f1(a3,a2,a1,a5,a4),a3,f1(a2,a5,a5,a2,a2)).
317503 [hyper,5191,1] a1=f1(f1(a3,a2,a1,a5,a4),a1,a1,a5,a2).
317538 [para_from,317503.1.2,8.3.3,unit_del,727,25] T(a5,f1(a3,a2,a1,a5,a4),a1).
317539 [hyper,317538,23] T(a1,f1(a3,a2,a1,a5,a4),a5).
317723 [hyper,317539,2,673] T(a1,f1(a3,a2,a1,a5,a4),a3).
317771 [hyper,317723,3,317539,unit_del,9664,27] a1=f1(a3,a2,a1,a5,a4)|$ANS(CH1).
318934 [para_from,317771.1.1,276936.1.1] -T(f1(a3,a2,a1,a5,a4),a3,f1(a2,a5,a5,a2,a2))|$ANS(CH1).
318935 [binary,318934.1,278209.1] $ANS(CH1).

You have strong evidence, if you compare the 42-step proof with the 30-step proof, of the value of giving the program the additional clause for the four-point theorem. Indeed, although a shorter proof was found, in its absence far, far less CPU time was required (to complete the 42-step proof) and far, far fewer new conclusions were retained. On the other hand, a possible mystery is offered, for you might guess that access to an additional hypothesis would ordinarily lead to a shorter proof. A possible answer asserts that the 42-step proof resulted because the presence of the additional hypothesis led OTTER down a sharply different path, an answer supported by the difference between the two proofs. When a further modification of the input file that produced the 30-step proof was made, namely, removing access to all lemmas and reasoning from just the Tarski 13-axiom system, no proof was obtained. You thus have more evidence of relying on an approach that uses lemmas. As an intermediate summary, Overbeek's position about not needing binary resolution has been supported by the data presented from the last discussed experiments.

For a more startling example of how the presence of but one additional hypothesis in the input file can produce dramatic results, I first remind you of a 10-step proof of the five-point theorem I presented earlier, one that completed in just over 32,868 CPU-seconds. The input file gave access for OTTER to Lemmas 1, 2, and 3, in addition to the Tarski 13-axiom system, including, therefore, the axioms of both transitivity and connectivity of betweenness. Almost immediately after I studied the bidirectional proofs just discussed, I turned to seeking a forward proof. Specifically, I gave the program an input file that contained the Tarski 13-axiom system and the three lemmas just cited and one additional lemma (or lesser theorem), namely, the four-point theorem of Overbeek. After all, as the data I've supplied shows, its presence can indeed have a sharp effect. Well, it did. The CPU time to complete a proof, in contrast to the approximately 32,868 CPU-seconds, was approximately 1,666 CPU-seconds. As noted, the 10-step proof required the retention of (130709) clauses; the proof under discussion required (48751) clauses.

And now for the drama: rather than completing a proof of roughly length 10, the proof that was returned to me has length 138. Again you see how the presence of one additional hypothesis, rather than leading to a shorter proof, led to a far longer proof; certainly a different path of reasoning was followed. The 10-step proof used but one added lemma, that for symmetry of betweenness; the 138-step used that lemma and also used the clause corresponding to the Overbeek four-point theorem. So, if a proof is finally found for his theorem and I learn of it, or discover it myself—which, actually, I did—the data suggests it will be most useful in the ways discussed here and in preceding paragraphs.

A bit more on the five-point theorem is still in order, especially in view of Overbeek's position about using lemmas and avoiding the use of binary resolution and in view of an important choice that you can make when seeking a proof of an interesting theorem. Specifically, at least with OTTER, you are asked to choose from among the clauses presenting the problem, which are to be placed in list(usable) and which in list(sos). As a reminder, OTTER is not permitted to reason from a set of clauses all of which are in the (initial) list(usable). Such clauses are used to complete the application of an inference rule. Therefore, the placement of an item in the usable list blocks the program from choosing it to begin a path of reasoning. The more items you place in list(usable), as opposed to placing such in list(sos), the more you are restricting the program's attack, an action that in the main adds to efficiency in the study of many areas of mathematics and logic. In general, restriction of reasoning increases the likelihood that unprofitable paths will be ignored. On the other hand, if a so-called key item is placed in list(usable), rather than in list(sos), then a path that might be required for exploration in order to find a proof may be blocked. So you see how complex can be the decisions to make when using an automated reasoning program. As I have continually noted, no algorithm exists for making choices of the types featured here in this notebook.

An experiment that strongly asks to be run has you place all the clauses in list(sos), as opposed to what was done in many of the preceding experiments that placed quite a few clauses in list(usable). That experiment also asks that binary resolution be avoided and that, perhaps in part as a consequence, that Lemmas 1, 2, and 3 be used. Also, in contrast to an experiment featured rather early in this section that yielded a 15-step proof, all thirteen of Tarski's axioms are to be used, including the two for, respectively, transitivity and connectivity of betweenness. For clarity, the clause corresponding to the Overbeek four-point theorem is not to be included. With an assignment of the value 15 to max_weight, the experiment was conducted. An 11-step forward proof was completed in approximately 16,096 CPU-seconds, with the retention of clause (164950). Present in the 11-step proof are the use of both Lemmas 1 and 3, T(x,y,y) and symmetry of betweenness.

But, are the three lemmas needed? Indeed, would the Tarski 13-axiom system suffice? The answer is that his axiom system does suffice, with no added lemmas—not even needed was the four-point theorem—as established by simply conducting another experiment, one in which the three lemmas are commented out. That experiment yielded a 15-step proof in approximately 2,003 CPU-seconds with retention of clause (107217). This 15-step proof is sharply different from the 15-step proof offered much earlier in this section. Oh, the wonders of mathematics and of automated reasoning! Indeed, rather than aiding the search for a proof by including three lemmas, their inclusion, although leading to an 11-step proof rather than a 15-step proof, resulted in far more computer time before the goal was reached. An examination of the just-cited proof shows that transitivity of betweenness was used, but not connectivity. So, yes, one more experiment is merited; indeed, some might naturally wonder how all would go if those two axioms, of the Tarski thirteen, were not available.

OTTER produced a 14-step forward proof in approximately 236 CPU-seconds with retention of clause (19465). Binary resolution was not used; indeed, ten of the fourteen steps were obtained with hyperresolution and four with paramodulation. And again charm is present: only six of Tarski's axioms are used. When and if the use of binary resolution will be necessary, as it appears to be required in a proof found in Section 8, awaits my study of deeper theorems in Tarskian geometry.

15. Emulation of Earlier Successes and More Proofs

OTTER's allowing one to prove many theorems in a single run is precisely what I almost demand, since I am an impatient researcher. Of course, the set of theorems to be considered in an experiment must share the axiom set being used—not necessarily using all of them, clearly. I now present an input file that proved almost all the theorems in Chapter 2 of the Szmielew book and some of the theorems in Chapter 3. I also list a superset of those theorems, shortly after I offer you an input file.

A Most Useful Input File

```
assign(max_weight,22).
assign(max_proofs,20).
set(hyper_res).
set(input_sos_first).
clear(order_hyper).
clear(print_kept).
set(para_into).
set(para_from).
set(ur_res).
% set(binary_res).
set(unit_deletion).

weight_list(pick_and_purge).
weight(f4(*(0),(0),*(0),*(0)),1).
```

```
end_of_list.

list(sos).
x = x.
ac1 != ac2.
T(ac1,ac2,ac3).
E(ac2,ac3,a4,a5).
E(b1,b2,b3,b4).
T(a7,a8,a9).
T(aa,ab,ac).
T(aa1,ab1,ac1).
T(da,db,dd).
T(db,dc,dd).
T(ba,bb,bc).
T(ba1,bb1,bc1).
T(a7,a8,a9).
E(aa,ab,aa1,ab1).
E(ab,ac,ab1,ac1).
E(eb,ec,eb1,ec1).
E(ba,bb,ba1,bb1).
E(bb,bc,bb1,bc1).
E(d,e,d3,d3).
E(b3,b4,b5,b6).
E(b1,b2,b3,b4).
T(da,db,dc).
T(db,da,dc).
T(a1,a2,a5).
T(a1,a4,a5).
T(a2,a3,a4).
T(A2,a6,a4).
T(a,b,c).
T(cba,cbb,cbc).
T(cbb,cbc,cbd).
T(fa,fb,fd).
T(fb,fc,fd).
T(ga,gb,gc).
T(ga,gc,gd).
T(gga,ggb,ggc).
T(gga,ggc,ggd).
T(ggga,gggb,gggc).
T(gggb,gggc,gggd).
T(cba,cbb,cbc).
T(cbb,cbc,cbd).
cbb != cbc.
T(ggga,gggb,gggc).
T(gggb,gggc,gggd).
gggb != gggc.
T(gga,ggb,ggc).
T(gga,ggc,ggd).
T(ga,gb,gc).
T(ga,gc,gd).
T(fa,fb,fd).
T(fb,fc,fd).
```

```
T(a,b,c).
% following 20 clauses are translations of first 20 clauses from ch6 of 1987 book focusing
  on Tarski's 13-axiom system
-T(x,y,x) | (x = y).
-T(x,y,u) | -T(y,z,u) | T(x,y,z).
-T(x,y,z) | -T(x,y,u) | (x = y) | T(x,z,u) | T(x,u,z).
E(x,y,y,x).
-E(x,y,z,z) | (x = y).
-E(x,y,z,u) | -E(x,y,v,w) | E(z,u,v,w).
-T(x,v,u) | -T(y,u,z) | T(x,f1(v,x,y,z,u),y).
-T(x,v,u) | -T(y,u,z) | T(z,v,f1(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,z,f2(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,y,f3(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(f2(v6,x,y,z,u),v6,f3(v6,x,y,z,u)).
-E(x1,y1,x2,y2) | -E(y1,z1,y2,z2) | -E(x1,u1,x2,u2) | -E(y1,u1,y2,u2) | -T(x1,y1,z1) | -T(x2,y2,z2) |
  (x1 = y1) | E(z1,u1,z2,u2).
T(x,y,f4(x,y,u,v)).
E(y,f4(x,y,u,v),u,v).
-T(c1,c2,c3).
-T(c2,c3,c1).
-T(c3,c1,c2).
-E(x,u,x,v) | -E(y,u,y,v) | -E(z,u,z,v) | (u = v) | T(x,y,z) | T(y,z,x) | T(z,x,y).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | E(u1,y1,u1,f5(x1,y1,z1,x2,z2,u1)).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | T(x2,f5(x1,y1,z1,x2,z2,u1),z2).
% T(x,y,y).
% T(x,x,y).
% -T(x,y,z) | T(z,y,x).
% -T(x,y1,z) | -T(x,y2,z) | T(x,y1,y2) | T(x,y2,y1).
end_of_list.
list(sos).
T(a2,a3,a4).
T(a1,a2,a5).
T(a1,a4,a5).
T(A2,a6,a4).
end_of_list.

list(passive).
f4(ac1,ac2,a4,a5) != ac3 | $ANS(CHZ12).
-T(cba,cbb,cbd) | $ANS(CH372).
-T(ggga,gggc,gggd) | $ANS(CHZ371).
-T(gga,ggb,ggd) | $ANS(CHZ362).
-T(gb,gc,gd) | $ANS(CHZ361).
-T(ga,gb,gd) | $ANS(CHZ361a).
-T(fa,fc,fd) | $ANS(CHZ352).
-T(c,b,a) | $ANS(CHZ32).
can != cb | $ANS(CHZ34).
d != e | $ANS(CHZZZ).
-E(a7,a8,a7,a8) | $ANS(CHZ1).
-E(b1,b1,b2,b2) | $ANS(CHZ8).
-E(b1,b2,b4,b3) | $ANS(CHZ5).
-E(b1,b2,b5,b6) | $ANS(CHZ3).
-E(b2,b1,b3,b4) | $ANS(CHZ4).
-E(b3,b4,b1,b2) | $ANS(CHZ2).
```

```
-E(ba,bc,ba1,bc1) | $ANS(CHZ11).
-T(a1,a1,a2) | $ANS(CH2).
-T(a1,a2,a2) | $ANS(CH3).
-T(da,db,dc) | $ANS(CHZ351).
% -T(a1,a3,a5) | $ANS(CH1).
% -T(a2,a3,a6) | $ANS(CH5A).
% -T(a2,a6,a3) | $ANS(CH5B).
% -T(a4,a3,a6) | $ANS(CH5C).
% -T(a4,a6,a3) | $ANS(CH5D).
end_of_list.
```

Note that both transitivity and connectivity of betweenness are included in the preceding file.

Especially in view of the title of this section, you might naturally ask about the birth of the given input file. I borrowed (emulated) that which worked and which I found pleasing from the files relied upon in the preceding section. Specifically, with a few modifications to parameter values of the input file, cited near the end of the preceding section that reached its goal with retention of clause (107217), I produced an appropriate input file. An inspection of the just-given input file reveals the presence of quite a number of ground clauses, clauses without reliance on variables. Their presence is explained by the items found in list(passive), the theorems to prove. Those uncommented in the passive list that have, for example, CH2 in the ANS literal correspond to theorems from Szmielew's Chapter 3. In particular, the denial of a nonunit clause (corresponding to a target theorem) produces a set of unit clauses. I now offer, for convenience, a superset of theorems that were the targets, found in list(passive), taken from the Szmielew book. To aid you in connecting these theorems to the items found in the passive list of the just-given input file, I will add to each the corresponding notation that you will find in the input file.

```
E(x,y,x,y). % Satz 2.1, CHZ1
-E(xa,xb,xc,xd) | E(xc,xd,xa,xb). % Satz 2.2, CHZ2
-E(xa,xb,xc,xd) | -E(xc,xd,xe,xf) | E(xa,xb,xe,xf). % Satz 2.3, CHZ3
-E(xa,xb,xc,xd) | E(xb,xa,xc,xd). % Satz 2.4, CHZ4
-E(xa,xb,xc,xd) | E(xa,xb,xd,xc). % Satz 2.5, CHZ5
E(x,x,y,y). % Satz 2.8, CHZ8
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xb,xa1,xb1) | -E(xb,xc,xb1,xc1) | E(xa,xc,xa1,xc1).
% Satz 2.11, CHZ11
xq = xa | -T(xq,xa,u) | -E(xa,u,xc,xd) | ext(xq,xa,xc,xd) = u. % Satz 2.12, CHZ12
T(x,y,y). % Satz 3.1, CH3
-E(u,v,x,x) | u=v. % Not one of Szmielew's theorems but we proved it, CHZZ
-T(xa,xb,xc) | T(xc,xb,xa). % Satz 3.2., CH4
T(xa,xa,xb). % Satz 3.3, CH2
-T(xa,xb,xc) | -T(xb,xa,xc) | xa = xb. % Satz 3.4, CHZ34
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xb,xc). % Satz 3.51, CHZ351
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xc,xd). % Satz 3.52, CHZ352
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd). % Satz 3.61, CHZ361
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd). % Satz 3.61a, CHZ361a
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd). % Satz 3.71, CHZ71
alpha != beta. % related to Satz 3.14; easily provable if added to sst 3h.in.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd). % Satz 3.62, CHZ362
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd). % Satz 3.71
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd). % Satz 3.72, CH72
```

First, you quickly observe that 3.51 is an odd inclusion, for it is the clause, among Tarski's thirteen axioms, for transitivity of betweenness and, therefore, is not actually a target. Second, you note that, in list(passive), where a nonunit theorem is concerned, you just find the denial of the positive literal(s), with the remaining literals, which arise from the denial of the theorem, found in other lists. The use of this input file yielded many proofs (to be discussed shortly). When a nonunit theorem was proved, the last line of that proof (before the item that signaled completion with binary resolution) is often a ground unit clause. As noted, such last lines must not be adjoined in future runs as so-called lemmas, but they could clearly be used as resonators.

The use of the sought-after single input file, designed to find many proofs, led to the following proofs.

----> UNIT CONFLICT at 0.00 sec ----> 130 [binary,129.1,86.1] $ANS(CHZZZ).
Length of proof is 1. Level of proof is 1.
----> UNIT CONFLICT at 0.00 sec ----> 135 [binary,134.1,87.1] $ANS(CHZ1).
Length of proof is 1. Level of proof is 1.
----> UNIT CONFLICT at 0.00 sec ----> 137 [binary,136.1,91.1] $ANS(CHZ4).
Length of proof is 1. Level of proof is 1.
----> UNIT CONFLICT at 0.03 sec ----> 563 [binary,562.1,85.1] $ANS(CHZ34).
Length of proof is 2. Level of proof is 2.
----> UNIT CONFLICT at 0.07 sec ----> 1129 [binary,1128.1,92.1] $ANS(CHZ2).
Length of proof is 2. Level of proof is 2.
----> UNIT CONFLICT at 0.12 sec ----> 1205 [binary,1204.1,89.1] $ANS(CHZ5).
Length of proof is 3. Level of proof is 3.
----> UNIT CONFLICT at 0.12 sec ----> 1207 [binary,1206.1,90.1] $ANS(CHZ3).
Length of proof is 3. Level of proof is 3.
----> UNIT CONFLICT at 0.28 sec ----> 1984 [binary,1983.1,88.1] $ANS(CHZ8).
Length of proof is 2. Level of proof is 2.
----> UNIT CONFLICT at 0.28 sec ----> 1986 [binary,1985.1,95.1] $ANS(CH3).
Length of proof is 2. Level of proof is 2.
----> UNIT CONFLICT at 2.59 sec ----> 11009 [binary,11008.1,94.1] $ANS(CH2).
Length of proof is 6. Level of proof is 5.
----> UNIT CONFLICT at 3.60 sec ----> 17481 [binary,17480.1,83.1] $ANS(CHZ32).
Length of proof is 6. Level of proof is 5.
----> UNIT CONFLICT at 23.29 sec ----> 80083 [binary,80082.1,81.1] $ANS(CHZ361a).
Length of proof is 9. Level of proof is 6.
----> UNIT CONFLICT at 23.71 sec ----> 81223 [binary,81222.1,79.1] $ANS(CHZ362).
Length of proof is 9. Level of proof is 6.
----> UNIT CONFLICT at 951.40 sec ----> 281761 [binary,281760.1,82.1] $ANS(CHZ352).
Length of proof is 14. Level of proof is 8.
----> UNIT CONFLICT at 1006.63 sec ----> 299952 [binary,299951.1,80.1] $ANS(CHZ361).
Length of proof is 15. Level of proof is 9.
----> UNIT CONFLICT at 5704.82 sec ----> 904661 [binary,904660.1,76.1] $ANS(CHZ12).
Length of proof is 6. Level of proof is 5.

Of the targets found in list(passive) of this input file, three were not reached: CHZ11, CHZ71, and CH72. In other words, with a single run, sixteen theorems of interest were proved. Further and most pleasing, all the sixteen proofs are forward proofs. Binary resolution was not used, nor were any lemmas; indeed, just the thirteen Tarski axioms sufficed.

As the following experiment with its results illustrates, automated reasoning and, it seems, mathematics continue to offer surprises. In particular, among my experiments, I modified the Most Useful Input File by commenting out both transitivity and connectivity of betweenness with the goal of—odd though it might be—proving one or more of the three targets not yet reached. The vague notion behind this move was that those two axioms might be so fruitful that OTTER would never reach any of the three sought-after proofs. I remind you that McCune frequently omitted axioms that were dependent when he studied various areas of abstract algebra. And, as you more than suspect because of my inclusion of this bit of discussion, OTTER did eventually find a gem, the following.

A Sought-After Proof of CHZ11

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Sat Dec 15 07:56:48 2012
The command was "otter". The process ID is 2021.
----> UNIT CONFLICT at 141384.79 sec ----> 2334603 [binary,2334602.1,91.1] $ANS(CHZ11).

Length of proof is 32. Level of proof is 13.

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

11 [] T(ba,bb,bc).
12 [] T(ba1,bb1,bc1).
17 [] E(ba,bb,ba1,bb1).
18 [] E(bb,bc,bb1,bc1).
52 [] -T(x,y,x)|x=y.
53 [] E(x,y,y,x).
54 [] -E(x,y,z,z)|x=y.
55 [] -E(x,y,z,u)| -E(x,y,v,w)|E(z,u,v,w).
56 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
57 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
61 [] -E(x1,y1,x2,y2)| -E(y1,z1,y2,z2)| -E(x1,u1,x2,u2)| -E(y1,u1,y2,u2)| -T(x1,y1,z1)| -T(x2,y2,z2)|x1=y1|E(z1,u1,z2,u2).
62 [] T(x,y,f4(x,y,u,v)).
63 [] E(y,f4(x,y,u,v),u,v).
91 [] -E(ba,bc,ba1,bc1)|$ANS(CHZ11).
103 [hyper,55,53,53] E(x,y,x,y).
118 [hyper,55,18,53] E(bb1,bc1,bc,bb).
119 [hyper,55,17,53] E(ba1,bb1,bb,ba).
413 [hyper,63,54] x=f4(y,x,z,z).
475 [hyper,103,55,18] E(bb1,bc1,bb,bc).
476 [hyper,103,55,17] E(ba1,bb1,ba,bb).
496 [hyper,118,55,53] E(bc1,bb1,bc,bb).
497 [hyper,118,55,53] E(bc,bb,bc1,bb1).
498 [hyper,119,55,53] E(bb1,ba1,bb,ba).
499 [hyper,119,55,53] E(bb,ba,bb1,ba1).
615 [para_from,413.1.2,63.1.2] E(x,x,y,y).
617 [para_from,413.1.2,62.1.3] T(x,y,y).
830 [hyper,617,57,12] T(bc1,bb1,f1(bb1,ba1,x,bc1,bc1)).
831 [hyper,617,57,11] T(bc,bb,f1(bb,ba,x,bc,bc)).
883 [hyper,617,56,12] T(ba1,f1(bb1,ba1,x,bc1,bc1),x).
884 [hyper,617,56,11] T(ba,f1(bb,ba,x,bc,bc),x).
1141 [hyper,615,61,476,475,498,12,11] ba1=bb1|E(bc1,ba1,bc,ba).
1221 [hyper,615,61,17,18,499,11,12] ba=bb|E(bc,ba,bc1,ba1).
1287 [para_from,1141.1.2,496.1.2] E(bc1,ba1,bc,bb)|E(bc1,ba1,bc,ba).
1297 [hyper,1221,55,103] ba=bb|E(bc1,ba1,bc,ba).
14003 [hyper,883,52] ba1=f1(bb1,ba1,ba1,bc1,bc1).
14091 [para_from,14003.1.2,830.1.3] T(bc1,bb1,ba1).
14402 [hyper,884,52] ba=f1(bb,ba,ba,bc,bc).
14490 [para_from,14402.1.2,831.1.3] T(bc,bb,ba).
14522 [hyper,14490,61,497,499,615,18,14091] bc=bb|E(ba,bc,ba1,bc1).
14674 [hyper,14522,55,53] bc=bb|E(ba1,bc1,bc,ba).
14969 [hyper,14674,55,53] bc=bb|E(bc1,ba1,bc,ba).
15694 [para_into,14969.1.2,1297.1.2] bc=ba|E(bc1,ba1,bc,ba).

16401 [para_into,15694.1.1,14969.1.1] bb=ba|E(bc1,ba1,bc,ba).
2334496 [para_into,1287.1.4,16401.1.1] E(bc1,ba1,bc,ba).
2334519 [hyper,2334496,55,53] E(bc,ba,ba1,bc1).
2334602 [hyper,2334519,55,53] E(ba,bc,ba1,bc1).
2334603 [binary,2334602.1,91.1] $ANS(CHZ11).

The retention of more than 2 million clauses and the use of approximately 141,300 CPU-seconds to complete this 32-step proof provide a fine example of the value of waiting for a long, long time as the program tirelessly seeks to complete an assignment. Again you encounter that mystery with no answer at this time: When is it more effective to exclude axioms? Yes, intuition suggests that more axioms and, perhaps, additional lemmas, should in general lead to proofs in shorter time, the retention of fewer clauses, and the finding of proofs net yet found. By the way, that experiment did not prove either Z34 or CHZ361. (As noted much earlier, CH4, the symmetry of betweenness, was proved with Input File 1.)

CHZ371 and CH372 still remain unproven in this discussion. Of course, perhaps some modifications of the parameter values and options in the Most Useful Input File would lead to the two sought-after proofs. As an aside, I have heard disappointment expressed when OTTER does not find a desired proof. My response has typically been that, if you consult a fellow researcher without gaining what is needed, you do not from thence forward never consult that person again. Usually what I do with OTTER is try many approaches, some of which I have detailed earlier in this notebook. Rather than listing various experiments, I have selected one that did in fact produce a proof of CHZ371. The corresponding input file—which I discuss in the continued goal of providing you more and more knowledge about research—included the following five lemmas, placed in list(sos).

T(x,y,y).
T(x,x,y).
-T(x,y,z) | T(z,y,x).
E(x,y,x,y). % Satz 2.1
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd). % Satz 3.61.

These five lemmas, as you recognize, are among the theorems proved with the Most Useful Input File. The first three are, of course, the three lemmas in focus in Section 1. But, instead of presenting the input file that was used to obtain a proof of CHZ71, I think it more instructive to discuss an extension of the Most Useful Input File.

In particular, I took that input file and adjoined in list(sos) the five lemmas (lesser theorems), expecting, because of my knowledge of a proof of CHZ371, that all five would most likely be used. Well, such is not the case, as you see from the following proof.

A 4-Step Proof of CHZ371

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Sat Dec 22 09:24:59 2012
The command was "otter". The process ID is 17316.
----> UNIT CONFLICT at 250.51 sec ----> 374532 [binary,374531.1,83.1] $ANS(CHZ371).

Length of proof is 4. Level of proof is 4.

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

4 [] E(x,y,x,y).
5 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xb,xc,xd).
47 [] T(ggga,gggb,gggc).
48 [] T(gggb,gggc,gggd).
49 [] gggb!=gggc.

61 [] -E(x,y,z,z)|x=y.
68 [] -E(x1,y1,x2,y2)| -E(y1,z1,y2,z2)| -E(x1,u1,x2,u2)| -E(y1,u1,y2,u2)| -T(x1,y1,z1)| -T(x2,y2,z2)|x1=y1|E(z1,u1,z2,u2).
69 [] T(x,y,f4(x,y,u,v)).
70 [] E(y,f4(x,y,u,v),u,v).
83 [] -T(ggga,gggc,gggd)|$ANS(CHZ371).
871 [hyper,69,5,47] T(gggb,gggc,f4(ggga,gggc,x,y)).
4152 [hyper,871,68,4,70,4,4,48,unit_del,49] E(f4(ggga,gggc,gggc,gggd),x,gggd,x).
374446 [hyper,4152,61] f4(ggga,gggc,gggc,gggd)=gggd.
374531 [para_from,374446.1.1,69.1.3] T(ggga,gggc,gggd).
374532 [binary,374531.1,83.1] $ANS(CHZ371).

In contrast to the earlier proof of CHZ371 (which gave me the idea for the extension of the Most Useful Input File), whose proof relied on so many of the five lemmas, as you see, this proof relies on just the last two lemmas now cited explicitly.

E(x,y,x,y).
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd).

I had OTTER continue to use the input file that yielded the cited proof of CHZ371, expecting that nothing would occur in the context of a proof of CH372, but knowing that the cost of doing so was zero. I find that my impatience is well served by running many experiments at the same time. Therefore, so-to-speak in parallel, I turned to the Beeson approach, that of lemma adjunction. Specifically, I adjoined to list(sos) Satz 3.71, equivalently, CHZ371. The effectiveness of this move, of this incarnation of lemma adjunction, is nicely demonstrated with the following proof of CH372, partially annotated to highlight the use of lemmas

A 9-Step Proof of CH372, Relying on CHZ371

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Fri Dec 28 08:21:06 2012
The command was "otter". The process ID is 18187.
----> UNIT CONFLICT at 0.23 sec ----> 3362 [binary,3361.1,81.1] $ANS(CH372).

Length of proof is 9. Level of proof is 4.

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

2 [] -T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xc,xd). % CHZ371
4 [] -T(x,y,z) | T(z,y,x). % CH4, sym of between
5 [] E(x,y,x,y). % CHZ1
34 [] T(cba,cbb,cbc). % from denial of CH372
35 [] T(cbb,cbc,cbd). % from denial of CH372
46 [] cbb!=cbc. % from denial of CH372
60 [] -E(x,y,z,z)|x=y.
61 [] -E(x,y,z,u)| -E(x,y,v,w)|E(z,u,v,w).
69 [] E(y,f4(x,y,u,v),u,v).
81 [] -T(cba,cbb,cbd)|$ANS(CH372).
100 [hyper,34,4] T(cbc,cbb,cba).
101 [hyper,35,4] T(cbd,cbc,cbb).
122 [ur,60,46] -E(cbb,cbc,x,x).
950 [hyper,69,60] x=f4(y,x,z,z).
1848 [hyper,101,2,100] cbc=cbb|T(cbd,cbb,cba).
3214 [ur,122,61,5] -E(x,x,cbb,cbc).

3221 [para_from,950.1.2,69.1.2] E(x,x,y,y).
3315 [para_from,1848.1.1,3214.1.4,unit_del,3221] T(cbd,cbb,cba).
3361 [hyper,3315,4] T(cba,cbb,cbd).
3362 [binary,3361.1,81.1] $ANS(CH372).

Precisely as I was writing this material, I discovered a lemma that Beeson had proved, taken from Szmielew, that, apparently, I never proved. I offer it to you as a challenge in case you wish to pause and play the game. You are asked to prove that E(x,y,y,x) is a theorem in Tarskian geometry, an intuitively obvious fact about equidistance. Since, from a cursory examination of my files, I cannot find a proof, I have no strong opinion about which, if any, lemmas to use, lemmas that you may have already proved, for example. It would be nice to see a proof of this symmetry of equidistance based solely on the thirteen Tarski axioms (given in his 1959 system), without including any other results in Tarskian geoemwtry.

Now I return to the discussion of proving CH372, with a focus on a result that is, for those who enjoy huge numbers, most entertaining. As I noted a bit earlier, the input file that produced the cited proof of CHZ371 was allowed to continue to be in use, running all the time I have been writing the preceding. From basketball: No harm, no foul. I expected nothing but focused on other experiments and on the writing of this notebook. Well, rather than nothing, you are now presented with the following proof.

A Startling Proof of CH372

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Sat Dec 22 09:24:59 2012
The command was "otter". The process ID is 17316.
----> UNIT CONFLICT at 111285.77 sec ----> 5546609 [binary,5546608.1,82.1] $ANS(CH372).

Length of proof is 9. Level of proof is 8.

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

3 [] -T(x,y,z)|T(z,y,x). % CH4, sym of between
4 [] E(x,y,x,y). % CHZ1
5 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xb,xc,xd). % CHZ361
44 [] T(cba,cbb,cbc). % from denial of CH372
45 [] T(cbb,cbc,cbd). % from denial of CH372
46 [] cbb!=cbc. % from denial of CH372
57 [] -T(x,y,x)|x=y.
61 [] -E(x,y,z,z)|x=y.
63 [] -T(x,v,u)| -T(y,u,z)|T(x,f1(v,x,y,z,u),y).
64 [] -T(x,v,u)| -T(y,u,z)|T(z,v,f1(v,x,y,z,u)).
68 [] -E(x1,y1,x2,y2)| -E(y1,z1,y2,z2)| -E(x1,u1,x2,u2)| -E(y1,u1,y2,u2)| -T(x1,y1,z1)|
 -T(x2,y2,z2)|x1=y1|E(z1,u1,z2,u2).
69 [] T(x,y,f4(x,y,u,v)).
70 [] E(y,f4(x,y,u,v),u,v).
82 [] -T(cba,cbb,cbd)|$ANS(CH372).
873 [hyper,69,5,44] T(cbb,cbc,f4(cba,cbc,x,y)).
4261 [hyper,873,68,4,70,4,4,45,unit_del,46] E(f4(cba,cbc,cbc,cbd),x,cbd,x).
379633 [hyper,4261,61] f4(cba,cbc,cbc,cbd)=cbd.
379718 [para_from,379633.1.1,69.1.3] T(cba,cbc,cbd).
381141 [hyper,379718,64,44] T(cbd,cbb,f1(cbb,cba,cba,cbd,cbc)).
381309 [hyper,379718,63,44] T(cba,f1(cbb,cba,cba,cbd,cbc),cba).
5542561 [hyper,381309,57] cba=f1(cbb,cba,cba,cbd,cbc).
5543626 [para_from,5542561.1.2,381141.1.3] T(cbd,cbb,cba).

5546608 [hyper,5543626,3] T(cba,cbb,cbd).
5546609 [binary,5546608.1,82.1] $ANS(CH372).

To quote J Moore of the Boyer-Moore brilliant work in program verification, how robust OTTER is, retaining more than 5.5 million new conclusions before the proof was completed. And, how marvelous, you have two 9-step proofs, but how they do differ! You might note that, in lieu of access to CHZ371, OTTER turned to CHZ361 and reliance on Tarski axioms not used when CHZ371 was in play.

Now to return to a theorem in focus early in this notebook. Have you, on your own, found a proof of the four-point theorem, suggested by Overbeek? Have you tried and tried, as I have throughout this writing, sought a proof, and with no success? As you recall, the theorem, though unproved at the time, was used at some points to prove other theorems, specifically, the five-point theorem. After all, intuitively at least, it was clearly true. The question that was present implicitly asked how to prove the four-point theorem, especially from the thirteen Tarski axioms with no assistance from lemmas. Well, I now officially thank Beeson, and his indispensable aid in the study of the four-point theorem, for providing me with what was sufficient to obtain a proof. He has an advantage over various researchers, including me; indeed, I am almost certain he is comfortable thinking in terms of Tarskian geometry. He suggested that I define a point, q, that extends the line involving a2 and a4, in the following manner.

q = f4(a4,a2,a4,a2).

The axiom that is pertinent, I believe, is the following.

axiom of segment construction (two clauses):
T(x,y,f4(x,y,u,v)).
E(y,f4(x,y,u,v),u,v).

You might well wish to see the set of clauses that I then used to rely on Beeson's suggestion, the following.

% following is for the four-point theorem.
q = f4(a4,a2,a4,a2).
T(a2,a3,a4).
T(a2,a6,a4).
-T(a2,a3,a6) | $ANS(CH5A).
-T(a2,a6,a3) | $ANS(CH5B).
-T(a3,a6,a4) | $ANS(CH5c).
-T(a6,a3,a4) | $ANS(CH5d).

The reason for including the variations that focus on different combinations is that I could not predict the direction OTTER would take. For example, would a2 play the key role, or would a4? I included those famous three lemmas you visited in the beginning, the following.

-T(x,y,z) | T(z,y,x).
T(x,y,y).
T(x,x,y).

I have not succeeded in producing a proof of the four-point theorem without these three lemmas. I present that task to you as a challenge. As you will see in the following proof OTTER found, the third of the three lemmas was in fact not used. You will also see how vital q is to the proof, appearing in nineteen lines. And you will see, as indicated with the last line, that the proof is bidirectional.

A Bidirectional Proof of the Four-Point Theorem

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Thu Dec 13 11:23:50 2012
The command was "otter". The process ID is 17686.
----> UNIT CONFLICT at 0.18 sec ----> 1495 [binary,1494.1,955.1] $ANS(CH5B)|$ANS(CH5A).

Length of proof is 26. Level of proof is 10.

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

1 [] q=f4(a4,a2,a4,a2).
2 [] T(a2,a3,a4).
3 [] T(a2,a6,a4).
4 [] -T(a2,a3,a6)|$ANS(CH5A).
5 [] -T(a2,a6,a3)|$ANS(CH5B).
10 [] -T(x,y,u)| -T(y,z,u)|T(x,y,z).
11 [] -T(x,y,z)| -T(x,y,u)|x=y|T(x,z,u)|T(x,u,z).
12 [] E(x,y,y,x).
13 [] -E(x,y,z,z)|x=y.
14 [] -E(x,y,z,u)| -E(x,y,v,w)|E(z,u,v,w).
21 [] T(x,y,f4(x,y,u,v)).
22 [] E(y,f4(x,y,u,v),u,v).
26 [] -T(x,y,z)|T(z,y,x).
27 [] T(x,y,y).
45 [binary,10.2,3.1] -T(x,a2,a4)|T(x,a2,a6).
46 [binary,10.2,2.1] -T(x,a2,a4)|T(x,a2,a3).
59 [binary,14.3,13.1] -E(x,y,z,u)| -E(x,y,v,v)|z=u.
60 [hyper,14,12,12] E(x,y,x,y).
83 [para_into,21.1.3,1.1.2] T(a4,a2,q).
87 [para_into,22.1.2,1.1.2] E(a2,q,a4,a2).
98 [binary,26.2,5.1] -T(a3,a6,a2)|$ANS(CH5B).
99 [binary,26.2,4.1] -T(a6,a3,a2)|$ANS(CH5A).
295 [binary,83.1,26.1] T(q,a2,a4).
395 [ur,10,27,99] $ANS(CH5A)| -T(a3,a2,a3).
520 [binary,295.1,46.1] T(q,a2,a3).
521 [binary,295.1,45.1] T(q,a2,a6).
581 [ur,10,27,395] $ANS(CH5A)| -T(a2,a3,a2).
625 [binary,520.1,26.1] T(a3,a2,q).
665 [binary,521.1,26.1] T(a6,a2,q).
817 [ur,10,625,581] -T(a2,a3,q)|$ANS(CH5A).
818 [ur,10,625,99] -T(a6,a3,q)|$ANS(CH5A).
870 [ur,10,665,98] -T(a3,a6,q)|$ANS(CH5B).
955 [binary,817.1,26.2] $ANS(CH5A)| -T(q,a3,a2).
964 [binary,818.1,26.2] $ANS(CH5A)| -T(q,a3,a6).
980 [binary,870.1,26.2] $ANS(CH5B)| -T(q,a6,a3).
1183 [ur,11,520,521,964,980] $ANS(CH5B)|q=a2|$ANS(CH5A).
1204 [para_from,1183.1.2,2.1.1] T(q,a3,a4)|$ANS(CH5B)|$ANS(CH5A).
1407 [para_into,87.1.1,1183.1.2] E(q,q,a4,a2)|$ANS(CH5B)|$ANS(CH5A).
1481 [hyper,59,1407,60] $ANS(CH5B)|$ANS(CH5A)|a4=a2.
1494 [para_from,1481.1.1,1204.1.3,factor_simp,factor_simp] T(q,a3,a2)
 $ANS(CH5B)|$ANS(CH5A).
1495 [binary,1494.1,955.1] $ANS(CH5B)|$ANS(CH5A).

As for two of the questions that naturally arise, the answer to the first is no, I have not been able to find a forward proof yet; and another challenge is issued. The answer to the second is yes, I have a proof that avoids the use of binary resolution, the following.

A Proof of the Four-Point Theorem Avoiding Binary Resolution

----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Sat Dec 29 09:52:40 2012
The command was "otter". The process ID is 23428.
----> UNIT CONFLICT at 0.38 sec ----> 1971 [binary,1970.1,922.1] $ANS(CH5A)|$ANS(CH5B).

Length of proof is 26. Level of proof is 11.

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

1 [] q=f4(a4,a2,a4,a2).
2 [] T(a2,a3,a4).
3 [] T(a2,a6,a4).
4 [] -T(a2,a3,a6)|$ANS(CH5A).
5 [] -T(a2,a6,a3)|$ANS(CH5B).
10 [] -T(x,y,u)| -T(y,z,u)|T(x,y,z).
11 [] -T(x,y,z)| -T(x,y,u)|x=y|T(x,z,u)|T(x,u,z).
12 [] E(x,y,y,x).
13 [] -E(x,y,z,z)|x=y.
14 [] -E(x,y,z,u)| -E(x,y,v,w)|E(z,u,v,w).
21 [] T(x,y,f4(x,y,u,v)).
22 [] E(y,f4(x,y,u,v),u,v).
26 [] -T(x,y,z)|T(z,y,x).
27 [] T(x,y,y).
56 [para_into,21.1.3,1.1.2] T(a4,a2,q).
58 [hyper,14,22,12] E(x,y,f4(z,u,x,y),u).
59 [hyper,13,22] x=f4(y,x,z,z).
60 [para_into,22.1.2,1.1.2] E(a2,q,a4,a2).
68 [ur,26,5] -T(a3,a6,a2)|$ANS(CH5B).
69 [ur,26,4] -T(a6,a3,a2)|$ANS(CH5A).
250 [hyper,26,56] T(q,a2,a4).
379 [ur,10,27,69] $ANS(CH5A)| -T(a3,a2,a3).
559 [hyper,10,250,3] T(q,a2,a6).
560 [hyper,10,250,2] T(q,a2,a3).
578 [ur,10,27,379] $ANS(CH5A)| -T(a2,a3,a2).
618 [hyper,26,559] T(a6,a2,q).
659 [hyper,26,560] T(a3,a2,q).
812 [ur,10,618,68] -T(a3,a6,q)|$ANS(CH5B).
865 [ur,10,659,578] -T(a2,a3,q)|$ANS(CH5A).
866 [ur,10,659,69] -T(a6,a3,q)|$ANS(CH5A).
916 [ur,26,812] $ANS(CH5B)| -T(q,a6,a3).
922 [ur,26,865] $ANS(CH5A)| -T(q,a3,a2).
930 [ur,26,866] $ANS(CH5A)| -T(q,a3,a6).
940 [para_from,59.1.2,58.1.3] E(x,x,y,y).
1064 [ur,11,559,560,916,930] $ANS(CH5A)|q=a2|$ANS(CH5B).
1074 [para_from,1064.1.1,60.1.2] E(a2,a2,a4,a2)|$ANS(CH5A)|$ANS(CH5B).
1092 [para_from,1064.1.2,2.1.1] T(q,a3,a4)|$ANS(CH5A)|$ANS(CH5B).
1510 [hyper,14,1074,940] $ANS(CH5A)|$ANS(CH5B)|E(a4,a2,x,x).
1949 [hyper,13,1510] $ANS(CH5A)|$ANS(CH5B)|a4=a2.
1970 [para_from,1949.1.1,1092.1.3] T(q,a3,a2)|$ANS(CH5A)|$ANS(CH5B).
1971 [binary,1970.1,922.1] $ANS(CH5A)|$ANS(CH5B).

As for a comparison of the two 26-step proofs, nine of the deduced steps are not shared. In the second proof, UR-resolution is applied far more. However, a careful consideration of the axioms, of the inclusion of the definition of the point q, and of the theorem to be proved perhaps produces little or no surprise that both proofs share, for example, the following three deductions.

T(a4,a2,q).
T(a3,a2,q).
T(a6,a2,q).

In the 26-step proof of the four-point theorem, you find interesting deductions such as clause (56), which says that a2 is between a4 and q. Clause (60) asserts that the distance from a2 to q equals the distance from a4 to a2. And clause (560) says that a2 is between q and a3. Sometimes, observations like these, based on examining deductions in a proof, can provide insight in the context of how the reasoning progresses. A comparison of using and not using binary resolution may, just possibly, lead to something useful. Again, Overbeek would assert binary resolution need not be used.

Before ending this part of the journey and turning to a new country, one additional experiment merits discussion. In that experiment, the main goal was to prove two theorems from Szmielew that offer difficulty and resistance.

-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd). % Satz 3.71
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd). % Satz 3.72

Their resistance to proof caused me to make one change to a preceding experiment, namely, the adjoining of the following three lemmas.

T(x,y,y).
T(x,x,y).
-T(x,y,z) | T(z,y,x).

In the preceding experiment, which I will call R, one theorem, CHZ12, had been proved. In the current experiment, called R1, three theorems were proved: CHZ371, Ch372, and (again) CHZ12. At least superficially, the inclusion of the cited three lemmas made the difference, although the proof of CHZ12 in R1 was the same as it was in experiment R. Indeed, CHZ71 was proved in 306 CPU-seconds with the retention of clause (375739) and with a length of 8. The theorem CH372 was proved in 64,109 CPU-seconds with retention of clause (3498451) and with a proof of length 13. For those who enjoy extremes, the preceding citation is impressive; but the following establishes a real high, showing how robust OTTER is. The proof of CHZ12 that was presented was completed in 326,455 CPU-seconds with retention of clause (8544653) and with a proof length of 6, as before.

I drew two conclusions about the inclusion of the three lemmas in R1 when compared with R. First, their presence enabled the program to produce two proofs that had resisted completion, which is as it should be. Second, the inclusion of the three lemmas caused the program to consume much, much more CPU time. Indeed, the earlier 6-step proof of CHZ12 was found in 6,144 CPU-seconds with retention of clause (966407) and with a proof length of 6.

Axiom Modification

I am now about to take (in the next section) a significant change in direction, one that requires replacement of some of the Tarski axioms by others. You could pause here for a long, long time and decide to consider, perhaps again, the challenges offered earlier in this notebook. You could proceed with or without the aid of an automated reasoning program. You might prefer some other program than OTTER. For any of the challenges, you might or might not include lemmas or might (in effect) include, perhaps with resonators, a so-called outline of the proof being sought. You might, for your choice of inference rules, keep in mind the failures of those decades ago and contrast them with some of the successes cited so far. With OTTER, in particular, choices must be made about options and parameter values. And, not the least of the possible obstacles or difficulties to be faced, you are asked to choose from among the three types of proof: forward, backward, and bidirectional. Despite this perhaps-dark view of how to proceed, its darkness is now misleading, when compared with the early 1980s. To provide more encouraging evidence, I now turn to an approach, axiomatically, that is taken from Szmielew to a great extent. These axioms are those studied jointly by Beeson and me.

16. Other Axioms, More Experiments

In place of the Tarski 13-axiom system offered in Section 1, one crucial change is made, one replacement. In place of the two clauses for Pasch, outer Pasch, two clauses for inner Pasch are included.

```
Outer Pasch's axiom (2 clauses):
% -T(x,v,u) | -T(y,u,z) | T(x,f1(v,x,y,z,u),y).
% -T(x,v,u) | -T(y,u,z) | T(z,v,f1(v,x,y,z,u)).
% A7, inner Pasch, two clauses, replacing preceding two for outer
-T(x,v,u) | -T(y,z,u) | T(v,f1(x,v,u,y,z),y).
-T(x,v,u) | -T(y,z,u) | T(z,f1(x,v,u,y,z),x).
```

For a small clue that might give you some insight regarding inner Pasch, first draw an equilateral triangle with a vertex at the top. Then remove the line at the bottom, and draw two additional lines, one from the bottom left to some point on the right side (perhaps the midpoint), and one from the bottom right to some point on the left side (perhaps the midpoint). Inner Pasch asserts the existence of a point that is the intersection of the two lines.

Certain experiments virtually demand to be made in the context of the modified axiom set, say, with the following input file that will remind you of A Most Useful Input file.

Another Most Useful Input File

```
assign(max_weight,22).
assign(max_proofs,20).
set(hyper_res).
set(input_sos_first).
clear(order_hyper).
clear(print_kept).
set(para_into).
set(para_from).
set(ur_res).
% set(binary_res).
set(unit_deletion).

weight_list(pick_and_purge).
weight(f4(*(0),(0),*(0),*(0)),1).
end_of_list.

list(sos).
x = x.
ac1 != ac2.
T(ac1,ac2,ac3).
E(ac2,ac3,a4,a5).
E(b1,b2,b3,b4).
T(a7,a8,a9).
T(aa,ab,ac).
T(aa1,ab1,ac1).
T(da,db,dd).
T(db,dc,dd).
T(ba,bb,bc).
T(ba1,bb1,bc1).
```

```
T(a7,a8,a9).
E(aa,ab,aa1,ab1).
E(ab,ac,ab1,ac1).
E(eb,ec,eb1,ec1).
E(ba,bb,ba1,bb1).
E(bb,bc,bb1,bc1).
E(d,e,d3,d3).
E(b3,b4,b5,b6).
E(b1,b2,b3,b4).
T(da,db,dc).
T(db,da,dc).
T(a1,a2,a5).
T(a1,a4,a5).
T(a2,a3,a4).
T(A2,a6,a4).
T(a,b,c).
T(cba,cbb,cbc).
T(cbb,cbc,cbd).
T(fa,fb,fd).
T(fb,fc,fd).
T(ga,gb,gc).
T(ga,gc,gd).
T(gga,ggb,ggc).
T(gga,ggc,ggd).
T(ggga,gggb,gggc).
T(gggb,gggc,gggd).
T(cba,cbb,cbc).
T(cbb,cbc,cbd).
cbb != cbc.
T(ggga,gggb,gggc).
T(gggb,gggc,gggd).
gggb != gggc.
T(gga,ggb,ggc).
T(gga,ggc,ggd).
T(ga,gb,gc).
T(ga,gc,gd).
T(fa,fb,fd).
T(fb,fc,fd).
T(a,b,c).
% following 20 are translations of first 20 from ch6 of book2 for Tarski
-T(x,y,x) | (x = y).
% -T(x,y,u) | -T(y,z,u) | T(x,y,z).
% -T(x,y,z) | -T(x,y,u) | (x = y) | T(x,z,u) | T(x,u,z).
E(x,y,y,x).
-E(x,y,z,z) | (x = y).
-E(x,y,z,u) | -E(x,y,v,w) | E(z,u,v,w).
% -T(x,v,u) | -T(y,u,z) | T(x,f1(v,x,y,z,u),y).
% -T(x,v,u) | -T(y,u,z) | T(z,v,f1(v,x,y,z,u)).
% A7, inner Pasch, two clauses, replacing preceding two for outer
-T(x,v,u) | -T(y,z,u) | T(v,f1(x,v,u,y,z),y).
-T(x,v,u) | -T(y,z,u) | T(z,f1(x,v,u,y,z),x).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,z,f2(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,y,f3(v,x,y,z,u)).
```

```
-T(x,u,v) | -T(y,u,z) | (x = u) | T(f2(v6,x,y,z,u),v6,f3(v6,x,y,z,u)).
-E(x1,y1,x2,y2) | -E(y1,z1,y2,z2) | -E(x1,u1,x2,u2) | -E(y1,u1,y2,u2) | -T(x1,y1,z1) |
  -T(x2,y2,z2) | (x1 = y1) | E(z1,u1,z2,u2).
T(x,y,f4(x,y,u,v)).
E(y,f4(x,y,u,v),u,v).
-T(c1,c2,c3).
-T(c2,c3,c1).
-T(c3,c1,c2).
-E(x,u,x,v) | -E(y,u,y,v) | -E(z,u,z,v) | (u = v) | T(x,y,z) | T(y,z,x) | T(z,x,y).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | E(u1,y1,u1,f5(x1,y1,z1,x2,z2,u1)).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | T(x2,f5(x1,y1,z1,x2,z2,u1),z2).
% T(x,y,y).
% T(x,x,y).
% -T(x,y,z) | T(z,y,x).
end_of_list.
list(sos).
T(a2,a3,a4).
T(a1,a2,a5).
T(a1,a4,a5).
T(A2,a6,a4).
end_of_list.

list(passive).
f4(ac1,ac2,a4,a5) != ac3 | $ANS(CHZ12).
-T(cba,cbb,cbd) | $ANS(CH372).
-T(ggga,gggc,gggd) | $ANS(CHZ371).
-T(gga,ggb,ggd) | $ANS(CHZ362).
-T(gb,gc,gd) | $ANS(CHZ361).
-T(ga,gb,gd) | $ANS(CHZ361a).
-T(fa,fc,fd) | $ANS(CHZ352).
-T(c,b,a) | $ANS(CHZ32).
can != cb | $ANS(CHZ34).
da != db | $ANS(CHZ34).
d != e | $ANS(CHZZZ).
-E(a7,a8,a7,a8) | $ANS(CHZ1).
-E(b1,b1,b2,b2) | $ANS(CHZ8).
-E(b1,b2,b4,b3) | $ANS(CHZ5).
-E(b1,b2,b5,b6) | $ANS(CHZ3).
-E(b2,b1,b3,b4) | $ANS(CHZ4).
-E(b3,b4,b1,b2) | $ANS(CHZ2).
-E(ba,bc,ba1,bc1) | $ANS(CHZ11).
-T(a1,a1,a2) | $ANS(CH2).
-T(a1,a2,a2) | $ANS(CH3).
-T(a9,a8,a7) | $ANS(CH4).
-T(da,db,dc) | $ANS(CHZ351).
% -T(a1,a3,a5) | $ANS(CH1).
% -T(a2,a3,a6) | $ANS(CH5A).
% -T(a2,a6,a3) | $ANS(CH5B).
% -T(a4,a3,a6) | $ANS(CH5C).
% -T(a4,a6,a3) | $ANS(CH5D).
end_of_list.
```

If you compare this input file with its earlier-offered cousin, in addition to replacing outer Pasch with inner, you will see that two clauses, for transitivity and connectivity of betweenness, have now been commented out.

-T(x,y,u) | -T(y,z,u) | T(x,y,z).
-T(x,y,z) | -T(x,y,u) | (x = y) | T(x,z,u) | T(x,u,z).

These two properties (in the presence of inner Pasch replacing outer Pasch), transitivity and connectivity of betweenness, can be proved to be dependent on the remaining items not commented out. (In contrast, with outer Pasch in place of inner Pasch, connectivity of betweenness is dependent, but transitivity of betweenness is not dependent.) I plan to include the appropriate proofs of the two dependencies, in the presence of inner Pasch. Now, if you wish yet additional challenges, you might try to prove the two dependencies on your own.

(For the careful and curious reader and the reader who wondered about earlier remarks about "thought-to-be dependent", you see that if you remove the axioms of transitivity and connectivity from the system relying on inner Pasch, because the two are dependent, no theorems are lost; but, since transitivity is not dependent when inner Pasch is replaced by outer Pasch, the removal of the two weakens the resulting axiom system. For greater clarity, until essentially this point in the development of this notebook, as so often noted in earlier sections, I thought, erroneously, that transitivity of betweenness was dependent in the 1959 system that relies on outer Pasch.) This latest input file also includes the three now-favorite lemmas, commented out, but to be featured in later experiments with the axiom system that relies on inner Pasch. (As an aside, I note that when Beeson and I made our study—a study that in fact continues—rather than the function f1, we used the function ip. A cursory examination suggests, and I do not know why, that the use of ip rather than f1 has OTTER find proofs a bit faster.)

Similar to what you might expect, I took the input file, preventing the program from relying on two dependent properties that have just been cited, and asked OTTER to prove a large number of theorems selected from Szmielew. The symmetry of betweenness is one of the theorems to be proved, whereas it was not considered with the Most Useful Input File. Also, the presence of CHZ351 is to be ignored for the moment because, if proved, the proof would be irrelevant in that its positive form is present in the file. With the given input file, OTTER produced the following.

```
----> UNIT CONFLICT at 0.00 sec ----> 100 [binary,99.1,84.1] $ANS(CHZZZ).
Length of proof is 1. Level of proof is 1.
----> UNIT CONFLICT at 0.00 sec ----> 105 [binary,104.1,85.1] $ANS(CHZ1).
Length of proof is 1. Level of proof is 1.
----> UNIT CONFLICT at 0.00 sec ----> 107 [binary,106.1,89.1] $ANS(CHZ4).
Length of proof is 1. Level of proof is 1.
----> UNIT CONFLICT at 0.02 sec ----> 452 [binary,451.1,90.1] $ANS(CHZ2).
Length of proof is 2. Level of proof is 2.
----> UNIT CONFLICT at 0.07 sec ----> 528 [binary,527.1,87.1] $ANS(CHZ5).
Length of proof is 3. Level of proof is 3.
----> UNIT CONFLICT at 0.07 sec ----> 530 [binary,529.1,88.1] $ANS(CHZ3).
Length of proof is 3. Level of proof is 3.
----> UNIT CONFLICT at 0.19 sec ----> 596 [binary,595.1,86.1] $ANS(CHZ8).
Length of proof is 2. Level of proof is 2.
----> UNIT CONFLICT at 0.19 sec ----> 598 [binary,597.1,93.1] $ANS(CH3).
Length of proof is 2. Level of proof is 2.
----> UNIT CONFLICT at 0.54 sec ----> 2491 [binary,2490.1,83.1] $ANS(CHZ34).
Length of proof is 6. Level of proof is 6.
----> UNIT CONFLICT at 2.72 sec ----> 16798 [binary,16797.1,92.1] $ANS(CH2).
Length of proof is 5. Level of proof is 5.
----> UNIT CONFLICT at 3.64 sec ----> 19923 [binary,19922.1,81.1] $ANS(CHZ32).
Length of proof is 5. Level of proof is 5.
----> UNIT CONFLICT at 5.10 sec ----> 29787 [binary,29786.1,94.1] $ANS(CH4).
```

Length of proof is 5. Level of proof is 5.
----> UNIT CONFLICT at 612.38 sec ----> 371593 [binary,371592.1,78.1] $ANS(CHZ361).
Length of proof is 11. Level of proof is 8.
----> UNIT CONFLICT at 8909.32 sec ----> 1312760 [binary,1312759.1,74.1] $ANS(CHZ12).
Length of proof is 6. Level of proof is 5.

Compared with the results obtained with A Most Useful Input File, the only new proof is that for CH4, the symmetry of betweenness. In the other direction, the proofs not found here, but found with the earlier useful file, are for CHZ361a, CHZ362, and CHZ352. As for the use of the function ip rather than f1, for an example of time difference, with the ip notation, CHZ12 was proved in approximately 8,379 CPU-seconds, again with retention of clause (1312759). From a cursory perusal, the proofs are the same for the ip and f1 usages.

Naturally you wonder, as I did, what would result from permitting OTTER to employ the two dependent clauses. Well, nothing of interest took place. Indeed, a cursory examination shows that the proofs are the same. However, the presence of the two dependent axioms, neither of which (apparently) participated in a proof of interest, caused the program to take substantially longer to complete the sought-after proofs. (An important word of warning, especially for the individual whose fascination produces possible haste: In an input file, you must be sure that the ground clauses, those free of variables, adjoined to prove a theorem are not misused to prove another theorem in the same run, where the fault can be with the use of constants that are not distinct enough from theorem to theorem.)

The next experiment includes the three now well-known lemmas. You would correctly predict, based on the earlier discussion in this notebook, that their presence could slow the program so much that some theorems proved when the lemmas are not present would become out of reach. You would also suggest that perhaps one of the targets not yet reached would be proved. Well, yes, CHZ371 was proved. However, in addition to not proving CH2 and CH3, which are present among the three included lemmas, CHZ12 was also not proved. Therefore, by way of a small summary, still to be proved are CHZ361a, CHZ362, CH372, and CHZ11, from among the targets considered earlier.

In that I had now a proof of CHZ371—whose proof steps I may have used in the next experiment, but I cannot be certain—I chose to find a proof of CH372 because of their similarity, as seen in the following.

-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd). % Satz 3.71
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd). % Satz 3.72

This experiment, designed to begin filling in missing pieces by seeking a proof of CH372, led to the discovery of a small amount of gold, as seen with the following.

----> UNIT CONFLICT at 0.37 sec ----> 4053 [binary,4052.1,80.1] $ANS(CH372).
Length of proof is 8. Level of proof is 5.
----> UNIT CONFLICT at 0.38 sec ----> 4243 [binary,4242.1,84.1] $ANS(CHZ361a).
Length of proof is 7. Level of proof is 5.
----> UNIT CONFLICT at 0.40 sec ----> 4429 [binary,4428.1,82.1] $ANS(CHZ362).
Length of proof is 7. Level of proof is 5.
----> UNIT CONFLICT at 0.41 sec ----> 4613 [binary,4612.1,85.1] $ANS(CHZ352).
Length of proof is 6. Level of proof is 5.

I was left at this point with the target of proving CHZ11.

When I looked back in this notebook to see what had occurred with this target, I realized that I could get a proof by modifying A Most Useful Input File. Besides commenting out the axioms of transitivity and connectivity of betweenness, the important moves were assigning the value 11 to max_weight and deleting perhaps twenty-one ground clauses that are irrelevant to proving CHZ11. That (first) proof, cited near the beginning of Section 3, that gem, was found only after the expenditure of more than 141,384 CPU-seconds. A 24-step proof was obtained in approximately 105 CPU-seconds, with retention of clause (8611). When the irrelevant ground clauses remained in the input, again CHZ11 was proved, but this time in approximately 758 CPU-seconds, with retention of clause (30323). As you would expect, the same 24-step proof was found, of course with different numbering.

A natural, and even pressing, question arises. With inner Pasch in place of outer Pasch, does one have a more useful and, in some important sense, a more powerful axiom system? The next section provides, perhaps, some insight in the context of this question.

17. Inner Pasch and Dependencies, More Experiments

You may have, all throughout the reading of this notebook, suspected that I was secretly after proofs of two dependencies, cited earlier in the context of betweenness, namely, transitivity and connectivity. You may have also conjectured that they have continually eluded me. You are indeed on target. Of course, as noted early in this notebook, I finally learned that transitivity of betweenness is *not* dependent on the axiom system in which outer Pasch is present, but connectivity is dependent. So, in the spirit of a seminar, I invite you to join me as I search for what might enable OTTER to produce the elusive proofs. The path at this point will focus on proving the two dependencies, rather than in the presence of outer Pasch, in the presence of inner Pasch. I will discuss different attacks that may, for some, add to insights for research in areas other than geometry. I already know, and you will learn shortly, of some possibly important items and aspects.

At this point in the expedition, to avoid writing an entire book, I will feature highlights of a long, long journey, leaving for you various challenges. You will be witnessing research as it occurred. My first goal was to prove transitivity of betweenness dependent on the other Tarski axioms, omitting connectivity, which eventually also became a target for proving dependency. The experiments that produced proofs of a number of theorems from Szmielew immediately came into play, those relying on a Tarski system that included inner Pasch, but not outer Pasch, and that avoided the use of both transitivity and connectivity of betweenness. Indeed, I adjoined the conclusion (so to speak) of each of the proved theorems, which included, among various others, as you will see in an input file I will supply, symmetry of betweenness and CHZ361, the following.

```
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd).  % Satz 3.61.
```

To revisit a tiny pedantic point, the negative unit clause (by negating a nonunit clause) that was used to complete a proof was, of course, not that which was included; the inclusion, as shown, was the entire nonunit clause that was negated to seek a proof.

In less than 7 CPU-seconds and the retention of less than 1,100 new conclusions, OTTER presented me with a 4-step proof of the dependence of transitivity of betweenness in the presence of inner Pasch. Just two items were used from the input, other than those resulting from the denial of the theorem, namely, CH4 (symmetry of betweenness) and CHZ361. To be careful, or to yield to paranoia, I verified that the proofs of these two theorems were obtained without the use of either transitivity or connectivity of betweenness. I now offer the input file that was used and the proof found with its use.

A Key Input File Focusing on Inner Pasch and Transitivity

```
assign(max_weight,23).
assign(max_proofs,10).
assign(max_distinct_vars,4).
assign(pick_given_ratio,4).
set(hyper_res).
set(input_sos_first).
clear(order_hyper).
clear(print_kept).
set(para_into).
set(para_from).
set(ur_res).
set(binary_res).
set(unit_deletion).
```

```
weight_list(pick_and_purge).
weight(f4(*(0),(0),*(0),*(0)),1).
end_of_list.

list(sos).
T(aaa,aab,aad).
T(aab,aac,aad).
% IFS(hjha,hjhb,hjhc,hjhd,hjha1,hjhb1,hjhc1,hjhd1).
% -E(hjhb,hjhd,hjhb1,hjhd1) | $ANS(CHZ42).
x = x.
% following 20 are translations of first 20 from ch6 of book2 for Tarski
-T(x,y,x) | (x = y).
% -T(x,y,u) | -T(y,z,u) | T(x,y,z).
% -T(x,y,z) | -T(x,y,u) | (x = y) | T(x,z,u) | T(x,u,z).
E(x,y,y,x).
-E(x,y,z,z) | (x = y).
-E(x,y,z,u) | -E(x,y,v,w) | E(z,u,v,w).
% -T(x,v,u) | -T(y,u,z) | T(x,f1(v,x,y,z,u),y).
% -T(x,v,u) | -T(y,u,z) | T(z,v,f1(v,x,y,z,u)).
% A7, inner Pasch, two clauses, replacing preceding two for outer
-T(x,v,u) | -T(y,z,u) | T(v,f1(x,v,u,y,z),y).
-T(x,v,u) | -T(y,z,u) | T(z,f1(x,v,u,y,z),x).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,z,f2(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,y,f3(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(f2(v6,x,y,z,u),v6,f3(v6,x,y,z,u)).
-E(x1,y1,x2,y2) | -E(y1,z1,y2,z2) | -E(x1,u1,x2,u2) | -E(y1,u1,y2,u2) |
  -T(x1,y1,z1) | -T(x2,y2,z2) | (x1 = y1) | E(z1,u1,z2,u2).
T(x,y,f4(x,y,u,v)).
E(y,f4(x,y,u,v),u,v).
-T(c1,c2,c3).
-T(c2,c3,c1).
-T(c3,c1,c2).
-E(x,u,x,v) | -E(y,u,y,v) | -E(z,u,z,v) | (u = v) | T(x,y,z) | T(y,z,x) | T(z,x,y).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | E(u1,y1,u1,f5(x1,y1,z1,x2,z2,u1)).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | T(x2,f5(x1,y1,z1,x2,z2,u1),z2).
end_of_list.

list(sos).
% Following proved so far, from out5s2q6dd5sss and ss2.
E(x,y,x,y).  % Satz 2.1
-E(xa,xb,xc,xd) | E(xc,xd,xa,xb).  % Satz 2.2
-E(xa,xb,xc,xd) | -E(xc,xd,xe,xf) | E(xa,xb,xe,xf).  % Satz 2.3
-E(xa,xb,xc,xd) | E(xb,xa,xc,xd).  % Satz 2.4
-E(xa,xb,xc,xd) | E(xa,xb,xd,xc).  % Satz 2.5
E(x,x,y,y).  % Satz 2.8
xq = xa | -T(xq,xa,u) | -E(xa,u,xc,xd) | ext(xq,xa,xc,xd) = u.    % Satz 2.12
T(x,y,y).    % Satz 3.1
-T(xa,xb,xc) | T(xc,xb,xa).  % Satz 3.2.
T(xa,xa,xb).  % Satz 3.3
-T(xa,xb,xc) | -T(xb,xa,xc) | xa = xb.  % Satz 3.4.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd).  % Satz 3.61.
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd).  % Satz 3.71
% following proved in temp.beeson.five.point.out5s2q6dd5sss4
```

```
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xc,xd).  % Satz 3.52.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd).  % Satz 3.61.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd).  % Satz 3.62.
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd).  % Satz 3.72
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xb,xa1,xb1) | -E(xb,xc,xb1,xc1) | E(xa,xc,xa1,xc1).  % Satz 2.11
end_of_list.

list(sos).
%  T(a2,a3,a4).
%  T(a1,a2,a5).
%  T(a1,a4,a5).
%  T(A2,a6,a4).
end_of_list.

list(passive).
-T(aaa,aab,aac)|$ANS(TR).
T(fa,fb,fd).
T(fb,fc,fd).
f4(ac1,ac2,a4,a5) != ac3 | $ANS(CHZ12).
-T(cba,cbb,cbd) | $ANS(CH372).
-T(ggga,gggc,gggd) | $ANS(CHZ371).
-T(gga,ggb,ggd) | $ANS(CHZ362).
-T(gb,gc,gd) | $ANS(CHZ361).
-T(ga,gb,gd) | $ANS(CHZ361a).
-T(fa,fc,fd) | $ANS(CHZ352).
%  -T(c,b,a) | $ANS(CHZ32).
%   can != cb | $ANS(CHZ34).
%  da != db | $ANS(CHZ34).
%  d != e | $ANS(CHZZZ).
%  -E(a7,a8,a7,a8) | $ANS(CHZ1).
%  -E(b1,b1,b2,b2) | $ANS(CHZ8).
%  -E(b1,b2,b4,b3) | $ANS(CHZ5).
%  -E(b1,b2,b5,b6) | $ANS(CHZ3).
%  -E(b2,b1,b3,b4) | $ANS(CHZ4).
%  -E(b3,b4,b1,b2) | $ANS(CHZ2).
%  -E(ba,bc,ba1,bc1) | $ANS(CHZ11).
%  -T(a1,a1,a2) | $ANS(CH2).
%  -T(a1,a2,a2) | $ANS(CH3).
%  -T(a9,a8,a7) | $ANS(CH4).
%  -T(da,db,dc) | $ANS(CHZ351).
%  %  -T(a1,a3,a5) | $ANS(CH1).
%  %  -T(a2,a3,a6) | $ANS(CH5A).
%  %  -T(a2,a6,a3) | $ANS(CH5B).
%  %  -T(a4,a3,a6) | $ANS(CH5C).
%  %  -T(a4,a6,a3) | $ANS(CH5D).
end_of_list.
```

A 4-Step Proof of Transitivity of Betweenness in the Presence of Inner Pasch

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Thu Feb  7 09:13:06 2013
```

```
The command was "otter".  The process ID is 30324.
----> UNIT CONFLICT at   0.12 sec ----> 785 [binary,784.1,40.1] $ANS(TR).

Length of proof is 4.  Level of proof is 3.

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

1 [] T(aaa,aab,aad).
2 [] T(aab,aac,aad).
30 [] -T(xa,xb,xc)|T(xc,xb,xa).
33 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xb,xc,xd).
40 [] -T(aaa,aab,aac)|$ANS(TR).
186 [binary,30.1,2.1] T(aad,aac,aab).
187 [binary,30.1,1.1] T(aad,aab,aaa).
453 [hyper,187,33,186] T(aac,aab,aaa).
784 [binary,453.1,30.1] T(aaa,aab,aac).
785 [binary,784.1,40.1] $ANS(TR).
```

The input file illustrates the use of lemma adjunction; indeed, when a lemma or theorem is used, adjoining it to list(usable) or list(sos) is in the spirit of learning and can be the difference between success and failure in the next experiment. That file also shows how, say, with Satz 2.11, you must adjoin the nonunit and not be content with the termination of a proof that uses just the negated literal. The proof nicely shows how the earlier theorems proved were used, namely, 3.61 and that for symmetry of betweenness. No other axioms of Tarski were used for this proof; but, of course, they were used in the proofs of the two cited theorems.

The story of my journey, fictitiously titled Traveling through Tarski, next offered three routes to follow: (1) experiments with other approaches to proving the theorem under discussion; (2) experiments to find a proof of transitivity in the presence of outer Pasch; or (3) experiments concerned with connectivity of betweenness in the presence of inner Pasch. I chose the first of the three paths, in part because it is the easiest to follow, as you will see.

The most obvious change to make was to avoid the use of binary resolution, as you have witnessed at various times in this notebook. The same 4-step proof of transitivity was produced. The main difference was the use of clause (541) rather than clause (784) to complete the proof; also, the CPU time was almost cut in half. Although the cited differences in this case are insignificant, when you tackle theorems that prove to be much harder to prove, you may find such citations to be of value.

Expecting that I might fail, I turned to another experiment, another approach, to seeking a proof of transitivity. The reason for my expectation was that I decided to avoid using any theorems I had proved with the use of inner Pasch. In other words, I asked OTTER to prove transitivity from the remaining Tarski axioms, including inner Pasch, but excluding connectivity of betweenness. I allowed binary resolution to be used and chose to rely on Veroff's hints strategy, as you see in the following input file.

An Input File Relying on Hints

```
assign(max_weight,23).
assign(max_proofs,10).
assign(max_distinct_vars,4).
assign(pick_given_ratio,4).
assign(bsub_hint_wt,1).
set(keep_hint_subsumers).
set(hyper_res).
set(input_sos_first).
clear(order_hyper).
```

```
clear(print_kept).
set(para_into).
set(para_from).
set(ur_res).
set(binary_res).
set(unit_deletion).

weight_list(pick_and_purge).
weight(f4(*(0),(0),*(0),*(0)),1).
end_of_list.

list(sos).
T(aaa,aab,aad).
T(aab,aac,aad).
IFS(hjha,hjhb,hjhc,hjhd,hjha1,hjhb1,hjhc1,hjhd1).
x = x.
ac1 != ac2.
T(ac1,ac2,ac3).
T(a7,a8,a9).
T(aa,ab,ac).
T(aa1,ab1,ac1).
T(ba,bb,bc).
T(ba1,bb1,bc1).
T(a7,a8,a9).
E(ba,bb,ba1,bb1).
E(bb,bc,bb1,bc1).
T(a1,a2,a5).
T(a1,a4,a5).
T(a2,a3,a4).
T(A2,a6,a4).
T(a,b,c).
T(fa,fb,fd).
T(fb,fc,fd).
cbb != cbc.
gggb != gggc.
T(fa,fb,fd).
T(fb,fc,fd).
T(a,b,c).
%  following 20 are translations of first 20 from ch6 of book2 for Tarski
-T(x,y,x) | (x = y).
%  -T(x,y,u) | -T(y,z,u) | T(x,y,z).
%  -T(x,y,z) | -T(x,y,u) | (x = y) | T(x,z,u) | T(x,u,z).
E(x,y,y,x).
-E(x,y,z,z) | (x = y).
-E(x,y,z,u) | -E(x,y,v,w) | E(z,u,v,w).
%  -T(x,v,u) | -T(y,u,z) | T(x,f1(v,x,y,z,u),y).
%  -T(x,v,u) | -T(y,u,z) | T(z,v,f1(v,x,y,z,u)).
%  A7, inner Pasch, two clauses, replacing preceding two for outer
-T(x,v,u) | -T(y,z,u) | T(v,f1(x,v,u,y,z),y).
-T(x,v,u) | -T(y,z,u) | T(z,f1(x,v,u,y,z),x).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,z,f2(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(x,y,f3(v,x,y,z,u)).
-T(x,u,v) | -T(y,u,z) | (x = u) | T(f2(v6,x,y,z,u),v6,f3(v6,x,y,z,u)).
```

```
-E(x1,y1,x2,y2) | -E(y1,z1,y2,z2) | -E(x1,u1,x2,u2) | -E(y1,u1,y2,u2) |
  -T(x1,y1,z1) | -T(x2,y2,z2) | (x1 = y1) | E(z1,u1,z2,u2).
T(x,y,f4(x,y,u,v)).
E(y,f4(x,y,u,v),u,v).
-T(c1,c2,c3).
-T(c2,c3,c1).
-T(c3,c1,c2).
-E(x,u,x,v) | -E(y,u,y,v) | -E(z,u,z,v) | (u = v) | T(x,y,z) | T(y,z,x) | T(z,x,y).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | E(u1,y1,u1,f5(x1,y1,z1,x2,z2,u1)).
-E(u1,x1,u1,x2) | -E(u1,z1,u1,z2) | -T(u1,x1,z1) | -T(x1,y1,z1) | T(x2,f5(x1,y1,z1,x2,z2,u1),z2).
end_of_list.

list(sos).
% % Following sorted proofs steps, 65 of them, taken from proofs from sss8 and sss
% a8=f1(a8,a9,a9,a7,a8).
% ba1=bb1|E(bc1,ba1,bc,ba).
% ba1=bb1|E(bc,ba,bc1,ba1).
% ba=bb|E(bc,ba,bc1,ba1).
% bb1=ba1|E(bc,ba,bc1,ba1).
% bc1=ba1|E(bc,ba,bc1,ba1).
% bc1=bb1|E(ba1,bc1,ba,bc).
% bc1=bb1|E(ba1,bc1,bc,ba).
% bc1=bb1|E(bc,ba,bc1,ba1).
% b=f1(b,c,c,a,b).
% da=db.
% da=f1(db,da,dc,da,db).
% db=da.
% db=f1(db,da,db,x,db).
% db=f1(db,da,dc,da,db).
% d=e.
% f4(ac1,ac2,a4,a5)=ac3.
% gb=f1(gb,gc,gc,ga,gb).
% gc=f1(gc,gb,ga,gd,gc).
% gc=f1(gc,gd,gd,ga,gc).
% x=f1(x,x,x,y,x).
% x=f4(y,x,z,z).
% -T(x,y,z) | T(z,y,x).
% % Following proved so far, from out5s2q6dd5sss and ss2.
% % following proved in temp.beeson.five.point.out5s2q6dd5sss4
% % The following many clauses are Definition 4.1
% -IFS(xa,xb,xc,xd,za,zb,zc,zd) | T(xa,xb,xc).
% -IFS(xa,xb,xc,xd,za,zb,zc,zd) | T(za,zb,zc).
% -IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xa,xc,za,zc).
% -IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xb,xc,zb,zc).
% -IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xa,xd,za,zd).
% -IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xc,xd,zc,zd).
%  -T(xa,xb,xc) | -T(za,zb,zc) | -E(xa,xc,za,zc) | -E(xb,xc,zb,zc) |
   -E(xa,xd,za,zd) | -E(xc,xd,zc,zd) | IFS(xa,xb,xc,xd,za,zb,zc,zd).
% % Following 4 are definition 4.4 for n=3
% -E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa1,xa2,xb1,xb2).
% -E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa1,xa3,xb1,xb3).
% -E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa2,xa3,xb2,xb3).
% -E(xa1,xa2,xb1,xb2) | -E(xa1,xa3,xb1,xb3) | -E(xa2,xa3,xb2,xb3) | E3(xa1,xa2,xa3,xb1,xb2,xb3).
```

```
% jjja != jjjb.
% Col(jjja,jjjb,jjjc).
end_of_list.

list(sos).
end_of_list.

list(passive).
-T(aaa,aab,aac)|$ANS(TR).
end_of_list.

list(hints).
% Following sorted 14 from out5s2q6dd5sss, prove 3.2 and 361
b=f1(b,c,c,a,b).
gb=f1(gb,gc,gc,ga,gb).
gc=f1(gc,gb,ga,gd,gc).
gc=f1(gc,gd,gd,ga,gc).
T(b,f1(x,c,c,a,b),x).
T(c,b,a).
T(gb,f1(x,gc,gc,ga,gb),x).
T(gb,gc,gd).
T(gc,f1(gc,gb,ga,gd,gc),gc).
T(gc,f1(x,gd,gd,ga,gc),x).
T(gc,gb,ga).
T(gd,gc,ga).
T(x,y,y).
x=f4(y,x,z,z).
end_of_list.
```

The use of hints does not replace and is not equivalent to the use of resonators. Indeed, in contrast to a resonator that treats all variables as indistinguishable and, therefore, focuses on equivalence classes of formulas or equations, a hint treats the variables precisely as written and focuses on items that are identical to the hint (which, of course, includes alphabetic variants), subsume the hint, or are subsumed by the hint, depending on the included options. In general, the hints strategy works faster than does the resonance strategy, probably because far fewer deduced conclusions match a given hint than match a given resonator. The option set(keep_hint_subsumers) often is useful as well as the option assign(bsub_hint_wt,2) (or some small value). The hints you find in the input file were taken from earlier unsuccessful runs.

The experiment was, to me, somewhat of a surprise, which you can appreciate as you read the following.

A Proof of Transitivity with Hints and Inner Pasch

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Thu Feb  7 11:51:27 2013
The command was "otter".  The process ID is 31511.
----> UNIT CONFLICT at   1.39 sec ----> 8999 [binary,8998.1,45.1] $ANS(TR).

Length of proof is 8.  Level of proof is 5.

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

1 [] T(aaa,aab,aad).
```

```
2 [] T(aab,aac,aad).
27 [] -T(x,y,x)|x=y.
29 [] -E(x,y,z,z)|x=y.
31 [] -T(x,v,u)| -T(y,z,u)|T(v,f1(x,v,u,y,z),y).
32 [] -T(x,v,u)| -T(y,z,u)|T(z,f1(x,v,u,y,z),x).
37 [] T(x,y,f4(x,y,u,v)).
38 [] E(y,f4(x,y,u,v),u,v).
45 [] -T(aaa,aab,aac)|$ANS(TR).
111 [binary,31.3,27.1] -T(x,y,z)| -T(y,u,z)|y=f1(x,y,z,y,u).
134 [hyper,31,1,2] T(aab,f1(aaa,aab,aad,aab,aac),aab).
407 [binary,38.1,29.1] x=f4(y,x,z,z).
419 [para_from,407.1.2,37.1.3] T(x,y,y).
2974 [binary,134.1,27.1] aab=f1(aaa,aab,aad,aab,aac).
2999 [para_from,2974.1.2,32.3.2,unit_del,1,2] T(aac,aab,aaa).
6015 [hyper,111,2999,419] aab=f1(aac,aab,aaa,aab,aaa).
8998 [para_from,6015.1.2,32.3.2,unit_del,2999,419] T(aaa,aab,aac).
8999 [binary,8998.1,45.1] $ANS(TR).
```

Six axioms were used, as you see, in addition to the three clauses that arise from negating transitivity. In particular, both clauses for inner Pasch were relied upon.

The next called-for experiment was to repeat the preceding one but with an avoidance of binary resolution. Because of results already cited, you might, as I did, expect that the 8-step proof just presented would be duplicated. It was not, as you now see.

A Second Proof of Transitivity with Hints and Inner Pasch

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Thu Feb  7 11:57:33 2013
The command was "otter".  The process ID is 31575.
----> UNIT CONFLICT at   6.13 sec ----> 21269 [binary,21268.1,45.1] $ANS(TR).

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

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

1 [] T(aaa,aab,aad).
2 [] T(aab,aac,aad).
27 [] -T(x,y,x)|x=y.
29 [] -E(x,y,z,z)|x=y.
31 [] -T(x,v,u)| -T(y,z,u)|T(v,f1(x,v,u,y,z),y).
32 [] -T(x,v,u)| -T(y,z,u)|T(z,f1(x,v,u,y,z),x).
37 [] T(x,y,f4(x,y,u,v)).
38 [] E(y,f4(x,y,u,v),u,v).
45 [] -T(aaa,aab,aac)|$ANS(TR).
96 [hyper,31,1,2] T(aab,f1(aaa,aab,aad,aab,aac),aab).
234 [hyper,38,29] x=f4(y,x,z,z).
239 [para_from,234.1.2,37.1.3] T(x,y,y).
1442 [hyper,96,27] aab=f1(aaa,aab,aad,aab,aac).
1465 [para_from,1442.1.2,32.3.2,unit_del,1,2] T(aac,aab,aaa).
1493 [hyper,1465,32,239] T(aab,f1(x,aaa,aaa,aac,aab),x).
21121 [hyper,1493,27] aab=f1(aab,aaa,aaa,aac,aab).
21268 [para_from,21121.1.2,31.3.2,unit_del,239,1465] T(aaa,aab,aac).
```

21269 [binary,21268.1,45.1] $ANS(TR).

One last experiment with transitivity was made. I tried repeating the preceding experiment with but one change: I replaced inner Pasch with outer Pasch. Of course, no proof was found; after all, as I learned well after this experiment failed, transitivity of betweenness is independent in the Tarski axiom system that employs outer Pasch. You might indeed be puzzled at my inclusion of this result. Well, the experiment illustrates the type of research that is merited, if you do not know all the facts, as I did not at the time. In particular, axiom replacement to produce a related theorem is worth investigation.

18. Winning a Big Prize

In this section, you are about to embark on a long, long trip; a glass of good wine might indeed be in order. The nature of this prize is a proof, a proof of one of the most challenging theorems to prove in Tarskian geometry. The theorem says that in the presence of inner Pasch (in place of outer Pasch) and the other axioms of Tarskian geometry given in Section 1, you can prove that connectivity of betweenness is a dependent axiom. This theorem was one of Art Quaife's four challenge problems. Of course, you are welcome to accept this challenge before reading the material of this section. However, required of me is the historical fact that Beeson and I spent a long, long time before we succeeded with OTTER. To succeed, we relied in part on Szmielew, in part on Beeson alone, in part on me alone, and in part on the joint effort of Beeson and me. Before presenting the first approach that produced a proof, a brief discussion of proof checking versus proof finding is in order.

The work of Szmielew up through Chapter 12 had been thoroughly proof checked in Coq by Narboux et al.: quite a feat. In proof checking, as you may already know, you take the author's proof and show that it is indeed a proof. You may be forced, or your program may be forced, to insert a few steps here and there. In proof finding, you do not use the steps offered by the author, but, instead, present the theorem to be proved by your program, with possibly various so-called helpers (as will be discussed). Although proof checking can present numerous obstacles, in general, researchers assert, proof finding is far more difficult. If success occurs with proof finding, the resulting proof can, more than occasionally, be quite different from the proof of the author in focus. Now to the approach taken by Beeson and me, his idea.

The Beeson approach, from the beginning, was to prove one theorem after another and work our way through Szmielew from Chapter 1 on. As we proved each theorem, we would add that theorem in the next set of experiments, designed to move forward. Of course, as definitions in Szmielew occurred, they would also be adjoined. We relied essentially on the notation used by Szmielew, which, in this section, I will also. Therefore, in place of the function *f1* occurring in earlier sections, I will use the function *ip*. Also, in place of the function *f4*, I will use the function *ext*. The use of both *ip* and *ext* may aid you in keeping track of what is occurring in the approach and in proofs.

One of the early and key decisions to make regarded the type of proof to seek. As you have read earlier, even with a bidirectional or backward proof in hand, the completion of a forward proof can present a monumental obstacle. Recall that in a forward proof, the denial (of the theorem to be proved), which may consist of one or more clauses as in denying or negating a nonunit clause, is not used during the search to draw conclusions and is placed in list(passive). More accurately, some of the units that result from denying a nonunit clause often do participate in the drawing of conclusions, for example, those that are positive. As an illustration, if you were seeking a proof of transitivity of betweenness, the following clause, you would have three unit clauses that result from denying it, from its negation.

-T(x,y,u) | -T(y,z,u) | T(x,y,z).

They might be T(a,b,d), T(b,c,d), and -T(a,b,c). In a search for a forward proof, ordinarily the third of the cited three would be placed in list(passive); the other two, usually, would be placed in list(sos). So, often, if the goal is to *find* a proof, any proof, the choice that offers the greatest chance of success is that of a bidirectional search. Indeed, the negative unit from a unit-represented theorem or the negative unit clause (or clauses) from the denying of a nonunit clause can play a significant role. For our attempt to win the big prize, we therefore chose to seek a bidirectional proof.

The input file that produced the first proof consists of more than eight hundred lines. Rather than displaying this monster, I will discuss its important aspects, stating where Szmielew contributed, where Beeson alone contributed, and the like. More than a proof of connectivity, you will be treated in this section to other delicacies. Proof finding, and not proof checking, was our study. If all goes as planned, I intend to eventually discuss differences between the Szmielew proof and the OTTER proof or proofs. Yes, I will discuss various proofs, after a while. (Later I will offer A Powerful Input File for the study of the theorem focusing on the dependence of connectivity of betweenness in the presence of inner Pasch. Before I do, I will discuss highlights of an earlier input file, supplied by Beeson, a discussion that will contribute to your understanding of research.)

Beeson presented me with an OTTER input file, a file that did not reason from all of the Tarskian axioms, but just the following.

```
E(x,y,y,x).     % A1 from page 10 of sst equidistance-reflexive
-E(x,y,z,v) | -E(x,y,z2,v2) | E(z,v,z2,v2).   % A2 trans-equidistance
-E(x,y,z,z) | x=y.   % A3 identity-equidist
T(x,y,ext(x,y,w,v)).   % A4, first half segment-construc
E(y,ext(x,y,w,v),w,v).   % A4, second half segment-construc
-E(x,y,x1,y1) | -E(y,z,y1,z1) | -E(x,v,x1,v1) | -E(y,v,y1,v1) |
  -T(x,y,z) | -T(x1,y1,z1) | x=y | E(z,v,z1,v1).
  % A5 five-segment
-T(x,y,x) | x=y.   % A6 identity-between
%  A7, inner Pasch, two clauses.
-T(xa,xp,xc) | -T(xb,xq,xc) | T(xp,ip(xa,xp,xc,xb,xq),xb).
-T(xa,xp,xc) | -T(xb,xq,xc) | T(xq,ip(xa,xp,xc,xb,xq),xa).
-T(alpha,beta,gamma).  %A8,  three lines. lower-dimension
-T(beta,gamma,alpha).
-T(gamma,alpha,beta).
```

His note to me said no more axioms would be needed, probably concluded from a reading of Szmielew. I am not certain, for we did not touch on every item. So, as you can check, some of the axioms were not included. On the other hand, the file I received included twenty-six clauses corresponding to theorems we had proved, mostly by Beeson, theorems from Szmielew covered before she turned to a proof of connectivity of betweenness. Various other sets of clauses were present, some from Szmielew definitions; so you see already how Szmielew and Beeson each contributed to the approach that was taken. But Beeson's contribution went much further; indeed, he included the following clauses, defined points, because of being certain that a diagram must be represented.

```
c1 = ext(a,d,c,d).
d1 = ext(a,c,c,d).
b1 = ext(a,c1,c,b).
b2 = ext(a,d1,d,b).
e = ip(c1,d,b,d1,c).
p = ext(e,c,c,d).
r = ext(d1,c,e,c).
q = ext(p,r,p,r).
```

So, you ask, how did I contribute?

Although I cannot be positive, I believe the following to be essentially accurate. I made many experiments before the one in focus. From those, I chose so-called intermediate targets, targets that, if proved, would suggest progress was occurring. Those targets (or the negated part after denial), in negated form, were placed in list(passive). It may have been my choice to place all the items from which deductions were to be made in a list(sos), rather than placing some or many in a list(usable), a list consulted for inference-rule completion, as opposed to inference-rule initiation. I may be the one who chose the options and parameter values, the following, to be discussed shortly.

```
set(hyper_res).
```

```
set(para_into).
set(para_from).
set(binary_res).
set(ur_res).
set(order_history).
assign(report,5400).
assign(max_mem,840000).
clear(print_kept).
set(input_sos_first).



assign(max_weight,20).
assign(max_distinct_vars,4).
assign(pick_given_ratio,4).
assign(max_proofs,10).
assign(heat,2).
```

A large set of resonators was included, items chosen from proof steps of earlier proofs, a set perhaps supplied in part by me and in part by Beeson; thus, joint effort was seen. Finally, the following two clauses were placed in list(hot); the reason for this choice I cannot recall.

```
list(hot).
-E(x,y,x1,y1) | -E(y,z,y1,z1) | -E(x,v,x1,v1) | -E(y,v,y1,v1) |
  -T(x,y,z) | -T(x1,y1,z1) | x=y | E(z,v,z1,v1).  % A5
AFS(d1,c,r,p,p,c,e,d1).
end_of_list.
```

The assignment of the value 2 to the heat parameter was chosen to have OTTER heavily and recursively consult list(hot). When compared with an assignment of 0, which means the hot list is totally ignored, the assignment of 2 enables a program to make various deductions far sooner than it would otherwise; however, often the latter assignment costs much in CPU time. As it turned out, heat was not used much, as seen by a glance at the output file, and used not at all in the proof that was eventually found after a long, long time.

The "set" commands that the input file relied upon included those for the use of hyperresolution, UR-resolution, paramodulation (both from and into), and, yes, binary resolution. An additional command, set(intput_sos_first), was included to cause OTTER to choose, for inference-rule initiation, all the items in the initial set of support before choosing any deduce clause for that purpose. In general, a good move is to include this command when seeking a proof. Three assign commands were also chosen by me, in addition to the cited set commands, namely, that assigning a value of 20 to max_weight, that assigning a value of 4 to pick_given_ratio, and that assigning a value of 4 to max_distinct_vars. The assignment of values is in no way a science; instead, the art of assigning is based on experience and guesses. Here, the idea was to avoid drowning the program in newly kept conclusions and, at the same time, give it enough room so that a proof might be found. Success resulted, and a discussion of the proof is next in order. In that discussion, you will learn which theorems from Szmielew played a role, some of which I leave to you to prove as more challenges.

OTTER succeeded on January 11, 2013, returning to me a proof, of the dependence of connectivity of betweenness (based on inner Pasch), of length 104 (new conclusions) in approximately 17,126 CPU-seconds after retention of clause (429191). Only the following five Tarski axioms were used; of course, others were used in proving the theorems that will be cited from Szmielew.

```
1 [] E(x,y,y,x).   % A1 reflex equitistance
2 [] -E(x,y,z,v)| -E(x,y,z2,v2)|E(z,v,z2,v2).  % A2 transitivity equitistance
4 [] T(x,y,ext(x,y,w,v)).  % A4, first half segment construction
5 [] E(y,ext(x,y,w,v),w,v).   %  A4, second half
6 [] -E(x,y,x1,y1)| -E(y,z,y1,z1)| -E(x,v,x1,v1)| -E(y,v,y1,v1)| -T(x,y,z)| -T(x1,y1,z1)|x=y|E(z,v,z1,v1).
```

% A5 five-segment

Of the twenty-six theorems of Szmielew found in the more-than-800-step input file, twenty, the following, were relied upon, as well as certain definitions.

14 [] -E(xa,xb,xc,xd)|E(xc,xd,xa,xb). % Satz 2.2
15 [] -E(xa,xb,xc,xd)|E(xb,xa,xc,xd). % Satz 2.4
16 [] -E(xa,xb,xc,xd)| -E(xc,xd,xe,xf)|E(xa,xb,xe,xf). % Satz 2.3
17 [] -E(xa,xb,xc,xd)|E(xa,xb,xd,xc). % Satz 2.5
18 [] E(x,x,y,y). % Satz 2.8
19 [] -T(xa,xb,xc)| -T(xa1,xb1,xc1)| -E(xa,xb,xa1,xb1)| -E(xb,xc,xb1,xc1)|E(xa,xc,xa1,xc1). % Satz 2.11
21 [] T(x,y,y). % Satz 3.1
23 [] -T(xa,xb,xc)|T(xc,xb,xa). % Satz 3.2.
24 [] T(xa,xa,xb). % Satz 3.3
25 [] -T(xa,xb,xc)| -T(xb,xa,xc)|xa=xb. % Satz 3.4.
26 [] -T(xa,xb,xd)| -T(xb,xc,xd)|T(xa,xb,xc). % Satz 3.51
27 [] -T(xa,xb,xd)| -T(xb,xc,xd)|T(xa,xc,xd). % Satz 3.52.
28 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xb,xc,xd). % Satz 3.61.
32 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xa,xb,xd). % Satz 3.62.
33 [] -T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xc,xd). % Satz 3.71
34 [] -T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xb,xd). % Satz 3.72
36 [] -T(xa,xb,xc)| -T(xa1,xb1,xc1)| -E(xa,xc,xa1,xc1)| -E(xb,xc,xb1,xc1)|E(xa,xb,xa1,xb1). % Satz 4.3
43 [] -IFS(xa,xb,xc,xd,za,zb,zc,zd)|T(za,zb,zc). % Part of Definition 4.1
(3 lines here; there are 7 lines in the definition)
46 [] -IFS(xa,xb,xc,xd,za,zb,zc,zd)|E(xa,xd,za,zd).
47 [] -IFS(xa,xb,xc,xd,za,zb,zc,zd)|E(xc,xd,zc,zd).
52 [] -E(xa1,xa2,xb1,xb2)| -E(xa1,xa3,xb1,xb3)| -E(xa2,xa3,xb2,xb3)|
E3(xa1,xa2,xa3,xb1,xb2,xb3). % Part of definition 4.4
57 [] -T(xa,xb,xc)| -E3(xa,xb,xc,xa1,xb1,xc1)|T(xa1,xb1,xc1). % Satz 4.6
59 [] Col(xa,xb,xc)| -T(xa,xb,xc). % half of Definition 4.10 (three lines)
60 [] Col(xa,xb,xc)| -T(xb,xc,xa).
61 [] Col(xa,xb,xc)| -T(xc,xa,xb).
76 [] xa=xb| -Col(xa,xb,xc)| -E(xa,xp,xa,xq)| -E(xb,xp,xb,xq)|E(xc,xp,xc,xq). Satz 4.17
78 [] -T(xa,xc,xb)| -E(xa,xc,xa,xc1)| -E(xb,xc,xb,xc1)|xc=xc1. Satz 4.19
87 [] -AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1)|T(xa,xb,xc). % part of Definition 2.10

(I must give you more information, eventually, if you intend to try to prove the theorems, as challenges, from 4.2 on.) The following five lemmas were used, each supplied by Beeson, but which he wished to avoid using as input; he might have simply conjectured, correctly, that they would be important.

96 [] AFS(b,c,d1,c1,b1,c1,d,c). % A lemma: TARG1INT
97 [] E(c1,d1,c,d). % TARGINT or TARG2INT
98 [] IFS(d,e,d1,c,d,e,d1,c1). % TARG3INT
99 [] IFS(c,e,c1,d,c,e,c1,d1). % TARG4INT
100 [] E(e,c,e,c1). % TARG5INT

(As you proceed further in this journey focusing on connectivity, you will revisit these five lemmas, motivated by Beeson.) For the denial of connectivity, the proof relied on the following.

115 [] a!=b. % negation of Satz 5.1, connectivity, the first of five lines
116 [] T(a,b,c).
117 [] T(a,b,d).
118 [] -T(a,c,d)|$ANS(CON).
119 [] -T(a,d,c)|$ANS(CON).

Finally—so many thanks to Beeson, again—five of his defined points were relied upon, the following.

120 [] c1=ext(a,d,c,d). % auxiliary points used to prove Satz 5.1, the first of five lines

121 [] d1=ext(a,c,c,d).
125 [] p=ext(e,c,c,d).
126 [] r=ext(d1,c,e,c).
127 [] q=ext(p,r,p,r).

Three additional points are found in the input file, the following, but were not used in this first proof.

b1 = ext(a,c1,c,b).
b2 = ext(a,d1,d,b).
e = ip(c1,d,b,d1,c).

Of the 104 deduced steps, 45 were obtained with binary resolution, 23 with hyperresolution, 10 with UR-resolution, 25 with paramodulation, and 1 with factoring. The proof is bidirectional; and, exciting it may be, I still do not have a forward proof of connectivity of betweenness in the presence of inner Pasch, although I do have a number of proofs that are not forward. Yes, you are presented with quite a challenge: Find a forward proof. You might wish to wait until you read more about these experiments.

Although I was indeed pleased that after so many, many experiments, we had a proof of connectivity of betweenness, Beeson was not so pleased, which I learned weeks later. Indeed, he informed me that the five lemmas used in the proof had not actually been proved, and therefore we did not yet have a proof. In other words, more work was required, namely, the finding of five proofs, followed by the seeking of a proof of connectivity that did not include, in the input file, the five lemmas, called, respectively, RG1 through RG5.

My approach was not particularly inventive. Indeed, I simply (essentially) took the input file that has just been discussed, that which yielded the 104-step proof, and commented out the five lemmas, and inserted their negations as targets. OTTER found a 40-step proof of the first of the five, TARG1int, in approximately 473 CPU-seconds with retention of clause (109977). The proof was bidirectional, as was expected. To my annoyance, none of the other target lemmas was proved. Instead, the program produced, in total, eighteen proofs of the first lemma, stopping as it reached the assigned value to max_proofs. You see, one of the side effects that can, and often does, occur when seeking a bidirectional proof is the completion of one proof after another, of the same theorem. Perhaps contributing factors to the ease of finding a bidirectional proof as opposed to a forward proof are, first, such proofs so to speak meet in the middle because of reasoning forward and backward and, second, the level of such proofs is in general lower than that of the corresponding forward proof. On the other hand, when seeking a forward proof, this annoyance is in general avoided. Do you see what action to take to enable OTTER to avoid proving this first lemma again and, instead, seek one or more proofs of the other four target lemmas? Well, pause here if you wish to ponder the question. My approach was the following.

I took the input file that had found a proof (actually, eighteen proofs) of the first lemma and made two changes. I commented back in the positive form of the lemma and commented out its negation. In approximately 730 CPU-seconds, with retention of clause (148036), OTTER presented me with a 42-step proof of the second of the five lemmas. In fact, annoyance repeating itself, the program found eighteen proofs of this second lemma. Rather than immediately proceeding in the quest for proofs of Lemmas 3 through 5, I repeated the experiment that had yielded the proof of the second lemma, but with an assignment of the value 0 to heat, with the goal of finding proofs much faster. A good move indeed: With retention of clause (52976) and the use of approximately 57 CPU-seconds, OTTER completed a 47-step proof of the second lemma.

So, as you would accurately predict, I commented back in the second target and commented out its negation for the next run. The targets were all reached, in order, the fourth, the third, and the fifth. The respective proof lengths are 20, 16, and 20. Thus, consistent with Beeson's position that we had best not use the five lemmas in an input file whose goal was the proof of connectivity, which meant that proofs of each of the five must be in hand, this phase of the research was complete; we had the five proofs.

Before turning to the next big move, the next stop on this lengthy and most intriguing expedition, I pause to answer a question that might naturally arise at this point. How hard is it to find forward proofs of each of the five lemmas and thus avoid the side effect of focusing on bidirectional proofs? Well, earlier today, I did make a run to seek those five, possibly elusive, proofs. Indeed, one of the charming facets of notebook writing says that such experiments offer a break from writing and can proceed while more material is added to a section. so far, the first two (of the five) lemmas have been proved, proved with a forward proof.

So you must wish to know what I did with the proofs of the five lemmas, and wonder about the nature of the next key experiment. Before answering such questions, I feel it incumbent on me to adhere to history. In particular, I believe prompted by Beeson, I ran various experiments to verify a conjecture. Specifically, he and I had succeeded in finding proofs of diverse theorems by resorting to a case analysis. For example, where constants $d1$ and e were present, we had OTTER make two runs, one with $D1 = e$ and one with $D1 \mathrel{!}= e$. Quite often, both runs succeeded. Now, some researchers would be satisfied with the two proofs and maintain that the theorem in focus had been proved. Not Beeson. He intended that we find a way for OTTER to prove the theorem in a single run. I hypothesized that a possible approach, rather than running two cases, was to adjoin one new clause, namely, $D1 = e \mid D1 \mathrel{!}= e$, a tautology.

The theorem to prove for the experiment—and also the choice because we were after a proof that avoided the use of the five lemmas in the input—was, again, connectivity of betweenness in the presence of inner Pasch. OTTER succeeded, producing a 113-step proof, relying on thirty axioms and previously proved theorems, on eight defined points, on both clauses for inner Pasch, and, yes, on *two* tautologies. What was also of interest in the cited proof, among the already-proved theorems, was the presence of transitivity of betweenness. We will return to the significance of such an inclusion. The use of tautology adjunction was, for me, nostalgic; indeed, I had written about it as an approach to dealing with cases many decades ago, in the context of proving that subgroups of index 2 are normal. For possible research, Beeson (in effect) posed a problem: He thought, or hoped, that some way must exist for OTTER to avoid being forced to consider cases, a way other than tautology adjunction. And with this so-called interruption aside, the possible use of proofs of the five lemmas, RG1 through RG5, now takes center stage, and you learn that case analysis and tautology adjunction are in fact not needed to prove connectivity.

With the goal of obtaining a proof of connectivity of betweenness that does *not* rely on any of the five lemmas, RG1 through RG5, three moves were in order, as you will see from an input file I give shortly. First, I removed the five lemmas from the input file to be used. Second, I emphasized the use of hints rather than resonators, motivated by the wish to speed things up. Third, I placed in list(hints), in the file to be used, a sorted set of proof steps of proofs of the five lemmas, proofs obtained with cited experiments. After sorting to remove duplicates, I had a set of eighty-seven items to insert into a new list(hints).

A Powerful Input File for the Study of Connectivity in the Presence of Inner Pasch

```
set(hyper_res).
set(para_into).
set(para_from).
set(binary_res).
set(ur_res).
set(order_history).
assign(report,5400).
assign(max_mem,840000).
clear(print_kept).
%set(very_verbose).
set(input_sos_first).



assign(max_weight,11).
assign(max_distinct_vars,4).
assign(pick_given_ratio,4).
assign(max_proofs,10).
```

```
assign(heat,0).
assign(bsub_hint_wt,1).
set(keep_hint_subsumers).

weight_list(pick_and_purge).
end_of_list.



list(sos).
E(x,y,y,x).                       % A1 from page 10 of sst
-E(x,y,z,v) | -E(x,y,z2,v2) | E(z,v,z2,v2).   % A2
-E(x,y,z,z) | x=y.                % A3
T(x,y,ext(x,y,w,v)).              % A4, first half
E(y,ext(x,y,w,v),w,v).            % A4, second half
-E(x,y,x1,y1) | -E(y,z,y1,z1) | -E(x,v,x1,v1) | -E(y,v,y1,v1) |
  -T(x,y,z) | -T(x1,y1,z1) | x=y | E(z,v,z1,v1).  % A5
-T(x,y,x) | x=y.                  % A6
-T(xa,xp,xc) | -T(xb,xq,xc) | T(xp,ip(xa,xp,xc,xb,xq),xb).
-T(xa,xp,xc) | -T(xb,xq,xc) | T(xq,ip(xa,xp,xc,xb,xq),xa).
-T(alpha,beta,gamma).  %A8,  three lines.
-T(beta,gamma,alpha).
-T(gamma,alpha,beta).
E(x,y,x,y).  % Satz 2.1
-E(xa,xb,xc,xd) | E(xc,xd,xa,xb).  % Satz 2.2
-E(xa,xb,xc,xd) | E(xb,xa,xc,xd).  % Satz 2.4
-E(xa,xb,xc,xd) | -E(xc,xd,xe,xf) | E(xa,xb,xe,xf).  %  Satz 2.3
-E(xa,xb,xc,xd) | E(xa,xb,xd,xc).  % Satz 2.5
E(x,x,y,y).  % Satz 2.8
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xb,xa1,xb1) | -E(xb,xc,xb1,xc1) | E(xa,xc,xa1,xc1).  % Satz 2.11
xq = xa | -T(xq,xa,u) | -E(xa,u,xc,xd) | ext(xq,xa,xc,xd) = u.    % Satz 2.12
T(x,y,y).    % Satz 3.1
-E(u,v,x,x) | u=v.   % Not one of Szmielew's theorems but we proved it.
-T(xa,xb,xc) | T(xc,xb,xa).  % Satz 3.2.
T(xa,xa,xb).  % Satz 3.3
-T(xa,xb,xc) | -T(xb,xa,xc) | xa = xb.  % Satz 3.4.
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xb,xc).  % Satz 3.51.
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xc,xd).  % Satz 3.52.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd).  % Satz 3.61.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd).  % Satz 3.61.
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd).  % Satz 3.71
alpha != beta.   %related to Satz 3.14;  easily provable if added to sst 3h.in.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd).  % Satz 3.62.
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd).  % Satz 3.71
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd).  % Satz 3.72
-IFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xd,xb1,xd1).  % Satz 4.2
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xc,xa1,xc1) | -E(xb,xc,xb1,xc1) | E(xa,xb,xa1,xb1).  % Satz 4.3

alpha != beta.  % Satz 3.13
beta != gamma.
alpha != gamma.
T(xa,xb,ext(xa,xb,alpha,gamma)).   % Satz 3.14, first half
xb != ext(xa,xb,alpha,gamma).    % Satz 3.14, second half
```

```
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | T(xa,xb,xc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | T(za,zb,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xa,xc,za,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xb,xc,zb,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xa,xd,za,zd).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xc,xd,zc,zd).
-T(xa,xb,xc) | -T(za,zb,zc) | -E(xa,xc,za,zc) | -E(xb,xc,zb,zc) | -E(xa,xd,za,zd) |
  -E(xc,xd,zc,zd) | IFS(xa,xb,xc,xd,za,zb,zc,zd).

% Following 4 are definition 4.4 for n=3
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa1,xa2,xb1,xb2).
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa1,xa3,xb1,xb3).
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa2,xa3,xb2,xb3).
-E(xa1,xa2,xb1,xb2) | -E(xa1,xa3,xb1,xb3) | -E(xa2,xa3,xb2,xb3) | E3(xa1,xa2,xa3,xb1,xb2,xb3).
% Following three lines are Satz 4.5
-T(xa,xb,xc) | -E(xa,xc,xa1,xc1) | T(xa1,insert(xa,xb,xa1,xc1),xc1).
-T(xa,xb,xc) | -E(xa,xc,xa1,xc1) | E3(xa,xb,xc,xa1,insert(xa,xb,xa1,xc1),xc1).
insert(xa,xb,xa1,xc1) = ext(ext(xc1,xa1,alpha,gamma),xa1,xa,xb).
-E3(x,y,z,u,v,w) | E3(x,z,y,u,w,v).   % See sst4q.in, not in Szmielew
-T(xa,xb,xc) | -E3(xa,xb,xc,xa1,xb1,xc1) | T(xa1,xb1,xc1).  % Satz 4.6

% following is Definition 4.10
-Col(xa,xb,xc) | T(xa,xb,xc) | T(xb,xc,xa) | T(xc,xa,xb).
Col(xa,xb,xc) | -T(xa,xb,xc).
Col(xa,xb,xc) | -T(xb,xc,xa).
Col(xa,xb,xc) | -T(xc,xa,xb).
% Following are Satz 4.11
-Col(x,y,z) | Col(y,z,x).
-Col(x,y,z) | Col(z,x,y).
-Col(x,y,z) | Col(z,y,x).
-Col(x,y,z) | Col(y,x,z).
-Col(x,y,z) | Col(x,z,y).
% following is Satz 4.12
Col(x,x,y).
% following is Satz 4.13
-Col(xa,xb,xc) | - E3(xa,xb,xc,xa1,xb1,xc1) | Col(xa1,xb1,xc1).
% following is Satz 4.14
-Col(xa,xb,xc) | -E(xa,xb,xa1,xb1) | E3(xa,xb,xc,xa1,xb1,insert5(xa,xb,xc,xa1,xb1)).
% following is Definition 4.15
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | Col(xa,xb,xc).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E3(xa,xb,xc,xa1,xb1,xc1).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xa,xd,xa1,xd1).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xd,xb1,xd1).
-Col(xa,xb,xc) | - E3(xa,xb,xc,xa1,xb1,xc1) | - E(xa,xd,xa1,xd1) |
  -E(xb,xd,xb1,xd1) | FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1).
% Following is Satz 4.16
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | xa = xb | E(xc,xd,xc1,xd1).
% Following is Satz 4.17
xa = xb | -Col(xa,xb,xc) | -E(xa,xp,xa,xq) | -E(xb,xp,xb,xq) | E(xc,xp,xc,xq).
% Following is Satz 4.18
xa = xb | -Col(xa,xb,xc) | -E(xa,xc,xa,xc1) | -E(xb,xc,xb,xc1) | xc = xc1.
% Following is Satz 4.19
-T(xa,xc,xb) | -E(xa,xc,xa,xc1) | -E(xb,xc,xb,xc1) | xc = xc1.
```

```
% Following defines T5
T5(x,y,z,u,v) | -T(x,y,z) | -T(x,y,u) | -T(x,y,v) | -T(x,z,u) | -T(x,z,v) | -T(x,u,v).
-T5(x,y,z,u,v) | T(x,y,z).
-T5(x,y,z,u,v) | T(x,y,u).
-T5(x,y,z,u,v) | T(x,y,v).
-T5(x,y,z,u,v) | T(x,z,u).
-T5(x,y,z,u,v) | T(x,z,v).
-T5(x,y,z,u,v) | T(x,u,v).
% Following defines AFS
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xb,xa1,xb1) | -E(xb,xc,xb1,xc1) |
  -E(xa,xd,xa1,xd1) | -E(xb,xd,xb1,xd1) | AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | T(xa,xb,xc).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | T(xa1,xb1,xc1).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xa,xb,xa1,xb1).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xc,xb1,xc1).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xd,xb1,xd1).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1).
% Following proved in sst5a.in
% following proved in sst5a2.in
% Following proved in sst5a3.in
end_of_list.

list(passive).
-AFS(b,c,d1,c1,b1,c1,d,c) |  $ANS(TARG1INT).
-E(c1,d1,c,d) |  $ANS(TARG2INT).
-IFS(d,e,d1,c,d,e,d1,c1) |  $ANS(TARG3INT).
-IFS(c,e,c1,d,c,e,c1,d1) |  $ANS(TARG4INT).
-E(e,c,e,c1) |  $ANS(TARG5INT).
%  %  following negs from 66 proof steps.
%  %  following 66 shorter, temp.beeson.sst5a4.of4t25
%  %  %  Following negs from a 79-step proof of what might be the primary target.
%  %  following 10 are later steps in proofs of 38 and F, temp.beeson.sst5a4.out26
end_of_list.

list(sos).
    %prove the theorem in case c = c1, which should be easy.
a != b.
T(a,b,c).
T(a,b,d).
-T(a,c,d) | $ANS(CON1).
-T(a,d,c) | $ANS(CON2).
c1 = ext(a,d,c,d).
d1 = ext(a,c,c,d).
b1 = ext(a,c1,c,b).
b2 = ext(a,d1,d,b).
e = ip(c1,d,b,d1,c).
p = ext(e,c,c,d).
r = ext(d1,c,e,c).
q = ext(p,r,p,r).
c != b.
end_of_list.

list(hints).
```

% Following 104 purport to prove trans ob betw depend on axms with inner Pasch.
-E(x,y,z,u)|E(z,u,y,x).
E(ext(x,y,z,u),y,z,u).
E(x,y,ext(z,u,x,y),u).
E(x,y,z,u)| -E(z,u,y,x).
E(x,y,z,u)| -E(y,x,u,z).
-E(x,x,y,z)|E(u,u,y,z).
-T(x,y,z)| -T(y,x,u)| -E(y,z,x,u)|E(x,z,y,u).
T(ext(x,y,z,u),y,x).
-T(x,y,z)| -T(y,u,z)| -T(u,x,z)|u=x.
-E(x,y,z,u)| -E(z,y,x,u)|E3(x,z,y,z,x,u).
T(b,c,d1).
-T(c1,x,d1)| -T(c,y,d)| -E(x,d1,y,d)|E(c1,x,c,y).
E(d,c,d,c1).
T(d,e,d1).
E(c1,d,c1,d1).
E(c,d,c,d1).
T(c,e,c1).
E(c,e,e,c1).
$ANS(CON)| -T(b,d,c).
T(a,d,c1).
E(c,d1,c,d).
T(a,c,d1).
E(c,p,c,d).
T(e,c,p).
E(c,r,e,c).
T(d1,c,r).
E(r,q,p,r).
T(p,r,q).
E(ext(x,y,z,u),y,u,z).
E(d1,c,c,d).
E(e,c,r,c).
E(c,d,p,c).
T(p,c,e).
Col(c,d1,b).
E(c,d,d,c1).
E(c1,d1,c1,d).
E(d,c,c,d1).
E3(c,d1,d,c,d,d1).
Col(c,d1,a).
-T(a,d1,d)|$ANS(CON2).
-T(c,d,d1)|$ANS(CON).
E(c,p,c,d1).
E(c,r,c,e).
E(c,r,e,c1).
Col(c,r,d1).
E(r,q,r,p).
E(r,c,c,e).
E(d1,c,p,c).
E(r,p,r,q).
-T(c,d1,d)|$ANS(CON).
d1=c|E(r,p,e,d1).
T(d1,e,d).

T(c1,e,c).
Col(c,c1,e).
e=c|T(c1,e,p).
e=c|T(c1,c,p).
c=c1|E(e,d1,e,d).
T(a,b,c1).
T(b,d,c1).
-T(d,c,c1)|$ANS(CON).
Col(a,b,c1).
$ANS(CON)| -T(d1,d,d1).
$ANS(CON)| -E3(x,y,y,c,d,d1).
$ANS(CON)|E(r,p,e,d1).
$ANS(CON)|E(p,r,d1,e).
$ANS(CON)|E(r,q,e,d1).
$ANS(CON)|E(d1,e,p,r).
$ANS(CON)|E(e,d1,e,d).
$ANS(CON)|E(r,q,e,d).
$ANS(CON)|E(e,d,r,q).
$ANS(CON)|E(p,q,d1,d).
$ANS(CON)|d1=e|E(d,c,q,c).
$ANS(CON)|E(p,q,d,d1).
$ANS(CON)| -E3(x,y,y,d,c,c1).
$ANS(CON)| -T(e,d,d1)|E(d,c,q,c).
T(c1,d1,c)|$ANS(CON)|E(d,c,q,c).
e=c| -T(x,c1,p)| -T(e,x,p)|e=x.
e=c| -T(c,c1,p).
e=c|Col(c1,c,p).
-E(c,c,d,d1)|$ANS(CON).
-E(d,d,c,c1)|$ANS(CON).
$ANS(CON)|E(d,c,q,c)|d1=d.
$ANS(CON)|E(d,c,q,c).
$ANS(CON)|E(c,d,c,q).
$ANS(CON)|E(c,p,c,q).
$ANS(CON)|c=r|E(d1,p,d1,q).
e=c| -T(p,c1,c).
$ANS(CON)| -E(x,x,d,d1).
$ANS(CON)| -E(x,x,c,c1).
$ANS(CON)| -E(x,x,p,q).
$ANS(CON)| -E(c1,c,x,x).
$ANS(CON)| -T(e,c,e).
$ANS(CON)| -T(r,c,r).
-T(c1,c,d1)|$ANS(CON).
$ANS(CON)| -T(p,c1,c).
$ANS(CON)|Col(c1,c,p).
$ANS(CON)|E(d1,p,d1,q).
$ANS(CON)|c=d1|E(a,p,a,q).
$ANS(CON)|c=d1|E(b,p,b,q).
$ANS(CON)|E(a,p,a,q).
$ANS(CON)|E(b,p,b,q).
$ANS(CON)|E(c1,p,c1,q).
$ANS(CON)|c1=c.
$ANS(CON)| -T(p,c,c).
% Following 87 sorted from temp.beeson.sst5a4.out40t1k1 -k4,

of five lemmas Beeson wishes to avoid.
$ANS(TARG1INT)| -E(b2,c,b1,c).
$ANS(TARG1INT)| -E(b,b2,b,b1).
$ANS(TARG4INT)| -T(c,e,c1).
$ANS(TARG4INT)| -T(c,ip(c1,d,b,d1,c),c1).
$ANS(TARGINT)| -E(c,d,c1,d1).
$ANS(TARGINT)| -IFS(x,c,y,d,z,c1,u,d1).
b1=b2.
Col(a,b,b1).
Col(e,d1,d).
E(a,b1,a,b2).
E(b1,c1,b,c).
E(b1,c1,c,b).
E(b1,d,b,d1).
E(b2,d1,d,b).
E(b2,d,b,d1).
E(b,b1,b2,b).
E(b,b1,b,b2).
E(b,b2,b1,b).
E(b,b2,b,b1).
-E(b,c1,b1,c)|$ANS(TARG1INT).
E(b,c1,b2,c).
E(b,c,b1,c1).
E(b,d,b2,d1).
E(c1,b1,c,b).
E(c1,d1,c1,d).
E(c1,d1,d1,c).
E(c1,d,c1,d1).
E(c1,d,c1,ext(a,c,c,d)).
E(c1,d,c,d1).
E(c,b2,c1,b).
E(c,d1,c1,d).
E(c,d1,c,d).
E(c,d,c1,d).
E(c,d,c,d1).
E(c,d,c,ext(a,c,c,d)).
E(c,d,d1,c).
e=d1|$ANS(TARG3INT).
E(d1,b2,d,b).
E(d1,c,d1,c1).
E(d1,c,e,c1)|$ANS(TARG3INT).
-E(d1,c,e,c1)|$ANS(TARG5INT)|$ANS(TARG3INT).
E(d,c1,c,d).
E(d,c1,d1,c).
E(d,c1,d,c).
E(d,c,d,c1).
-E(d,d1,d,d1)|$ANS(TARG3INT).
E(ext(x,y,z,u),y,z,u).
e=ip(ext(a,d,c,d),d,b,d1,c).
E(x,y,ext(z,u,x,y),u).
E(x,y,z,u)| -E(y,x,u,z).
E(x,y,z,u)| -E(z,u,y,x).
IFS(b,c,b2,d,b2,c1,b,d1).

```
-IFS(c,e,c1,d,c,e,c1,ext(a,c,c,d))|$ANS(TARG4INT).
IFS(d,e,d1,c,d,e,d1,c1).
T(a,b,b1).
T(a,b,b2).
T(a,b,c1).
T(a,c1,b1).
T(a,c,b2).
T(a,c,d1).
T(a,d1,b2).
T(a,d,c1).
T(b1,c1,a).
T(b1,c1,b).
T(b1,c1,d).
T(b2,c1,b).
T(b2,c,a).
T(b2,c,b).
T(b2,d1,a).
T(b2,d1,c).
T(b,c1,b1).
T(b,c,b2).
T(b,c,d1).
T(b,d,c1).
T(b,d,ext(a,d,x,y)).
T(c1,b,a).
T(c1,d,a).
T(c1,d,b).
T(c,b,a).
T(c,d1,b2).
T(c,ip(c1,d,b,d1,c),c1).
T(d1,c,a).
T(d1,c,b).
T(d,b,a).
T(d,e,d1).
T(ext(a,d,x,y),d,b).
-T(x,y,z)| -T(x,y,u)| -E(y,z,y,u)|E(x,z,x,u).
end_of_list.
```

If a proof would be found with this file, the prediction was that the proof would be longer, perhaps much longer, than the earlier-obtained 104-step proof. For you to see immediately why such is the case, by way of a totally trivial example, the inclusion of steps near the end of a proof in hand in an input file should, and usually does, lead to a quite short proof. So, in a sense conversely, the removal of items from an input file, such as the removal of Lemmas RG1 through RG5, should, if all goes well, lead to finding a longer proof than that found in their presence. Now, with the input file just given, you might discover that one or more of the five lemmas would be deduced and used in a sought-after proof. Although none of the runs that proved one or more of the five lemmas (even after much CPU time was used) led to a proof of connectivity, as so often occurred, use of the corresponding proofs might just get my goal. And indeed OTTER did find the following longer proof, a proof that I will examine in some detail.

A Proof Using Inner Pasch of Length 154 of Connectivity of Betweenness

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Sun Feb  3 09:24:12 2013
```

The command was "otter". The process ID is 14677.
----> UNIT CONFLICT at 1267.28 sec ----> 187660 [binary,187659.1,185347.1] $ANS(CON2)|$ANS(CON1).

Length of proof is 154. Level of proof is 27.

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

1 [] E(x,y,y,x).
2 [] -E(x,y,z,v)| -E(x,y,z2,v2)|E(z,v,z2,v2).
4 [] T(x,y,ext(x,y,w,v)).
5 [] E(y,ext(x,y,w,v),w,v).
6 [] -E(x,y,x1,y1)| -E(y,z,y1,z1)| -E(x,v,x1,v1)| -E(y,v,y1,v1)| -T(x,y,z)| -T(x1,y1,z1)|x=y|E(z,v,z1,v1).
8 [] -T(xa,xp,xc)| -T(xb,xq,xc)|T(xp,ip(xa,xp,xc,xb,xq),xb).
9 [] -T(xa,xp,xc)| -T(xb,xq,xc)|T(xq,ip(xa,xp,xc,xb,xq),xa).
13 [] E(x,y,x,y).
14 [] -E(xa,xb,xc,xd)|E(xc,xd,xa,xb).
15 [] -E(xa,xb,xc,xd)|E(xb,xa,xc,xd).
16 [] -E(xa,xb,xc,xd)| -E(xc,xd,xe,xf)|E(xa,xb,xe,xf).
17 [] -E(xa,xb,xc,xd)|E(xa,xb,xd,xc).
18 [] E(x,x,y,y).
19 [] -T(xa,xb,xc)| -T(xa1,xb1,xc1)| -E(xa,xb,xa1,xb1)| -E(xb,xc,xb1,xc1)|E(xa,xc,xa1,xc1).
21 [] T(x,y,y).
23 [] -T(xa,xb,xc)|T(xc,xb,xa).
24 [] T(xa,xa,xb).
26 [] -T(xa,xb,xd)| -T(xb,xc,xd)|T(xa,xb,xc).
27 [] -T(xa,xb,xd)| -T(xb,xc,xd)|T(xa,xc,xd).
28 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xb,xc,xd).
32 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xa,xb,xd).
33 [] -T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xc,xd).
35 [] -IFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1)|E(xb,xd,xb1,xd1).
36 [] -T(xa,xb,xc)| -T(xa1,xb1,xc1)| -E(xa,xc,xa1,xc1)| -E(xb,xc,xb1,xc1)|E(xa,xb,xa1,xb1).
48 [] -T(xa,xb,xc)| -T(za,zb,zc)| -E(xa,xc,za,zc)| -E(xb,xc,zb,zc)|
 -E(xa,xd,za,zd)| -E(xc,xd,zc,zd)|IFS(xa,xb,xc,xd,za,zb,zc,zd).
52 [] -E(xa1,xa2,xb1,xb2)| -E(xa1,xa3,xb1,xb3)| -E(xa2,xa3,xb2,xb3)|E3(xa1,xa2,xa3,xb1,xb2,xb3).
57 [] -T(xa,xb,xc)| -E3(xa,xb,xc,xa1,xb1,xc1)|T(xa1,xb1,xc1).
59 [] Col(xa,xb,xc)| -T(xa,xb,xc).
60 [] Col(xa,xb,xc)| -T(xb,xc,xa).
61 [] Col(xa,xb,xc)| -T(xc,xa,xb).
76 [] xa=xb| -Col(xa,xb,xc)| -E(xa,xp,xa,xq)| -E(xb,xp,xb,xq)|E(xc,xp,xc,xq).
77 [] xa=xb| -Col(xa,xb,xc)| -E(xa,xc,xa,xc1)| -E(xb,xc,xb,xc1)|xc=xc1.
98 [] a!=b.
99 [] T(a,b,c).
100 [] T(a,b,d).
101 [] -T(a,c,d)|$ANS(CON1).
102 [] -T(a,d,c)|$ANS(CON2).
103 [] c1=ext(a,d,c,d).
104 [] d1=ext(a,c,c,d).
105 [] b1=ext(a,c1,c,b).
106 [] b2=ext(a,d1,d,b).
107 [] e=ip(c1,d,b,d1,c).
108 [] p=ext(e,c,c,d).
109 [] r=ext(d1,c,e,c).
110 [] q=ext(p,r,p,r).

304 [binary,2.2,1.1] -E(x,y,z,u)|E(z,u,y,x).
307 [hyper,2,1,5] E(ext(x,y,z,u),y,z,u).
308 [hyper,2,5,1] E(x,y,ext(z,u,x,y),u).
309 [hyper,2,5,13] E(x,y,z,ext(u,z,x,y)).
311 [binary,15.1,14.2] E(x,y,z,u)| -E(z,u,y,x).
313 [hyper,16,5,1] E(x,ext(y,x,z,u),u,z).
314 [binary,17.1,15.2] E(x,y,z,u)| -E(y,x,u,z).
316 [binary,18.1,16.1] -E(x,x,y,z)|E(u,u,y,z).
319 [binary,19.3,13.1] -T(x,y,z)| -T(x,y,u)| -E(y,z,y,u)|E(x,z,x,u).
320 [binary,19.3,1.1] -T(x,y,z)| -T(y,x,u)| -E(y,z,x,u)|E(x,z,y,u).
326 [binary,23.1,4.1] T(ext(x,y,z,u),y,x).
425 [binary,52.1,1.1] -E(x,y,z,u)| -E(z,y,x,u)|E3(x,z,y,z,x,u).
554 [binary,99.1,23.1] T(c,b,a).
616 [binary,100.1,23.1] T(d,b,a).
640 [hyper,28,100,4] T(b,d,ext(a,d,x,y)).
727 [para_from,103.1.2,5.1.2] E(d,c1,c,d).
728 [para_from,103.1.2,4.1.3] T(a,d,c1).
729 [para_from,104.1.2,5.1.2] E(c,d1,c,d).
730 [para_from,104.1.2,4.1.3] T(a,c,d1).
732 [para_from,105.1.2,5.1.2] E(c1,b1,c,b).
733 [para_from,105.1.2,4.1.3] T(a,c1,b1).
735 [para_from,106.1.2,5.1.2] E(d1,b2,d,b).
736 [para_from,106.1.2,4.1.3] T(a,d1,b2).
737 [para_into,107.1.2.1,103.1.1] e=ip(ext(a,d,c,d),d,b,d1,c).
738 [para_from,108.1.2,5.1.2] E(c,p,c,d).
739 [para_from,108.1.2,4.1.3] T(e,c,p).
741 [para_from,109.1.2,5.1.2] E(c,r,e,c).
742 [para_from,109.1.2,4.1.3] T(d1,c,r).
747 [para_from,110.1.2,5.1.2] E(r,q,p,r).
748 [para_from,110.1.2,4.1.3] T(p,r,q).
770 [para_into,307.1.1,106.1.2] E(b2,d1,d,b).
772 [para_into,307.1.1,104.1.2] E(d1,c,c,d).
786 [para_into,308.1.3,109.1.2] E(e,c,r,c).
787 [para_into,308.1.3,108.1.2] E(c,d,p,c).
790 [para_into,308.1.3,104.1.2] E(c,d,d1,c).
791 [para_into,308.1.3,103.1.2] E(c,d,c1,d).
832 [para_into,309.1.4,104.1.2] E(c,d,c,d1).
833 [para_into,309.1.4,103.1.2] E(c,d,d,c1).
889 [para_into,326.1.1,108.1.2] T(p,c,e).
890 [para_into,326.1.1,106.1.2] T(b2,d1,a).
891 [para_into,326.1.1,105.1.2] T(b1,c1,a).
892 [para_into,326.1.1,104.1.2] T(d1,c,a).
893 [para_into,326.1.1,103.1.2] T(c1,d,a).
1046 [hyper,26,326,616] T(ext(a,d,x,y),d,b).
1092 [para_into,640.1.3,103.1.2] T(b,d,c1).
1137 [binary,727.1,311.2] E(d,c,d,c1).
1214 [hyper,32,100,728] T(a,b,c1).
1245 [ur,26,728,102] -T(d,c,c1)|$ANS(CON2).
1248 [binary,729.1,311.2] E(d,c,c,d1).
1280 [binary,730.1,61.2] Col(c,d1,a).
1327 [hyper,28,99,730] T(b,c,d1).
1356 [ur,26,730,101] -T(c,d,d1)|$ANS(CON1).
1388 [para_into,313.1.2,110.1.2] E(r,q,r,p).

1389 [para_into,313.1.2,109.1.2] E(c,r,c,e).
1394 [binary,732.1,314.2] E(b1,c1,b,c).
1592 [hyper,32,730,736] T(a,c,b2).
1595 [hyper,28,730,736] T(c,d1,b2).
1625 [para_from,737.1.2,8.3.2,unit_del,1046] -T(d1,c,b)|T(d,e,d1).
1738 [binary,741.1,314.2] E(r,c,c,e).
1775 [binary,742.1,61.2] Col(c,r,d1).
1853 [binary,747.1,311.2] E(r,p,r,q).
2035 [binary,770.1,311.2] E(b,d,b2,d1).
2260 [hyper,16,772,787] E(d1,c,p,c).
2295 [hyper,16,727,790] E(d,c1,d1,c).
2328 [hyper,16,729,791] E(c,d1,c1,d).
2358 [hyper,52,729,832,1] E3(c,d1,d,c,d,d1).
2374 [hyper,16,738,832] E(c,p,c,d1).
2379 [hyper,2,791,832] E(c1,d,c,d1).
2978 [hyper,27,890,892] T(b2,c,a).
2983 [hyper,26,890,892] T(b2,d1,c).
2985 [hyper,26,892,554] T(d1,c,b).
3097 [hyper,27,893,616] T(c1,b,a).
3100 [hyper,26,891,893] T(b1,c1,d).
3102 [hyper,26,893,616] T(c1,d,b).
3412 [binary,1214.1,59.2] Col(a,b,c1).
3468 [hyper,32,1214,733] T(a,b,b1).
3473 [hyper,28,1214,733] T(b,c1,b1).
3567 [ur,57,21,1245] $ANS(CON2)| -E3(x,y,y,d,c,c1).
3679 [binary,1327.1,61.2] Col(c,d1,b).
4004 [ur,57,21,1356] $ANS(CON1)| -E3(x,y,y,c,d,d1).
4334 [hyper,32,99,1592] T(a,b,b2).
4341 [hyper,28,99,1592] T(b,c,b2).
5016 [binary,2358.1,57.2,unit_del,1356] -T(c,d1,d)|$ANS(CON1).
5068 [hyper,6,2260,1389,1,2374,742,889] d1=c|E(r,p,e,d1).
5314 [hyper,26,2978,554] T(b2,c,b).
5476 [hyper,19,1092,2983,2035,2295] E(b,c1,b2,c).
5876 [hyper,26,891,3097] T(b1,c1,b).
6051 [hyper,19,3100,1327,1394,2379] E(b1,d,b,d1).
6185 [hyper,19,1595,3102,2328,735] E(c,b2,c1,b).
6208 [hyper,9,3102,2985] T(c,ip(c1,d,b,d1,c),c1).
6284 [binary,3468.1,59.2] Col(a,b,b1).
6566 [binary,3567.1,425.3,unit_del,833] $ANS(CON2)| -E(d,d,c,c1).
6648 [binary,4004.1,425.3,unit_del,1248] $ANS(CON1)| -E(c,c,d,d1).
7053 [para_from,5068.1.1,5016.1.2,unit_del,24] $ANS(CON1)|E(r,p,e,d1).
7236 [hyper,19,3473,5314,5476,732] E(b,b1,b2,b).
7562 [para_into,6208.1.2,107.1.2] T(c,e,c1).
7568 [binary,6566.1,316.2] $ANS(CON2)| -E(x,x,c,c1).
7587 [binary,6648.1,316.2] $ANS(CON1)| -E(x,x,d,d1).
7606 [binary,7053.1,314.2] $ANS(CON1)|E(p,r,d1,e).
7638 [hyper,16,1388,7053] $ANS(CON1)|E(r,q,e,d1).
7645 [hyper,2,7053,1853] $ANS(CON1)|E(e,d1,r,q).
7660 [binary,7236.1,17.1] E(b,b1,b,b2).
7790 [binary,7562.1,23.1] T(c1,e,c).
7913 [binary,7568.1,314.1] $ANS(CON2)| -E(x,x,c1,c).
7915 [binary,7568.1,304.2] $ANS(CON2)| -E(c1,c,x,x).
7963 [binary,7587.1,311.1] $ANS(CON1)| -E(d,d1,x,x).

7997 [binary,7606.1,14.1] $ANS(CON1)|E(d1,e,p,r).
8149 [hyper,319,3468,4334,7660] E(a,b1,a,b2).
8201 [binary,7790.1,60.2] Col(c,c1,e).
8249 [hyper,33,7790,739] e=c|T(c1,c,p).
8436 [hyper,77,6284,8149,7660,unit_del,98] b1=b2.
8648 [binary,8249.2,59.2] e=c|Col(c1,c,p).
8745 [para_from,8436.1.1,6051.1.1] E(b2,d,b,d1).
8746 [para_from,8436.1.1,5876.1.1] T(b2,c1,b).
9103 [hyper,48,4341,8746,1,6185,2035,8745] IFS(b,c,b2,d,b2,c1,b,d1).
9345 [binary,9103.1,35.1] E(c,d,c1,d1).
66751 [hyper,2,9345,791] E(c1,d1,c1,d).
66752 [hyper,2,9345,790] E(c1,d1,d1,c).
66757 [ur,36,24,9345,832,7913] -T(c1,c,d1)|$ANS(CON2).
66859 [hyper,76,8201,729,66751] c=c1|E(e,d1,e,d).
66914 [binary,66752.1,304.1] E(d1,c,d1,c1).
67000 [binary,66757.1,23.2] $ANS(CON2)| -T(d1,c,c1).
67148 [para_from,66859.1.1,66757.1.2,unit_del,24] $ANS(CON2)|E(e,d1,e,d).
67237 [binary,67148.1,314.2] $ANS(CON2)|E(d1,e,d,e).
67321 [hyper,2,67148,7645] $ANS(CON2)|E(e,d,r,q)|$ANS(CON1).
99483 [hyper,16,7606,67237] $ANS(CON2)|E(p,r,d,e)|$ANS(CON1).
99495 [ur,36,24,13,67237,7963] $ANS(CON2)| -T(d,d1,e)|$ANS(CON1).
99519 [binary,99495.1,23.2] $ANS(CON2)|$ANS(CON1)| -T(e,d1,d).
176170 [binary,1625.1,2985.1] T(d,e,d1).
176196 [binary,1625.2,23.1,unit_del,2985] T(d1,e,d).
176239 [hyper,48,176170,176170,13,13,1137,66914] IFS(d,e,d1,c,d,e,d1,c1).
176317 [hyper,19,748,176170,99483,7638,factor_simp] E(p,q,d,d1)|$ANS(CON2)|$ANS(CON1).
177084 [hyper,6,7997,67321,2260,786,176196,748,factor_simp] d1=e|E(d,c,q,c)|$ANS(CON1)|$ANS(CON2).
177441 [binary,176239.1,35.1] E(e,c,e,c1).
177560 [ur,16,176317,7587,factor_simp] $ANS(CON2)|$ANS(CON1)| -E(x,x,p,q).
177708 [para_from,177084.1.1,99519.3.2,unit_del,24,factor_simp,factor_simp]
$ANS(CON2)|$ANS(CON1)|E(d,c,q,c).
177804 [binary,177708.1,314.2] $ANS(CON2)|$ANS(CON1)|E(c,d,c,q).
177994 [hyper,16,738,177804] $ANS(CON2)|$ANS(CON1)|E(c,p,c,q).
178086 [hyper,76,1775,177994,1853] $ANS(CON2)|$ANS(CON1)|c=r|E(d1,p,d1,q).
183682 [binary,177441.1,15.1] E(c,e,e,c1).
183798 [hyper,16,741,177441] E(c,r,e,c1).
183841 [binary,183682.1,320.3,unit_del,7562,7568] -T(e,c,e)|$ANS(CON2).
183961 [ur,19,7562,1738,183798,7568] -T(r,c,r)|$ANS(CON2).
184125 [para_into,183841.1.1,8648.1.1,unit_del,24] $ANS(CON2)|Col(c1,c,p).
184128 [para_into,183841.1.1,8249.1.1,unit_del,24] $ANS(CON2)|T(c1,c,p).
184282 [para_into,183961.1.1,178086.1.2,unit_del,24,factor_simp]
$ANS(CON2)|$ANS(CON1)|E(d1,p,d1,q).
185347 [ur,36,184128,24,177994,7915,factor_simp,factor_simp]
$ANS(CON2)| -E(c1,p,c,q)|$ANS(CON1).
186100 [hyper,76,3679,177994,184282,factor_simp,factor_simp]
$ANS(CON2)|$ANS(CON1)|c=d1|E(b,p,b,q).
186101 [hyper,76,1280,177994,184282,factor_simp,factor_simp]
$ANS(CON2)|$ANS(CON1)|c=d1|E(a,p,a,q).
186449 [para_from,186100.1.1,67000.2.2,unit_del,24,factor_simp]
$ANS(CON2)|$ANS(CON1)|E(b,p,b,q).
187134 [para_from,186101.1.1,67000.2.2,unit_del,24,factor_simp]
$ANS(CON2)|$ANS(CON1)|E(a,p,a,q).
187326 [hyper,76,3412,187134,186449,unit_del,98,factor_simp,factor_simp]

$ANS(CON2)|$ANS(CON1)|E(c1,p,c1,q).
187515 [hyper,76,184125,187326,177994,unit_del,177560,factor_simp,factor_simp,
factor_simp,factor_simp,factor_simp] $ANS(CON2)|$ANS(CON1)|c1=c.
187659 [para_from,187515.1.1,187326.3.3,factor_simp,factor_simp]
$ANS(CON2)|$ANS(CON1)|E(c1,p,c,q).
187660 [binary,187659.1,185347.1] $ANS(CON2)|$ANS(CON1).

Before completing the cited bidirectional 154-step proof, the program found, in order, forward proofs of RG1, 2, 4, 3, and 5, of respective lengths 45, 60, 65, 69, and 70. Although none of the five was included in the input, and each was proved, the third and fifth actually are found in the 154-step proof. Somewhat piquant, when I blocked the use of binary resolution and made no other changes, OTTER proved only the first cited three of the lemmas. Also, in the presence of binary resolution but with demodulators to block the retention when deduced of the third and fifth lemmas, the proof of connectivity is still not forthcoming. As in the 104-step proof, transitivity of betweenness is employed in the 154-step proof. In contrast to the 104-step proof that relies on inner Pasch implicitly, in the 154-step proof, both clauses for inner Pasch are relied upon explictly. For clarity, by relying on 3.51 and 3.61, implicit use of inner Pasch is seen. In other words, the 154-step proof relies explicitly on two additional clauses from among the Tarski axiom system that includes inner Pasch, when compared with the 104-step proof.

For further comparison, among the theorems previously proved, the 154-step proof relies on the use of 2.1, 4.2, part of the definition of 4.1, and 4.18. Whereas the 104-step proof relies on the use of five defined points (by Beeson), the 154-step proof relies on eight defined points. Most important to note, other than Beeson's crucial suggestions and guidance and McCune's incredible automated reasoning program OTTER, is the fact that neither the 104-step nor the 154-step proof would have been found without the direction provided to Beeson and me by the Szmielew book and its approach to proving connectivity. That theorem, had we not followed Szmielew's approach, would have been clearly out of reach for OTTER and for us. To amplify just a bit, when in 2002 the formula *XCB* of equivalential calculus was established to be a single axiom for this area of logic, no proof existed; nothing was extant that could be followed or emulated.

At this point in the experiments I was conducting, my curiosity arose in the context of how much connection exists between transitivity and connectivity of betweenness. In other words, if transitivity is removed from the input file that yielded the 154-step proof, will OTTER still find a proof of the dependence of connectivity? Of course, as you and I know, to tidy things up, proofs of all the theorems used in such a proof would be nice to have in hand, proofs in which transitivity is not used in the input, which is possible since transitivity has been proved dependent. (I do not intend to follow that thorough path at the moment but simply try for a proof based on the input file that was used to obtain the 154-step proof, with transitivity of course deleted. I note, however, that my study of the Tarski axioms given in Section 1, in which outer Pasch is present, resulted in my finding many proofs of Szmielew theorems with an input file in which both transitivity and connectivity of betweenness were absent.)

Again, success was the result. OTTER found a 146-step bidirectional proof in approximately 151 CPU-seconds and with retention of clause (118384). You have an interesting example of using fewer axioms and, yet, obtaining a shorter proof, one of length less than 154. Again, as in the 154-step proof, OTTER's proof relied on eight defined points, the following.

102 [] c1=ext(a,d,c,d).
103 [] d1=ext(a,c,c,d).
104 [] b1=ext(a,c1,c,b).
105 [] b2=ext(a,d1,d,b).
106 [] e=ip(c1,d,b,d1,c).
107 [] p=ext(e,c,c,d).
108 [] r=ext(d1,c,e,c).
109 [] q=ext(p,r,p,r).

Both clauses for inner Pasch were cited, as well as the two for extensionality. As for piquant differences, thirty-one deduced clauses are present in the 146-step proof that are not found in the 154-step proof.

With the finding of a proof of connectivity that does not rely on transitivity, not even at the deduced level, I naturally wondered whether I could block, at the same time, some other theorem that was used from among those relied upon among the input clauses. An appealing target was Szmielew's 4.3, a theorem I had not sought a proof of, but simply accepted it for use; Beeson I am quite sure had proved it.

```
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xc,xa1,xc1) | -E(xb,xc,xb1,xc1) |
  E(xa,xb,xa1,xb1).  % Satz 4.3
```

My experiment with the goal of proving connectivity in the absence of transitivity and Satz 4.3 yielded no proof. After various other experiments, I chose as target (to avoid using) 4.19, the following.

```
-T(xa,xc,xb)| -E(xa,xc,xa,xc1)| -E(xb,xc,xb,xc1)|xc=xc1.  Satz 4.19
```

And now you are introduced even more to the vagaries of research. Specifically, after checking, I found that 4.19 was not used in the 154-step proof, among the items that occur before deductions are given. However—and how fortunate that I was hasty and did not check before conducting the experiment—OTTER did find a proof of length 151 that avoids, at the so-called axiomatic level, transitivity and Satz 4.19. But, you are perhaps confused in that 4.19 was not used in the 154-step proof, but I simply took the input file that produced it and commented out 4.19, and, as I noticed later, 4.3 was still commented out from failed attempts. I have no explanation. Instead, the 151-step proof, with more than a chuckle from it, is a proof that avoids, before deduced clauses are presented, transitivity and 4.19 and, surprising to me (because of so many simultaneous experiments), finally avoids 4.3. Yes, a direct attempt at avoiding the use of 4.3, along with avoiding transitivity, in the input yielded nothing, and yet the added avoidance of 4.19 won the game. As expected, I leave to you the solving of the puzzle just described. By the way, among my experiments, I have obtained proofs that rely on forty-three items from among those offered in the input, and no fewer; the 104-step proof relies on forty-eight. Would Alama enjoy such successes, a reduction in the number of axioms and proved theorems needed to reach a desired target? More generally, would those individuals interested in proof shortening find so-called assumption reduction of interest?

With the cited successes in hand, those familiar with my research would accurately predict I would turn to proof shortening. Of course, I turned to the use of ancestor subsumption, the approach McCune devised, motivated by his intent to aid me in my interest in finding shorter proofs. When you instruct OTTER by including set(ancestor_subsume), and wisely include set(back_sub), the program automatically compares two proofs of the same conclusion and, if found, prefers the strictly shorter. I began with the input file that produced the 154-step proof, and I adjoined the two cited commands. Rather than detailing all the experiments, I will present the high points.

When the sequence of experiments ceased yielding shorter and still shorter proofs of connectivity, a sequence that was based on replacing one set of hints by that corresponding to a found shorter proof, my approach was that of relying on demodulation. To start, what you do is take an input file, place weight(junk,1000), for example, in weight_list(pick_and_purge), where 1000 is strictly greater than the value assigned to max_weight, and include a list(demodulators). The plan is to make a series of runs in which each run is based on the preceding with a single new demodulator placed in list(demodulators). The new demodulator is adjoined and chosen because it corresponds to blocking one of the deduced steps of the proof currently in hand and because that blocking enables the program to find a strictly shorter proof than that in hand. For example, assume that you have a proof **P** and that if you place the following demodulator, where *T* is a deduced step in **P**, in list(demodulators), the program will present you with a proof **Q** with the length of **Q** strictly less than the length of **P**.

```
EQ(T,junk).
```

As part of the plan, you iterate, focusing on your sequence of runs until no strictly shorter proof results. McCune wrote for me a program, otter-loop, that has me produce a file of demodulators (almost always corresponding to the deduced steps of the proof in hand) to try, an input file, a file to accept a summary of results after blocking the proof steps one at a time, and a message file. The command is the following.

```
otter-loop filename1 filename2 demodulators filename3, filename4
```

The filename1 is the input file, 2 is that containing the demodulators to consider, 3 is the file that contains the results of demodulator blocking, and 4 is for the messages (about errors, for example, which can be made). McCune produced various programs that I still use, programs that materially aid my research. Yes, he was a splendid colleague, and brilliant!

You might naturally wish to have one or more clues that explain why this demodulator-blocking can lead to shorter proofs. The following example template will serve nicely. Let **P** be a 40-step proof such that its twentieth step and its tenth together are the parents of the thirtieth. And, to avoid distraction, let us assume that the twentieth is not used elsewhere. It can happen that when the twentieth is blocked (by demodulating it, when deduced, to junk and discarded because of junk being assigned the value 1000), the twenty-second considered with the twelfth also provides the needed parents for the deduction of the thirtieth. Thus, with the removal of the twentieth from consideration, the program can produce a proof **Q** of length 39, which, ignoring parentage, is a subproof of the 40-step proof. Yes, I have often found a shorter proof all of whose steps were among those of a longer proof, but the parentage was different. Much satisfaction, and even amusement, can result from eventually finding a 34-step proof all of whose steps are among those of a 40-step proof that was used to initiate the search for a short proof.

A move that is often profitable has you, once the iterative process focusing on demodulation blocking yields no additional progress, produce another input file based on the file that was used in the beginning. The new file is obtained from the older by now removing all the adjoined demodulators and replacing the hints, or resonators, with the deduced steps of the latest and shortest proof that has been found. You can then begin anew, blocking the steps of this latest proof one at a time. McCune, in his genius, provided me with a variant of otter-loop, a program that enabled me to block proof steps two at a time, three at a time, and more, if I thought it profitable. Sometimes it was profitable.

I did apply the approach just discussed in the preceding paragraphs, beginning with the 154-step proof and its input file. I eventually was presented, by OTTER, with a proof of length 122, the following.

A Shorter Proof of the Dependency of Connectivity

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Fri Mar 22 07:23:22 2013
The command was "otter". The process ID is 31397.
----> UNIT CONFLICT at   3.10 sec ----> 6410 [binary,6409.1,3371.1] $ANS(CON2)|$ANS(CON1).

Length of proof is 122.  Level of proof is 27.

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

1 [] E(x,y,y,x).
2 [] -E(x,y,z,v)| -E(x,y,z2,v2)|E(z,v,z2,v2).
4 [] T(x,y,ext(x,y,w,v)).
5 [] E(y,ext(x,y,w,v),w,v).
6 [] -E(x,y,x1,y1)| -E(y,z,y1,z1)| -E(x,v,x1,v1)| -E(y,v,y1,v1)| -T(x,y,z)| -T(x1,y1,z1)|x=y|E(z,v,z1,v1).
8 [] -T(xa,xp,xc)| -T(xb,xq,xc)|T(xp,ip(xa,xp,xc,xb,xq),xb).
9 [] -T(xa,xp,xc)| -T(xb,xq,xc)|T(xq,ip(xa,xp,xc,xb,xq),xa).
13 [] E(x,y,x,y).
14 [] -E(xa,xb,xc,xd)|E(xc,xd,xa,xb).
15 [] -E(xa,xb,xc,xd)|E(xb,xa,xc,xd).
16 [] -E(xa,xb,xc,xd)| -E(xc,xd,xe,xf)|E(xa,xb,xe,xf).
17 [] -E(xa,xb,xc,xd)|E(xa,xb,xd,xc).
18 [] E(x,x,y,y).
19 [] -T(xa,xb,xc)| -T(xa1,xb1,xc1)| -E(xa,xb,xa1,xb1)| -E(xb,xc,xb1,xc1)|E(xa,xc,xa1,xc1).
21 [] T(x,y,y).
23 [] -T(xa,xb,xc)|T(xc,xb,xa).
```

24 [] T(xa,xa,xb).
26 [] -T(xa,xb,xd)| -T(xb,xc,xd)|T(xa,xb,xc).
28 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xb,xc,xd).
32 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xa,xb,xd).
33 [] -T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xc,xd).
35 [] -IFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1)|E(xb,xd,xb1,xd1).
36 [] -T(xa,xb,xc)| -T(xa1,xb1,xc1)| -E(xa,xc,xa1,xc1)| -E(xb,xc,xb1,xc1)|E(xa,xb,xa1,xb1).
48 [] -T(xa,xb,xc)| -T(za,zb,zc)| -E(xa,xc,za,zc)| -E(xb,xc,zb,zc)|
-E(xa,xd,za,zd)| -E(xc,xd,zc,zd)|IFS(xa,xb,xc,xd,za,zb,zc,zd).
52 [] -E(xa1,xa2,xb1,xb2)| -E(xa1,xa3,xb1,xb3)| -E(xa2,xa3,xb2,xb3)|E3(xa1,xa2,xa3,xb1,xb2,xb3).
57 [] -T(xa,xb,xc)| -E3(xa,xb,xc,xa1,xb1,xc1)|T(xa1,xb1,xc1).
59 [] Col(xa,xb,xc)| -T(xa,xb,xc).
60 [] Col(xa,xb,xc)| -T(xb,xc,xa).
61 [] Col(xa,xb,xc)| -T(xc,xa,xb).
65 [] -Col(x,y,z)|Col(y,x,z).
76 [] xa=xb| -Col(xa,xb,xc)| -E(xa,xp,xa,xq)| -E(xb,xp,xb,xq)|E(xc,xp,xc,xq).
77 [] xa=xb| -Col(xa,xb,xc)| -E(xa,xc,xa,xc1)| -E(xb,xc,xb,xc1)|xc=xc1.
98 [] a!=b.
99 [] T(a,b,c).
100 [] T(a,b,d).
101 [] -T(a,c,d)|$ANS(CON1).
102 [] -T(a,d,c)|$ANS(CON2).
103 [] c1=ext(a,d,c,d).
104 [] d1=ext(a,c,c,d).
105 [] b1=ext(a,c1,c,b).
106 [] b2=ext(a,d1,d,b).
107 [] e=ip(c1,d,b,d1,c).
108 [] p=ext(e,c,c,d).
109 [] r=ext(d1,c,e,c).
110 [] q=ext(p,r,p,r).
493 [hyper,28,100,4] T(b,d,ext(a,d,x,y)).
584 [para_from,103.1.2,5.1.2] E(d,c1,c,d).
585 [para_from,103.1.2,4.1.3] T(a,d,c1).
588 [para_from,104.1.2,5.1.2] E(c,d1,c,d).
589 [para_from,104.1.2,4.1.3] T(a,c,d1).
593 [para_from,105.1.2,5.1.2] E(c1,b1,c,b).
594 [para_from,105.1.2,4.1.3] T(a,c1,b1).
598 [para_from,106.1.2,5.1.2] E(d1,b2,d,b).
599 [para_from,106.1.2,4.1.3] T(a,d1,b2).
600 [para_into,107.1.2.1,103.1.1] e=ip(ext(a,d,c,d),d,b,d1,c).
604 [para_from,108.1.2,5.1.2] E(c,p,c,d).
605 [para_from,108.1.2,4.1.3] T(e,c,p).
609 [para_from,109.1.2,5.1.2] E(c,r,e,c).
610 [para_from,109.1.2,4.1.3] T(d1,c,r).
617 [para_from,110.1.2,5.1.2] E(r,q,p,r).
618 [para_from,110.1.2,4.1.3] T(p,r,q).
624 [binary,493.1,23.1] T(ext(a,d,x,y),d,b).
640 [binary,584.1,14.1] E(c,d,d,c1).
660 [hyper,2,584,1] E(c,d,c1,d).
714 [hyper,32,100,585] T(a,b,c1).
739 [ur,26,585,102] -T(d,c,c1)|$ANS(CON2).
743 [binary,588.1,15.1] E(d1,c,c,d).
744 [binary,588.1,14.1] E(c,d,c,d1).

764 [hyper,2,588,1] E(c,d,d1,c).
765 [binary,589.1,61.2] Col(c,d1,a).
807 [hyper,28,99,589] T(b,c,d1).
830 [ur,26,589,101] -T(c,d,d1)|$ANS(CON1).
834 [binary,593.1,15.1] E(b1,c1,c,b).
947 [binary,598.1,17.1] E(d1,b2,b,d).
1012 [hyper,32,589,599] T(a,c,b2).
1015 [hyper,28,589,599] T(c,d1,b2).
1038 [para_from,600.1.2,8.3.2,unit_del,624] -T(d1,c,b)|T(d,e,d1).
1065 [binary,604.1,17.1] E(c,p,d,c).
1090 [hyper,2,604,1] E(c,d,p,c).
1109 [binary,605.1,23.1] T(p,c,e).
1155 [binary,609.1,17.1] E(c,r,c,e).
1179 [hyper,2,609,1] E(e,c,r,c).
1181 [binary,610.1,61.2] Col(c,r,d1).
1375 [para_into,624.1.1,103.1.2] T(c1,d,b).
1377 [binary,640.1,15.1] E(d,c,d,c1).
1449 [hyper,16,588,660] E(c,d1,c1,d).
1496 [hyper,32,714,594] T(a,b,b1).
1499 [hyper,28,714,594] T(b,c1,b1).
1565 [ur,57,21,739] $ANS(CON2)| -E3(x,y,y,d,c,c1).
1641 [hyper,52,588,744,1] E3(c,d1,d,c,d,d1).
1651 [hyper,16,604,744] E(c,p,c,d1).
1654 [hyper,2,660,744] E(c1,d,c,d1).
1686 [binary,807.1,61.2] Col(c,d1,b).
1704 [binary,807.1,23.1] T(d1,c,b).
1798 [ur,57,21,830] $ANS(CON1)| -E3(x,y,y,c,d,d1).
1809 [binary,834.1,17.1] E(b1,c1,b,c).
1858 [hyper,2,947,1] E(b,d,b2,d1).
1905 [hyper,32,99,1012] T(a,b,b2).
1908 [hyper,28,99,1012] T(b,c,b2).
2027 [binary,1038.1,23.2,unit_del,807] T(d,e,d1).
2046 [binary,1038.2,23.1,unit_del,1704] T(d1,e,d).
2095 [hyper,16,743,1090] E(d1,c,p,c).
2382 [hyper,19,1015,1375,1449,598] E(c,b2,c1,b).
2403 [binary,1496.1,59.2] Col(a,b,b1).
2504 [binary,1499.1,23.1] T(b1,c1,b).
2567 [ur,52,1,640,1565] $ANS(CON2)| -E(d,d,c,c1).
2569 [binary,1641.1,57.2,unit_del,830] -T(c,d1,d)|$ANS(CON1).
2684 [hyper,9,1375,1704] T(c,ip(c1,d,b,d1,c),c1).
2713 [ur,52,743,1,1798] $ANS(CON1)| -E(c,c,d,d1).
2720 [binary,1809.1,14.1] E(b,c,b1,c1).
3124 [hyper,6,2095,1155,1,1651,610,1109] d1=c|E(r,p,e,d1).
3225 [hyper,26,2504,1375] T(b1,c1,d).
3286 [ur,16,18,2567] $ANS(CON2)| -E(x,x,c,c1).
3371 [para_into,2684.1.2,107.1.2] T(c,e,c1).
3396 [ur,16,18,2713] $ANS(CON1)| -E(x,x,d,d1).
3419 [hyper,19,1908,2504,2720,2382] E(b,b2,b1,b).
3443 [para_from,3124.1.1,2569.1.2,unit_del,24] $ANS(CON1)|E(r,p,e,d1).
3496 [hyper,19,3225,807,1809,1654] E(b1,d,b,d1).
3569 [binary,3371.1,23.1] T(c1,e,c).
3667 [binary,3419.1,17.1] E(b,b2,b,b1).
3710 [binary,3443.1,17.1] $ANS(CON1)|E(r,p,d1,e).

3713 [binary,3443.1,15.1] $ANS(CON1)|E(p,r,e,d1).
3823 [binary,3569.1,60.2] Col(c,c1,e).
3859 [hyper,33,3569,605] e=c|T(c1,c,p).
3914 [binary,3667.1,14.1] E(b,b1,b,b2).
3974 [hyper,2,3710,1] $ANS(CON1)|E(d1,e,p,r).
4007 [hyper,16,617,3713] $ANS(CON1)|E(r,q,e,d1).
4032 [binary,3859.2,59.2] e=c|Col(c1,c,p).
4158 [hyper,19,1496,1905,13,3914] E(a,b1,a,b2).
4220 [binary,4007.1,14.1] $ANS(CON1)|E(e,d1,r,q).
4355 [hyper,77,2403,4158,3914,unit_del,98] b1=b2.
4416 [hyper,16,3443,4220,factor_simp] $ANS(CON1)|E(r,p,r,q).
4440 [para_from,4355.1.1,3496.1.1] E(b2,d,b,d1).
4443 [para_from,4355.1.1,2504.1.1] T(b2,c1,b).
4558 [hyper,48,1908,4443,1,2382,1858,4440] IFS(b,c,b2,d,b2,c1,b,d1).
4624 [binary,4558.1,35.1] E(c,d,c1,d1).
4687 [hyper,2,4624,764] E(c1,d1,d1,c).
4689 [hyper,2,4624,660] E(c1,d1,c1,d).
4722 [hyper,2,4687,1] E(d1,c,d1,c1).
4741 [hyper,76,3823,588,4689] c=c1|E(e,d1,e,d).
4865 [hyper,48,2027,2027,13,13,1377,4722] IFS(d,e,d1,c,d,e,d1,c1).
4895 [para_from,4741.1.1,102.1.3,unit_del,585] $ANS(CON2)|E(e,d1,e,d).
4972 [binary,4865.1,35.1] E(e,c,e,c1).
4974 [binary,4895.1,17.1] $ANS(CON2)|E(e,d1,d,e).
5025 [hyper,2,4895,4220] $ANS(CON2)|E(e,d,r,q)|$ANS(CON1).
5088 [hyper,2,1179,4972] E(r,c,e,c1).
5091 [ur,19,3371,13,4972,3286] -T(c,e,c)|$ANS(CON2).
5127 [hyper,16,3713,4974] $ANS(CON2)|E(p,r,d,e)|$ANS(CON1).
5132 [ur,36,24,4974,1,3396] $ANS(CON2)| -T(d,d1,e)|$ANS(CON1).
5186 [hyper,6,3974,5025,2095,1179,2046,618,factor_simp] $ANS(CON2)|
$ANS(CON1)|d1=e|E(d,c,q,c).
5219 [ur,19,3371,1155,5088,3286] -T(c,r,c)|$ANS(CON2).
5285 [para_into,5091.1.1,4032.1.2,unit_del,24] $ANS(CON2)|Col(c1,c,p).
5313 [hyper,19,618,2027,5127,4007,factor_simp] $ANS(CON2)|$ANS(CON1)|E(p,q,d,d1).
5435 [para_from,5186.1.1,5132.2.2,unit_del,21,factor_simp,factor_simp]
$ANS(CON2)|$ANS(CON1)|E(d,c,q,c).
5494 [binary,5285.1,65.1] $ANS(CON2)|Col(c,c1,p).
5709 [ur,16,5313,3396,factor_simp] $ANS(CON2)|$ANS(CON1)| -E(x,x,p,q).
5772 [binary,5435.1,17.1] $ANS(CON2)|$ANS(CON1)|E(d,c,c,q).
5949 [hyper,16,1065,5772] $ANS(CON2)|$ANS(CON1)|E(c,p,c,q).
6043 [hyper,76,1181,5949,4416,factor_simp] $ANS(CON2)|$ANS(CON1)|c=r|E(d1,p,d1,q).
6082 [para_from,6043.1.1,5219.1.3,unit_del,21,factor_simp] $ANS(CON2)|$ANS(CON1)|E(d1,p,d1,q).
6105 [hyper,76,1686,5949,6082,factor_simp,factor_simp] $ANS(CON2)|$ANS(CON1)|c=d1|E(b,p,b,q).
6106 [hyper,76,765,5949,6082,factor_simp,factor_simp] $ANS(CON2)|$ANS(CON1)|c=d1|E(a,p,a,q).
6148 [para_from,6105.1.1,2569.1.1,unit_del,24,factor_simp] $ANS(CON1)|$ANS(CON2)|E(b,p,b,q).
6157 [para_from,6106.1.1,2569.1.1,unit_del,24,factor_simp] $ANS(CON1)|$ANS(CON2)|E(a,p,a,q).
6298 [hyper,6,13,13,6157,6148,714,714,unit_del,98,factor_simp,factor_simp]
$ANS(CON1)|$ANS(CON2)|E(c1,p,c1,q).
6342 [hyper,76,5494,5949,6298,unit_del,5709,factor_simp,factor_simp,
factor_simp,factor_simp,factor_simp] $ANS(CON1)|$ANS(CON2)|c=c1.
6409 [para_from,6342.1.1,5091.1.3,factor_simp] -T(c,e,c1)|$ANS(CON2)|$ANS(CON1).
6410 [binary,6409.1,3371.1] $ANS(CON2)|$ANS(CON1).

With the input file that led to this proof forthcoming very soon, a few notes about this proof and a bit about how it was finally found are in order. Ancestor subsumption was in use. The output file displays proof lengths, in order, of 43, 51, 52, 125, 125, 122, 122, and 125. Yes, something was up. In view of the cited proof lengths, have you guessed what was happening in the attempt to find this 122-step proof? You have just a couple of seconds before learning of the truth, as Overbeek would say.

By the way, the proof I just included is the first of the two 122-step proofs. The first three proofs are forward, proofs of three of the unwanted Beeson lemmas; the last five are bidirectional, proofs of connectivity. The lengths of the first three proofs are each less than 60, whereas the lengths of the last five each are equal to or exceed 122. The answer to the small mystery about what was happening is that, in contrast to most of my experiments that proved connectivity of betweenness dependent, not all of the five lemmas (unwanted by Beeson) RG1 through RG5 were proved. Indeed, RG2, RG3, and RG5 were proved, but not 1 or 4. My guess as to why they were not proved focuses on the presence of some demodulators in the following input file, the one that yielded the given 122-step proof. A glance at the just-given proof shows that eight points, defined by Beeson, were relied upon, which contrast with just five when the first proof of connectivity was found, a proof relying on all five of the so-called unwanted lemmas. The three points that are used in the 122-step proof, which are not used in the first proof (that of length 104), are those for b1, b2, and e. As for a comparison of the given 122-step proof with (what I am told is) the Szmielew proof or at least thirty-six steps are present in the Szmielew proof that are not among those of the 122-step proof. Clearly satisfying to me is this sharp difference between OTTER's proof and the proof in SST, illustrating some of what can happen with proof finding as opposed to proof checking.

Input File Producing the 122-Step Proof

```
% Tarski-Szmielew's axiom system
% T is Tarski's B, non-strict betweenness
% E is equidistance
% Names for the axioms as in SST.
% Tries to prove Satz 5.1 assuming things proved in sst5a.in and sst5a2.in and sst5a3.in as well as b!=c.
% Contains more subgoals in list(passive). These should suffice to finish off Satz 5.1.


set(hyper_res).
set(para_into).
set(para_from).
set(binary_res).
set(ur_res).
% set(unit_deletion).
set(order_history).
assign(report,5400).
assign(max_seconds,7).
assign(max_mem,840000).
clear(print_kept).
%set(very_verbose).
set(input_sos_first).
set(ancestor_subsume).
set(back_sub).
% set(sos_queue).



assign(max_weight,11).
assign(max_distinct_vars,4).
```

```
assign(pick_given_ratio,4).
assign(max_proofs,8).
assign(heat,0).
assign(bsub_hint_wt,1).
set(keep_hint_subsumers).

weight_list(pick_and_purge).
weight(junk,1000).
end_of_list.



list(usable).
E(x,y,y,x).                         % A1 from page 10 of sst
-E(x,y,z,v) | -E(x,y,z2,v2) | E(z,v,z2,v2).   % A2
-E(x,y,z,z) | x=y.                  % A3
T(x,y,ext(x,y,w,v)).                % A4, first half
E(y,ext(x,y,w,v),w,v).              % A4, second half
-E(x,y,x1,y1) | -E(y,z,y1,z1) | -E(x,v,x1,v1) | -E(y,v,y1,v1) |
-T(x,y,z) | -T(x1,y1,z1) | x=y | E(z,v,z1,v1).  % A5
-T(x,y,x) | x=y.                    % A6
-T(xa,xp,xc) | -T(xb,xq,xc) | T(xp,ip(xa,xp,xc,xb,xq),xb).
-T(xa,xp,xc) | -T(xb,xq,xc) | T(xq,ip(xa,xp,xc,xb,xq),xa).
-T(alpha,beta,gamma).  %A8,  three lines.
-T(beta,gamma,alpha).
-T(gamma,alpha,beta).
% We don't need more of Tarski's axioms than that here.
E(x,y,x,y).  % Satz 2.1
-E(xa,xb,xc,xd) | E(xc,xd,xa,xb).  % Satz 2.2
-E(xa,xb,xc,xd) | E(xb,xa,xc,xd).  % Satz 2.4
-E(xa,xb,xc,xd) | -E(xc,xd,xe,xf) | E(xa,xb,xe,xf).  % Satz 2.3
-E(xa,xb,xc,xd) | E(xa,xb,xd,xc).  % Satz 2.5
E(x,x,y,y).  % Satz 2.8
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xb,xa1,xb1) |
-E(xb,xc,xb1,xc1) | E(xa,xc,xa1,xc1).  % Satz 2.11
xq = xa | -T(xq,xa,u) | -E(xa,u,xc,xd) | ext(xq,xa,xc,xd) = u.    % Satz 2.12
T(x,y,y).    % Satz 3.1
-E(u,v,x,x) | u=v.   % Not one of Szmielew's theorems but we proved it.
-T(xa,xb,xc) | T(xc,xb,xa).  % Satz 3.2.
T(xa,xa,xb).  % Satz 3.3
-T(xa,xb,xc) | -T(xb,xa,xc) | xa = xb.  % Satz 3.4.
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xb,xc).  % Satz 3.51.
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xc,xd).  % Satz 3.52.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd).  % Satz 3.61.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd).  % Satz 3.61.
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd).  % Satz 3.71
alpha != beta.   %related to Satz 3.14;  easily provable if added to sst 3h.in.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd).  % Satz 3.62.
 -T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd).  % Satz 3.71
 -T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd).  % Satz 3.72
-IFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xd,xb1,xd1).  % Satz 4.2
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xc,xa1,xc1)
| -E(xb,xc,xb1,xc1) | E(xa,xb,xa1,xb1).  % Satz 4.3
```

```
 alpha != beta.  % Satz 3.13
 beta != gamma.
 alpha != gamma.
T(xa,xb,ext(xa,xb,alpha,gamma)).   % Satz 3.14, first half
xb != ext(xa,xb,alpha,gamma).    % Satz 3.14, second half
 % The following many clauses are Definition 4.1
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | T(xa,xb,xc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | T(za,zb,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xa,xc,za,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xb,xc,zb,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xa,xd,za,zd).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xc,xd,zc,zd).
-T(xa,xb,xc) | -T(za,zb,zc) | -E(xa,xc,za,zc) | -E(xb,xc,zb,zc)
| -E(xa,xd,za,zd) | -E(xc,xd,zc,zd) | IFS(xa,xb,xc,xd,za,zb,zc,zd).

% Following 4 are definition 4.4 for n=3
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa1,xa2,xb1,xb2).
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa1,xa3,xb1,xb3).
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa2,xa3,xb2,xb3).
-E(xa1,xa2,xb1,xb2) | -E(xa1,xa3,xb1,xb3) | -E(xa2,xa3,xb2,xb3)
| E3(xa1,xa2,xa3,xb1,xb2,xb3).
% Following three lines are Satz 4.5
-T(xa,xb,xc) | -E(xa,xc,xa1,xc1) | T(xa1,insert(xa,xb,xa1,xc1),xc1).
-T(xa,xb,xc) | -E(xa,xc,xa1,xc1) | E3(xa,xb,xc,xa1,insert(xa,xb,xa1,xc1),xc1).
insert(xa,xb,xa1,xc1) = ext(ext(xc1,xa1,alpha,gamma),xa1,xa,xb).
-E3(x,y,z,u,v,w) | E3(x,z,y,u,w,v).   % See sst4q.in, not in Szmielew
-T(xa,xb,xc) | -E3(xa,xb,xc,xa1,xb1,xc1) | T(xa1,xb1,xc1).  % Satz 4.6

% following is Definition 4.10
-Col(xa,xb,xc) | T(xa,xb,xc) | T(xb,xc,xa) | T(xc,xa,xb).
Col(xa,xb,xc) | -T(xa,xb,xc).
Col(xa,xb,xc) | -T(xb,xc,xa).
Col(xa,xb,xc) | -T(xc,xa,xb).
% Following are Satz 4.11
-Col(x,y,z) | Col(y,z,x).
-Col(x,y,z) | Col(z,x,y).
-Col(x,y,z) | Col(z,y,x).
-Col(x,y,z) | Col(y,x,z).
-Col(x,y,z) | Col(x,z,y).
% following is Satz 4.12
Col(x,x,y).
% following is Satz 4.13
-Col(xa,xb,xc) | - E3(xa,xb,xc,xa1,xb1,xc1) | Col(xa1,xb1,xc1).
% following is Satz 4.14
-Col(xa,xb,xc) | -E(xa,xb,xa1,xb1)
| E3(xa,xb,xc,xa1,xb1,insert5(xa,xb,xc,xa1,xb1)).
% following is Definition 4.15
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | Col(xa,xb,xc).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E3(xa,xb,xc,xa1,xb1,xc1).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xa,xd,xa1,xd1).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xd,xb1,xd1).
-Col(xa,xb,xc) | - E3(xa,xb,xc,xa1,xb1,xc1) | - E(xa,xd,xa1,xd1)
  | -E(xb,xd,xb1,xd1) | FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1).
```

```
% Following is Satz 4.16
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | xa = xb | E(xc,xd,xc1,xd1).
% Following is Satz 4.17
xa = xb | -Col(xa,xb,xc) | -E(xa,xp,xa,xq) | -E(xb,xp,xb,xq) | E(xc,xp,xc,xq).
% Following is Satz 4.18
xa = xb | -Col(xa,xb,xc) | -E(xa,xc,xa,xc1) | -E(xb,xc,xb,xc1) | xc = xc1.
% Following is Satz 4.19
-T(xa,xc,xb) | -E(xa,xc,xa,xc1) | -E(xb,xc,xb,xc1) | xc = xc1.
% Following defines T5
T5(x,y,z,u,v) | -T(x,y,z) | -T(x,y,u) | -T(x,y,v)
| -T(x,z,u) | -T(x,z,v) | -T(x,u,v).
-T5(x,y,z,u,v) | T(x,y,z).
-T5(x,y,z,u,v) | T(x,y,u).
-T5(x,y,z,u,v) | T(x,y,v).
-T5(x,y,z,u,v) | T(x,z,u).
-T5(x,y,z,u,v) | T(x,z,v).
-T5(x,y,z,u,v) | T(x,u,v).
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xb,xa1,xb1)
 | -E(xb,xc,xb1,xc1) | -E(xa,xd,xa1,xd1) | -E(xb,xd,xb1,xd1)
| AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | T(xa,xb,xc).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | T(xa1,xb1,xc1).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xa,xb,xa1,xb1).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xc,xb1,xc1).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xd,xb1,xd1).
-AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | AFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1).
% Following proved in sst5a.in
%  T5(a,b,c,d1,b2).
%  T5(a,b,d,c1,b1).
%  E(b,c1,b2,c).
%  E(b,b1,b2,b).
%  b2 = b1.
%  T(c,e,c1).
%  T(d,e,d1).
% following proved in sst5a2.in
%  E(c1,d1,c,d).   % by cases according as b = c or b != c.
%  IFS(d,e,d1,c,d,e,d1,c1).
%  IFS(c,e,c1,d,c,e,c1,d1).
% Following proved in sst5a3.in
%  E(e,c,e,c1).
%  E(e,d,e,d1).
end_of_list.

list(passive).
-AFS(b,c,d1,c1,b1,c1,d,c) |  $ANS(TARG1INT).
-E(c1,d1,c,d) |  $ANS(TARG2INT).
-IFS(d,e,d1,c,d,e,d1,c1) |  $ANS(TARG3INT).
-IFS(c,e,c1,d,c,e,c1,d1) |  $ANS(TARG4INT).
-E(e,c,e,c1) |  $ANS(TARG5INT).
%  %  following 10 are later steps in proofs of 38 and F, temp.beeson.sst5a4.out26
end_of_list.

list(sos).
```

```
% c != c1.  % we assume this; separately we should
        %prove the theorem in case c = c1, which should be easy.
a != b.
T(a,b,c).
T(a,b,d).
-T(a,c,d) | $ANS(CON1).
-T(a,d,c) | $ANS(CON2).
c1 = ext(a,d,c,d).
d1 = ext(a,c,c,d).
b1 = ext(a,c1,c,b).
b2 = ext(a,d1,d,b).
e = ip(c1,d,b,d1,c).
p = ext(e,c,c,d).
r = ext(d1,c,e,c).
q = ext(p,r,p,r).
% c != b.
% c != c1.
% c != d1.
end_of_list.

list(hot).
end_of_list.

list(hints2).
% following 123 prove con, without c!=b used
T(b,d,ext(a,d,x,y)).
E(d,c1,c,d).
T(a,d,c1).
E(c,d1,c,d).
T(a,c,d1).
E(c1,b1,c,b).
T(a,c1,b1).
E(d1,b2,d,b).
T(a,d1,b2).
e=ip(ext(a,d,c,d),d,b,d1,c).
E(c,p,c,d).
T(e,c,p).
E(c,r,e,c).
T(d1,c,r).
E(r,q,p,r).
T(p,r,q).
T(ext(a,d,x,y),d,b).
E(c,d,d,c1).
E(c,d,c1,d).
T(a,b,c1).
-T(d,c,c1)|$ANS(CON2).
E(d1,c,c,d).
E(c,d,c,d1).
E(c,d,d1,c).
Col(c,d1,a).
T(b,c,d1).
-T(c,d,d1)|$ANS(CON1).
E(b1,c1,c,b).
```

E(d1,b2,b,d).
T(a,c,b2).
T(c,d1,b2).
-T(d1,c,b)|T(d,e,d1).
E(c,p,d,c).
E(c,d,p,c).
T(p,c,e).
E(c,r,c,e).
E(e,c,r,c).
Col(c,r,d1).
E(r,q,r,p).
T(c1,d,b).
E(d,c,d,c1).
E(c,d1,c1,d).
T(a,b,b1).
T(b,c1,b1).
$ANS(CON2)| -E3(x,y,y,d,c,c1).
E(c,p,c,d1).
E(c1,d,c,d1).
Col(c,d1,b).
T(d1,c,b).
$ANS(CON1)| -E3(x,y,y,c,d,d1).
E(b1,c1,b,c).
E(b,d,b2,d1).
T(a,b,b2).
T(b,c,b2).
T(d,e,d1).
T(d1,e,d).
E(d1,c,p,c).
E(r,p,r,q).
E(c,b2,c1,b).
Col(a,b,b1).
T(b1,c1,b).
$ANS(CON2)| -E(d,d,c,c1).
T(c,ip(c1,d,b,d1,c),c1).
$ANS(CON1)| -E(c,c,d,d1).
E(b,c,b1,c1).
d1=c|E(r,p,e,d1).
T(b1,c1,d).
$ANS(CON2)| -E(x,x,c,c1).
T(c,e,c1).
$ANS(CON1)| -E(x,x,d,d1).
E(b,b2,b1,b).
E(b1,d,b,d1).
$ANS(CON2)| -E(x,x,c1,c).
T(c1,e,c).
E(b,b2,b,b1).
Col(c,c1,e).
e=c|T(c1,c,p).
E(b,b1,b,b2).
e=c|Col(c1,c,p).
E(a,b1,a,b2).
b1=b2.

E(b2,d,b,d1).
T(b2,c1,b).
IFS(b,c,b2,d,b2,c1,b,d1).
E(c,d,c1,d1).
E(c1,d1,d1,c).
E(c1,d1,c1,d).
-T(c1,c,d1)|$ANS(CON2).
E(d1,c,d1,c1).
c=c1|E(e,d1,e,d).
$ANS(CON2)|E(r,p,e,d1).
IFS(d,e,d1,c,d,e,d1,c1).
$ANS(CON2)|E(e,d1,e,d).
$ANS(CON2)|E(r,p,d1,e).
$ANS(CON2)|E(p,r,e,d1).
$ANS(CON2)|E(r,q,e,d1).
$ANS(CON2)|E(e,d1,r,q).
E(e,c,e,c1).
$ANS(CON2)|E(e,d1,d,e).
$ANS(CON2)|E(d1,e,p,r).
$ANS(CON2)|E(e,d,r,q).
E(r,c,e,c1).
-T(c,e,c)|$ANS(CON2).
$ANS(CON2)|E(p,r,d,e).
$ANS(CON2)| -T(d,d1,e)|$ANS(CON1).
$ANS(CON2)|d1=e|E(d,c,q,c).
-T(c,r,c)|$ANS(CON2).
$ANS(CON2)|Col(c1,c,p).
$ANS(CON2)|E(p,q,d,d1).
$ANS(CON2)|$ANS(CON1)|E(d,c,q,c).
$ANS(CON2)|Col(c,c1,p).
$ANS(CON2)| -E(x,x,p,q)|$ANS(CON1).
$ANS(CON2)|$ANS(CON1)|E(d,c,c,q).
$ANS(CON2)|$ANS(CON1)|E(c,p,c,q).
$ANS(CON2)|$ANS(CON1)|c=r|E(d1,p,d1,q).
$ANS(CON2)|$ANS(CON1)|E(d1,p,d1,q).
$ANS(CON2)|$ANS(CON1)|c=d1|E(b,p,b,q).
$ANS(CON2)|$ANS(CON1)|c=d1|E(a,p,a,q).
$ANS(CON2)|$ANS(CON1)|E(b,p,b,q).
$ANS(CON2)|$ANS(CON1)|E(a,p,a,q).
$ANS(CON2)|$ANS(CON1)|E(c1,p,c1,q).
$ANS(CON2)|$ANS(CON1)|c=c1.
-T(c,e,c1)|$ANS(CON2)|$ANS(CON1).
% Following 135 prove conn of betw, may not be using trans of betw, temp.beeson.sst5a4.out40t 36ha
T(b,d,ext(a,d,x,y)).
E(d,c1,c,d).
T(a,d,c1).
E(c,d1,c,d).
T(a,c,d1).
E(c1,b1,c,b).
T(a,c1,b1).
E(d1,b2,d,b).
T(a,d1,b2).
e=ip(ext(a,d,c,d),d,b,d1,c).

E(c,p,c,d).
T(e,c,p).
E(c,r,e,c).
T(d1,c,r).
E(r,q,p,r).
T(p,r,q).
T(ext(a,d,x,y),d,b).
E(d,c1,d,c).
E(c,d,d,c1).
E(c,d,c1,d).
T(a,b,c1).
-T(d,c,c1)|$ANS(CON2).
E(d1,c,c,d).
E(c,d,c,d1).
E(c,d,d1,c).
Col(c,d1,a).
T(b,c,d1).
-T(c,d,d1)|$ANS(CON1).
E(b1,c1,c,b).
E(d1,b2,b,d).
T(a,c,b2).
T(c,d1,b2).
-T(d1,c,b)|T(d,e,d1).
E(c,p,d,c).
E(c,d,p,c).
T(p,c,e).
E(c,r,c,e).
E(r,c,e,c).
E(e,c,r,c).
Col(c,r,d1).
E(r,q,r,p).
T(c1,d,b).
E(d,c,d,c1).
E(c,d1,c1,d).
Col(a,b,c1).
T(a,b,b1).
T(b,c1,b1).
$ANS(CON2)| -E3(x,y,y,d,c,c1).
E3(c,d1,d,c,d,d1).
E(c,p,c,d1).
E(c1,d,c,d1).
Col(c,d1,b).
T(d1,c,b).
$ANS(CON1)| -E3(x,y,y,c,d,d1).
E(b1,c1,b,c).
T(a,b,b2).
T(b,c,b2).
E(d1,c,p,c).
E(r,p,r,q).
E(c,b2,c1,b).
Col(a,b,b1).
T(b1,c1,b).
$ANS(CON2)| -E(d,d,c,c1).

-T(c,d1,d)|$ANS(CON1).
T(c,ip(c1,d,b,d1,c),c1).
$ANS(CON1)| -E(c,c,d,d1).
E(b,c,b1,c1).
d1=c|E(r,p,e,d1).
T(b1,c1,d).
$ANS(CON2)| -E(x,x,c,c1).
T(c,e,c1).
$ANS(CON1)| -E(x,x,d,d1).
E(b,b2,b1,b).
$ANS(CON1)|E(r,p,e,d1).
E(b1,d,b,d1).
$ANS(CON2)| -E(x,x,c1,c).
$ANS(CON2)| -E(c1,c,x,x).
T(c1,e,c).
E(b,b2,b,b1).
$ANS(CON1)|E(r,p,d1,e).
$ANS(CON1)|E(p,r,e,d1).
$ANS(CON1)|E(r,q,e,d1).
$ANS(CON1)|E(e,d1,r,q).
Col(c,c1,e).
e=c|T(c1,c,p).
E(b,b1,b,b2).
e=c|Col(c1,c,p).
E(a,b1,a,b2).
b1=b2.
E(b2,d,b,d1).
T(b2,c1,b).
E(b,d,b2,d1).
IFS(b,c,b2,d,b2,c1,b,d1).
E(c,d,c1,d1).
$ANS(CON1)|E(d1,e,p,r).
E(c1,d1,d1,c).
E(c1,d1,c1,d).
-T(c1,c,d1)|$ANS(CON2).
E(d1,c,d1,c1).
c=c1|E(e,d1,e,d).
$ANS(CON2)| -T(d1,c,c1).
$ANS(CON2)|E(e,d1,e,d).
$ANS(CON2)|E(e,d1,d,e).
$ANS(CON2)|E(e,d,r,q)|$ANS(CON1).
$ANS(CON2)|E(p,r,d,e)|$ANS(CON1).
$ANS(CON2)| -T(d,d1,e)|$ANS(CON1).
$ANS(CON2)|$ANS(CON1)| -T(d1,e,d1).
T(d,e,d1).
T(d1,e,d).
IFS(d,e,d1,c,d,e,d1,c1).
E(p,q,d,d1)|$ANS(CON2)|$ANS(CON1).
d1=e|E(d,c,q,c)|$ANS(CON1)|$ANS(CON2).
E(e,c,e,c1).
$ANS(CON2)|$ANS(CON1)| -E(x,x,p,q).
$ANS(CON2)|$ANS(CON1)|E(d,c,q,c).
$ANS(CON2)|$ANS(CON1)|E(d,c,c,q).

```
E(r,c,e,c1).
-T(c,e,c)|$ANS(CON2).
$ANS(CON2)|Col(c1,c,p).
$ANS(CON2)|T(c1,c,p).
$ANS(CON2)|Col(c,c1,p).
$ANS(CON2)|T(p,c,c1).
$ANS(CON2)| -T(p,d1,c).
$ANS(CON2)|$ANS(CON1)|E(c,p,c,q).
$ANS(CON2)|$ANS(CON1)|c=r|E(d1,p,d1,q).
-T(c,r,c)|$ANS(CON2).
$ANS(CON2)|$ANS(CON1)|E(d1,p,d1,q).
$ANS(CON2)|$ANS(CON1)|c=d1|E(b,p,b,q).
$ANS(CON2)|$ANS(CON1)|c=d1|E(a,p,a,q).
$ANS(CON2)|$ANS(CON1)|E(b,p,b,q).
$ANS(CON2)|$ANS(CON1)|E(a,p,a,q).
$ANS(CON2)|$ANS(CON1)|E(c1,p,c1,q).
$ANS(CON2)|$ANS(CON1)|c=c1.
$ANS(CON2)|$ANS(CON1)| -E(c,p,c1,q).
$ANS(CON2)|$ANS(CON1)|E(c,p,c1,q).
end_of_list.

list(demodulators).
EQ(E(r,q,r,p),junk).
EQ(E(r,c,e,c),junk).
%  EQ(T(b,d,c1),junk).
%  EQ(Col(a,b,c1),junk).
%  EQ(E3(c,d1,d,c,d,d1),junk).
EQ(E(d,c1,d,c),junk).
%%EQ(Col(c,d1,a),junk).
%%EQ(E(r,c,e,c),junk).
%%EQ(Col(a,b,c1),junk).
%%EQ(E3(c,d1,d,c,d,d1),junk).
%  EQ(T(b,d,c1),junk).
%  EQ(E(c,e,e,c1),junk).
%  EQ(T(d1,c,a),junk).
%  EQ(T(b1,c1,a),junk).
%  EQ(E(d,c1,d1,c),junk).
%  EQ(E(c,r,e,c1),junk).
end_of_list.
```

The following three demodulators, found in the just-given file, might indeed prevent OTTER from deducing RG1 and RG4.

```
EQ(E(r,q,r,p),junk).
EQ(E(r,c,e,c),junk).
EQ(E(d,c1,d,c),junk).
```

These demodulators were used in the iterative process that eventually culminated in the 122-step proof but that were commented out in the end in order to enable OTTER to offer this cited proof. Or, perhaps, the program would have proved those two Beeson unwanted lemmas, RG1 and RG4, if allowed more latitude in various ways. A scholar would most likely investigate this question; I will not, at least at this time. You see that since both clauses for inner Pasch are used explicitly, blocking either or both from use might lead to an interesting proof.

With the use of ancestor subsumption, as you see, you are likely to find more than one proof of any given target in the output file. Such was not the case with the three Beeson lemmas, but was, as shown, the case with connectivity. Had I not included the use of ancestor subsumption, what do you think would have occurred? I and you are about to learn, for I am at this writing running the appropriate input file. I still expect the program to find more than one proof of connectivity because of seeking, in contrast to the five Beeson lemmas, a bidirectional proof.

A winner, if the prize is given for the use of ancestor subsumption. Specifically, the program found proofs of respective lengths 43, 51, 52, 131, 131, 129, 129, and 129. The world still turns: multiple proofs for a target when the choice is bidirectional; shorter proofs when ancestor subsumption is in use. You may have paused to study the 122-step proof and observed that both transitivity and Satz 4.3 are back, are used to find the proof and are found before the deduced clauses are presented. And what amounts to an aside in that all the votes are not yet in, you might enjoy watching (so to speak) research as it occurs. In particular, I am currently running an experiment that closely resembles that which yielded the 122-step proof; namely, I made one change, that of avoiding the use of binary resolution (in my continued tribute to Overbeek, and out of curiosity). I am also running a second experiment, one in which both binary resolution and ancestor subsumption are not in use.

After much CPU time, neither experiment succeeded. Therefore, apparently, binary resolution is needed under reasonable conditions, depending on the theorem to be proved. Of course, appropriate value assignments and parameter settings might lead to a proof in which binary resolution was not used. I leave that study to others. Instead, I now turn to a topic that may indeed remind you of proof shortening.

The OTTER proofs, as seen throughout this notebook, present formulas or equations (clauses) that are taken from an input file before presenting any of the deduced formulas or equations that make up the proof. Alama, if I understand correctly, is most interested in axiom dependence. Many years ago, McCune also made such a study, focusing (if memory serves) on John Kalman's right group calculus. Kalman's axiom system consists of five members; McCune was able to prove four of them dependent, as I recall, depending on which of the five (as it turns out) is chosen to be proved to be a single axiom. (I extended McCune's successful study, in a paper I wrote on the cramming strategy; three of the five can serve as a single axiom, if I have checked my work correctly.) Perhaps Alama's interest, communicated to me by e-mail, served as a wellspring, causing me to consider the following.

In some of the proofs I had of the dependence of connectivity of betweenness (where inner Pasch was present), forty-five or more elements from the corresponding input file were used to obtain the proof. I became interested in what might be called axiom-and-theorem dependence. In particular, the items that preceded the deduced clauses in many of the studies reported here consist of Tarski axioms, from one of such systems, followed by theorems that had already been proved, some by me, some by Beeson, and some by our collaboration. Those theorems were taken from Szmielew.

I began a series of experiments, focusing on the connectivity theorem, to see whether I could find a proof that relied on fewer than forty-five items; and, yes, I did find such a proof, one that depends on just forty-two. My approach was indeed straightforward: Take an input file that proves the dependence of connectivity, choose an item from the proof that occurs in the input file, comment it out (so it is inaccessible as a so-called axiom), and see what happens. Have you formed an opinion about what might occur? Also, have you made your choice for an item, axiom, or proved theorem to avoid using among that which the input file offers? Well, OTTER might fail to get any proof for connectivity, if, for example, the item being blocked is crucial to all proofs. Or, the resulting proof might be longer, in deduced steps, than the one in hand, which would not be a tragedy in that proof length is not the object at this stage. Or—annoying or disappointing—the program might complete a number of proofs, but each might rely on too many input items, axioms or proved theorems.

Because of what you have read so far, you have chosen, perhaps, transitivity of betweenness to block. That was my choice, having in hand a proof of length 151, a proof that relied on transitivity of betweenness, as a proved theorem. All was right on the ship; indeed, OTTER presented me with a proof that relied on forty-four items from the input file, omitting, as demanded, the use of transitivity. The proof—and here you encounter that oddity again—of length not longer than 151, but, instead, of length 146.

You can imagine my wandering around in the maze of items that could be blocked from use, axioms or proved theorems. Some of the experiments produced no proofs. But, rather than my detailing all the ports of call, a summary of where things now stand is in order. At this time, I have a proof that relies on just forty-two axioms and proved theorems. The proof, of length 151, was completed in approximately 133 CPU-seconds with retention of clause (104910). The proof is, of course, bidirectional. (Someday, perhaps, I will have in hand from some source a forward proof of the dependence of connectivity; one of my readers might send me such.) The proof completes, to my amusement, with the deduction of a clause that conflicts with T(x,y,y). The following three items were used in the proof relying on forty-five input items and absent from the proof relying on forty-two items.

-T(xa,xb,xd)| -T(xb,xc,xd)|T(xa,xb,xc).
-T(xa,xb,xc)| -T(xa1,xb1,xc1)| -E(xa,xc,xa1,xc1)| -E(xb,xc,xb1,xc1)|E(xa,xb,xa1,xb1).
-Col(x,y,z)|Col(y,x,z).

To obtain this result, I had OTTER avoid the use, in the input, of 3.51 (transitivity), Satz 4.3, Satz 4.19, and the five items listed under Satz 4.11. To remove confusion, not all of the cited items were used to produce the proof relying on forty-five items; some came and went along my trail of experiments. Yes, I have tried repeatedly to find a proof that relies on strictly fewer than forty-two input items, axioms and proved theorems, but to no avail. Some of my experiments led to the finding, after a long, long, long CPU time, of a long, long, long proof, but a proof that did not satisfy the objective. Indeed, one of the proofs has length 246, completed in approximately 47,807 CPU-seconds, with retention of clause (1495163). That experiment had OTTER avoid the use of both T(x,x,y) and T(x,y,y). And, so piquant and somewhat startling, rather than less than forty-five items to be used, forty-nine were used. When I tried to combine goals, seeking a proof that was short and that depended on perhaps fewer than forty-two input items, focusing on the already-presented 122-step proof, the best that I could do after a bit of experimentation was a proof of length 129. In a different experiment, I did find a proof of length 122 that relied on but forty-four input items, in contrast to the forty-five relied upon by the given 122-step proof. The input item that can be dispensed with is the following.

-Col(x,y,z)|Col(y,x,z).

At this point on this labyrinthian journey, I now turn to a study that is more in the spirit of proof checking, according to Beeson and, I assume, others, in contrast to proof finding.

19. Szmielew Plays a Bigger Role, Deducing Outer Pasch

You may well ask why I would focus on proof checking in view of my earlier remarks. The explanation is, in a sense, sad and yet practical. You see, three theorems remain on my (so-to-speak) private list for discussion in this notebook. Specifically, I would like to focus on the theorem that asserts that outer Pasch can be proved from inner Pasch, that inner Pasch can be proved from outer Pasch, and that connectivity is dependent on the axiom system offered in Section 1, the system that relies on outer Pasch rather than on inner. My attempts so far have led to essentially nothing in the context of each of these three theorems. However, I can now focus on the first of the cited three, a deduction of outer Pasch from inner Pasch.

I am able to do so, at the simplest level, because of an input file sent to me by Beeson. At a deeper level, from what he writes, Gupta merits all the credit. Indeed, hus proof provided the necessary steps, to be used as resonators or as hints, that could be used by OTTER to successfully prove the theorem in focus. Yes, so much was taken from Gupta's proof that the study must be termed proof checking. So, you further ask—even though the theorem has been on my list from the outset—why feature it at all. Simply, you may advise, present an input file and a proof. Rather than following that program, I will be concerned here with two aspects. First, I will again discuss proof shortening, of this theorem. Second, in the Alama spirit touched on in Section 1, I will spend some time on minimizing the number of assumptions needed to prove the theorem.

So that you (and Alama and his possible colleagues) can proceed on your own in the context of proof shortening and assumption reduction, I now present the input file sent to me by e-mail from Beeson. (By the way, for the curious who might wish to have even more motivation for studying assumption reduction, you might glance at various theorems that hold for group theory, but also hold for semigroups.)

An Input File for Proving Outer Pasch from Inner

```
% Tarski-Szmielew's axiom system
% T is Tarski's B, non-strict betweenness
% E is equidistance
% Names for the axioms as in SST.
% Assumes key parts of earlier chapters and proves outer Pasch (Satz 9.6)


set(hyper_res).
set(para_into).
set(para_from).
set(binary_res).
set(ur_res).
% set(unit_deletion).
set(order_history).
assign(report,5400).
assign(max_seconds,10000).
assign(max_mem,840000).
%clear(print_kept).
%set(very_verbose).
set(input_sos_first).
set(ancestor_subsume).
% set(sos_queue).



assign(max_weight,12).
assign(max_distinct_vars,5).
assign(pick_given_ratio,4).
assign(max_proofs,1).


weight_list(pick_and_purge).
weight(-T(c,p,q),-1).
weight(-T(q,c,p),-1).
weight(-T(p,q,c),-1).
weight(-opposite(a,p,q,b)|Col(p,q,d),-1).
weight(-opposite(a,p,q,b)|T(a,d,b),-1).
weight(-T(p,c,a)| -T(b,d,a)|T(d,t,p),-1).
weight(-T(p,c,a)| -T(b,d,a)|T(c,t,b),-1).
weight(Col(c,p,a),-1).
weight(T(p,c,a),-1).
weight(Col(c,b,q),-1).
weight(-T(x,b,q)|b=q|T(x,b,c),-1).
weight(T(c,q,b),-1).
weight(-T(p,q,b),-1).
weight(-T(a,x,b)| -Col(p,q,x)|T(q,x,p)|T(x,p,q),-1).
weight(-T(a,q,b),-1).
weight(-opposite(a,p,q,b)| -T(p,q,d),-1).
weight(-opposite(a,p,q,b)|T(b,d,a),-1).
```

```
weight(T(c,p,a)|sameside(c,p,a),-1).
weight(Col(c,a,p),-1).
weight(Col(b,c,q),-1).
weight(T(c,a,p)|sameside(c,a,p),-1).
weight(T(d,t,p)| -T(a,d,b),-1).
weight(-T(a,d,b)|T(p,t,d),-1).
weight(T(d,t,p)| -opposite(a,p,q,b),-1).
weight(T(p,t,d)| -opposite(a,p,q,b),-1).
weight(-T(q,p,q),-1).
weight(-T(p,q,p),-1).
weight(sameside(c,p,a)|p=c,-1).
weight(-T(q,c,q),-1).
weight(-T(a,q,c),-1).
weight(-T(b,c,p),-1).
weight(sameside(c,p,a),-1).
weight(c!=p,-1).
weight(sameside(c,a,p)|a=c,-1).
weight(-T(b,p,q),-1).
weight(-T(p,b,c),-1).
weight(sameside(c,a,p),-1).
weight(c!=a,-1).
weight(T(c,t,b)| -opposite(a,p,q,b),-1).
weight(-opposite(a,p,q,b)|Col(b,c,t),-1).
weight(-opposite(a,p,q,b)| -opposite(x,b,c,t),-1).
weight(-Col(p,q,b)|sameside(p,q,b),-1).
weight(-Col(p,q,b)|T(q,b,p),-1).
weight(-T(p,c,b),-1).
weight(-T(b,q,b),-1).
weight(-Col(p,q,b)|b!=q,-1).
weight(-Col(p,q,b)|T(p,b,q),-1).
weight(-T(b,q,a),-1).
weight(-T(a,c,b),-1).
weight(b=q| -Col(p,q,b),-1).
weight(-Col(p,q,b),-1).
weight(-Col(q,p,q)|sameside(q,p,q),-1).
weight(sameside(q,p,q),-1).
weight(Col(q,p,q),-1).
weight(Col(p,q,q),-1).
weight(p=q|opposite(c,p,q,b),-1).
weight(opposite(c,p,q,b),-1).
weight(-Col(p,q,p)|sameside(p,q,p),-1).
weight(sameside(p,q,p),-1).
weight(Col(p,q,p),-1).
weight(opposite(a,p,q,b),-1).
weight(T(p,t,d),-1).
weight(T(d,t,p),-1).
weight(-T(p,q,d),-1).
weight(T(a,d,b),-1).
weight(Col(p,q,d),-1).
weight(-T(b,c,q),-1).
weight(-T(a,b,c),-1).
weight(-opposite(t,b,c,x),-1).
weight(-T(c,a,b),-1).
```

```
weight(-T(c,b,q),-1).
weight(-T(b,c,a),-1).
weight(-T(c,p,b),-1).
weight(-T(p,q,t),-1).
weight(sameside(b,c,q),-1).
weight(b!=c,-1).
weight(-Col(b,c,a),-1).
weight(sameside(c,b,q),-1).
weight(sameside(c,b,c),-1).
weight(Col(c,b,c),-1).
weight(Col(b,c,c),-1).
weight(Col(b,c,p)|opposite(p,b,c,a),-1).
weight(opposite(p,b,c,a),-1).
weight(-sameside(p,q,t),-1).
weight(-Col(p,q,t),-1).
weight(-T(t,p,q),-1).
weight(-T(q,t,p),-1).
weight(-T(d,p,q),-1).
weight(-T(q,d,p),-1).
end_of_list.


list(usable).
E(x,y,y,x).                          % A1 from page 10 of sst
-E(x,y,z,v) | -E(x,y,z2,v2) | E(z,v,z2,v2).   % A2
-E(x,y,z,z) | x=y.                   % A3
T(x,y,ext(x,y,w,v)).                 % A4, first half
E(y,ext(x,y,w,v),w,v).               % A4, second half
-E(x,y,x1,y1) | -E(y,z,y1,z1) | -E(x,v,x1,v1) | -E(y,v,y1,v1) | -T(x,y,z) |
  -T(x1,y1,z1) | x=y | E(z,v,z1,v1).  % A5
-T(x,y,x) | x=y.                     % A6
% A7, inner Pasch, two clauses.
-T(xa,xp,xc) | -T(xb,xq,xc) | T(xp,ip(xa,xp,xc,xb,xq),xb).
-T(xa,xp,xc) | -T(xb,xq,xc) | T(xq,ip(xa,xp,xc,xb,xq),xa).
-T(alpha,beta,gamma).  %A8,  three lines.
-T(beta,gamma,alpha).
-T(gamma,alpha,beta).
% We don’t need more of Tarski’s axioms than that here.
E(x,y,x,y).  % Satz 2.1
-E(xa,xb,xc,xd) | E(xc,xd,xa,xb).  % Satz 2.2
-E(xa,xb,xc,xd) | E(xb,xa,xc,xd).  % Satz 2.4
-E(xa,xb,xc,xd) | -E(xc,xd,xe,xf) | E(xa,xb,xe,xf).  % Satz 2.3
-E(xa,xb,xc,xd) | E(xa,xb,xd,xc).  % Satz 2.5
E(x,x,y,y).  % Satz 2.8
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xb,xa1,xb1) | -E(xb,xc,xb1,xc1) | E(xa,xc,xa1,xc1).  % Satz 2.11
xq = xa | -T(xq,xa,u) | -E(xa,u,xc,xd) | ext(xq,xa,xc,xd) = u.    % Satz 2.12
T(x,y,y).    % Satz 3.1
-T(xa,xb,xc) | T(xc,xb,xa).  % Satz 3.2.
T(xa,xa,xb).  % Satz 3.3
-T(xa,xb,xc) | -T(xb,xa,xc) | xa = xb.  % Satz 3.4.
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xb,xc).  % Satz 3.51.
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xc,xd).  % Satz 3.52.
```

```
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd).  % Satz 3.61.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd).  % Satz 3.61.
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd).  % Satz 3.71
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd).  % Satz 3.62.
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd).  % Satz 3.71
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd).  % Satz 3.72
-IFS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xd,xb1,xd1).  % Satz 4.2
-T(xa,xb,xc) | -T(xa1,xb1,xc1) | -E(xa,xc,xa1,xc1) | -E(xb,xc,xb1,xc1) | E(xa,xb,xa1,xb1).  % Satz 4.3

alpha != beta.  % Satz 3.13
beta != gamma.
alpha != gamma.
T(xa,xb,ext(xa,xb,alpha,gamma)).   % Satz 3.14, first half
xb != ext(xa,xb,alpha,gamma).    % Satz 3.14, second half
% The following many clauses are Definition 4.1
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | T(xa,xb,xc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | T(za,zb,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xa,xc,za,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xb,xc,zb,zc).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xa,xd,za,zd).
-IFS(xa,xb,xc,xd,za,zb,zc,zd) | E(xc,xd,zc,zd).
-T(xa,xb,xc) | -T(za,zb,zc) | -E(xa,xc,za,zc) | -E(xb,xc,zb,zc) |
  -E(xa,xd,za,zd) | -E(xc,xd,zc,zd) | IFS(xa,xb,xc,xd,za,zb,zc,zd).

% Following 4 are definition 4.4 for n=3
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa1,xa2,xb1,xb2).
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa1,xa3,xb1,xb3).
-E3(xa1,xa2,xa3,xb1,xb2,xb3) | E(xa2,xa3,xb2,xb3).
-E(xa1,xa2,xb1,xb2) | -E(xa1,xa3,xb1,xb3) | -E(xa2,xa3,xb2,xb3) | E3(xa1,xa2,xa3,xb1,xb2,xb3).
% Following three lines are Satz 4.5
-T(xa,xb,xc) | -E(xa,xc,xa1,xc1) | T(xa1,insert(xa,xb,xa1,xc1),xc1).
-T(xa,xb,xc) | -E(xa,xc,xa1,xc1) | E3(xa,xb,xc,xa1,insert(xa,xb,xa1,xc1),xc1).
insert(xa,xb,xa1,xc1) = ext(ext(xc1,xa1,alpha,gamma),xa1,xa,xb).
-T(xa,xb,xc) | -E3(xa,xb,xc,xa1,xb1,xc1) | T(xa1,xb1,xc1).  % Satz 4.6

% following is Definition 4.10
-Col(xa,xb,xc) | T(xa,xb,xc) | T(xb,xc,xa) | T(xc,xa,xb).
Col(xa,xb,xc) | -T(xa,xb,xc).
Col(xa,xb,xc) | -T(xb,xc,xa).
Col(xa,xb,xc) | -T(xc,xa,xb).
% Following are Satz 4.11
-Col(x,y,z) | Col(y,z,x).
-Col(x,y,z) | Col(z,x,y).
-Col(x,y,z) | Col(z,y,x).
-Col(x,y,z) | Col(y,x,z).
-Col(x,y,z) | Col(x,z,y).
% following is Satz 4.12
Col(x,x,y).
% following is Satz 4.13
-Col(xa,xb,xc) | - E3(xa,xb,xc,xa1,xb1,xc1) | Col(xa1,xb1,xc1).
% following is Satz 4.14
-Col(xa,xb,xc) | -E(xa,xb,xa1,xb1) | E3(xa,xb,xc,xa1,xb1,insert5(xa,xb,xc,xa1,xb1)).
% following is Definition 4.15
```

```
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | Col(xa,xb,xc).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E3(xa,xb,xc,xa1,xb1,xc1).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xa,xd,xa1,xd1).
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | E(xb,xd,xb1,xd1).
-Col(xa,xb,xc) | - E3(xa,xb,xc,xa1,xb1,xc1) | - E(xa,xd,xa1,xd1) |
  -E(xb,xd,xb1,xd1) | FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1).
% Following is Satz 4.16
-FS(xa,xb,xc,xd,xa1,xb1,xc1,xd1) | xa = xb | E(xc,xd,xc1,xd1).
% Following is Satz 4.17
xa = xb | -Col(xa,xb,xc) | -E(xa,xp,xa,xq) | -E(xb,xp,xb,xq) | E(xc,xp,xc,xq).
% Following is Satz 4.18
xa = xb | -Col(xa,xb,xc) | -E(xa,xc,xa,xc1) | -E(xb,xc,xb,xc1) | xc = xc1.
% Following is Satz 4.19
-T(xa,xc,xb) | -E(xa,xc,xa,xc1) | -E(xb,xc,xb,xc1) | xc = xc1.
% Following is Satz 5.1
xa = xb | -T(xa,xb,xc) | -T(xa,xb,xd) | T(xa,xc,xd) | T(xa,xd,xc).
% Following is Satz 5.2
xa = xb | -T(xa,xb,xc) | -T(xa,xb,xd)| T(xb,xc,xd) | T(xb,xd,xc).
% Following is Satz 5.3
-T(xa,xb,xd) | -T(xa,xc,xd) | T(xa,xb,xc) | T(xa,xc,xb).
% Following is Definition 5.4
-T(xc,y,xd) | -E(xa,xb,xc,y) | le(xa,xb,xc,xd).
-le(xa,xb,xc,xd) | T(xc,insert(xa,xb,xc,xd),xd).  %ab inserted into cd
-le(xa,xb,xc,xd) | E(xa,xb,xc,insert(xa,xb,xc,xd)).
% Following is Satz 5.5a
-le(xa,xb,xc,xd) | T(xa,xb,insert(xc,xd,xa,xb)).
-le(xa,xb,xc,xd) | E(xa,insert(xc,xd,xa,xb),xc,xd).
% Following is Satz 5.5b
-T(xa,xb,x) | -E(xa,x,xc,xd) | le(xa,xb,xc,xd).
% Following is Satz 5.6
-le(xa,xb,xc,xd) | -E(xa,xb,xa1,xb1) | -E(xc,xd,xc1,xd1) | le(xa1,xb1,xc1,xd1).
% Following is Satz 5.7
le(xa,xb,xa,xb).
% Following is Satz 5.8
-le(xa,xb,xc,xd) | - le(xc,xd,xe,xf) | le(xa,xb,xe,xf).
% Following is Satz 5.9
-le(xa,xb,xc,xd) | -le(xc,xd,xa,xb) | E(xa,xb,xc,xd).
% Following is Satz 5.10
le(xa,xb,xc,xd) | le(xc,xd,xa,xb).
% Following is Satz 5.11
le(xa,xa,xc,xd).
% Following is Definition 6.1
sameside(xa,xp,xb) | xa=xp | xb = xp | -T(xp,xa,xb).
sameside(xa,xp,xb) | xa=xp | xb = xp | -T(xp,xb,xa).
-sameside(xa,xp,xb) | xa != xp.
-sameside(xa,xp,xb) | xb != xp.
-sameside(xa,xp,xb) | T(xp,xa,xb) | T(xp,xb,xa).
%Following is Satz 6.2
xa=xp | xb=xp | xc = xp | -T(xa,xp,xc) | -T(xb,xp,xc) | sameside(xa,xp,xb).
xa=xp | xb=xp | xc = xp | -T(xa,xp,xc) | T(xb,xp,xc) | -sameside(xa,xp,xb).
% following is Satz 6.3
-sameside(xa,xp,xb) | xa != xp.
-sameside(xa,xp,xb) | xb != xp.
```

```
-sameside(xa,xp,xb) | f63(xa,xp,xb) != xp.
-sameside(xa,xp,xb) | T(xa,xp,f63(xa,xp,xc)).
-sameside(xa,xp,xb) | T(xb,xp,f63(xa,xp,xc)).
xa = xp | xb = xp | xc = xp | -T(xa,xp,xc) | -T(xb,xp,xc) | sameside(xa,xp,xb).
% following is Satz 6.4
-sameside(xa,xp,xb) | Col(xa,xp,xb).
-sameside(xa,xp,xb) | -T(xa,xp,xb).
-Col(xa,xp,xb) | T(xa,xp,xb) | sameside(xa,xp,xb).
% following is Satz 6.5
xa=xp | sameside(xa,xp,xa).
% following is Definition 9.1
xp = xq | Col(xp,xq,xa) | Col(xp,xq,xb) | -T(xa,xt,xb) | -Col(xp,xq,xt) | opposite(xa,xp,xq,xb).
-opposite(xa,xp,xq,xb) | -Col(xp,xq,xa).
-opposite(xa,xp,xq,xb) | -Col(xp,xq,xb).
-opposite(xa,xp,xq,xb) | T(xa,il(xa,xb,xp,xq),xb).
-opposite(xa,xp,xq,xb) | Col(xp,xq,il(xa,xb,xp,xq)).
% following is Satz 9.2
-opposite(xa,xp,xq,xb) | opposite(xb,xp,xq,xa).
% following is Satz 9.5
-opposite(xa,xp,xq,xc) | -Col(xp,xq,xr) | -sameside(xa,xr,xb) | opposite(xb,xp,xq,xc).  % Satz 9.5
end_of_list.


list(passive).
%-opposite(c,p,q,b) | $ANS(1).
%-sameside(c,p,a) | $ANS(2).
%-opposite(a,p,q,b) | $ANS(3).
%t != q | $ANS(4).
end_of_list.

list(sos).
T(p,q,c) | -T(p,q,c).  % case 1a or Case 1b.
Col(p,c,q) | -Col(p,q,c).   % case 1 or case 2
Col(b,p,q) | -Col(b,p,q).    % case 2a or case 2b;
d = il(a,b,p,q).
t = ip(p,c,a,b,d).
T(a,c,p).
T(b,q,c).
-T(a,x,b) | -T(p,q,x).
end_of_list.
```

With the given input file, OTTER produces a proof of length 111 in less than 30 CPU-seconds. I note that, should you block the use of binary resolution, as I have just done at this very moment, you may not obtain a proof; I have not so far. However, perhaps that inference rule is in fact unnecessary, if you choose appropriate values to assign to the various parameters, if such exist. The 111-step proof relies on thirty-seven items from the input, displayed (as is the style of OTTER) before the deduced items are presented. Also, rather than UNIT CONFLICT, the proof completes with the deduction of the empty clause.

Yes, the use of this input file is indeed in the spirit of proof checking. I think it safe to say, that, without the provision of steps from Gupta's proof, none of my methods would have ever led to a proof of outer Pasch from inner Pasch. Some might then conclude that OTTER is not nearly as powerful as my notebooks strongly suggest it is. If you lean in that direction, you might, before taking a firm stand, review successes obtained with OTTER in various areas of abstract algebra and logic. In many, many cases, no guidance was provided, from a book or a paper. If you wonder why Tarskian geometry offers such often-impenetrable obstacles, I suggest (as Overbeek has) that the non-Hornness of the axioms is indeed a problem. In particular, when axioms are involved that contain more than one positive literal, the terrain is far rockier.

Of the two areas for study, proof shortening and assumption reduction, because of many years of research, I knew far more about the former than about the latter. Therefore, I turned immediately to finding a proof shorter than length 111. So typical of my studies, I decided to use otter-loop, a program (as noted) written for me by McCune to aid me in proof shortening. I planned to block each of the deduced steps of the 111-step proof to see what would occur. Blocking the last step ordinarily is fruitless in that it often is the final important step of the proof. But, since the 111-step proof required almost 30 CPU-seconds to complete, I needed to make some move that would reduce the time, not wishing to wait 111 times 30 seconds to see whether progress would occur. I turned to the inclusion of a hints list, placing all 111 deduced items in that list as hints, a move that reduced the CPU time to a few seconds to produce a proof.

I made a few modifications and additions to produce the input file I used to seek ever-shorter proofs, the following.

```
assign(max_seconds,2).
set(back_sub).
assign(max_distinct_vars,4).
assign(max_proofs,6).
assign(bsub_hint_wt,-1).
set(keep_hint_subsumers).
weight(junk,1000).

list(demodulators).
end_of_list).
```

The intention is to demodulate each step of the 111-step proof, one at a time, to junk, which will, because of the assignment of 1000 to junk, cause the item, when deduced, to be discarded in that the max_weight is assigned a value far smaller than 1000. The demodulator list is included to so-to-speak catch new demodulators, if and when I decided to adjoin such. With the use of ancestor subsumption, you should always include set(back_sub). Because the move of relying on hints sharply reduced the time to complete a proof, I assigned the small value 2 to max_seconds. I assigned the value 6 to max_proofs in that, when OTTER finds more than one proof of a given theorem, the proof lengths can vary. Further, you see two items focusing on hints, both of which had best be included when using hints; one or both can easily be forgotten.

How convenient to use otter-loop, for which I will always be indebted to McCune. One single instruction and, in this case, 111 jobs are run, one after the other. When one of the runs showed a reduction in proof length, I adjoined the corresponding demodulator, to block the indicated proof step. After iterating, I found the following two demodulators whose use enabled the program to present to me a 90-step proof deducing outer Pasch from inner Pasch.

```
EQ(Col(d,q,p),junk).
EQ(Col(c,b,q),junk).
```

The fourth proof, the following, of the six found with the amended file, has proof length 90, while the other five have proof lengths that vary from 91 to 93.

A 90-Step Proof of Inner Implies Outer Pasch

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Sun Mar 31 08:30:39 2013
The command was "otter". The process ID is 28254.
-----> EMPTY CLAUSE at   1.09 sec ----> 6392 [binary,6327.1,95.2,unit_del,3348,6350] $F.
```

Length of proof is 90. Level of proof is 19.

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

8 [] -T(xa,xp,xc)| -T(xb,xq,xc)|T(xp,ip(xa,xp,xc,xb,xq),xb).
9 [] -T(xa,xp,xc)| -T(xb,xq,xc)|T(xq,ip(xa,xp,xc,xb,xq),xa).
21 [] T(x,y,y).
22 [] -T(xa,xb,xc)|T(xc,xb,xa).
23 [] T(xa,xa,xb).
24 [] -T(xa,xb,xc)| -T(xb,xa,xc)|xa=xb.
25 [] -T(xa,xb,xd)| -T(xb,xc,xd)|T(xa,xb,xc).
26 [] -T(xa,xb,xd)| -T(xb,xc,xd)|T(xa,xc,xd).
27 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xb,xc,xd).
30 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xa,xb,xd).
32 [] -T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xb,xd).
55 [] -Col(xa,xb,xc)|T(xa,xb,xc)|T(xb,xc,xa)|T(xc,xa,xb).
57 [] Col(xa,xb,xc)| -T(xb,xc,xa).
58 [] Col(xa,xb,xc)| -T(xc,xa,xb).
62 [] -Col(x,y,z)|Col(y,x,z).
63 [] -Col(x,y,z)|Col(x,z,y).
64 [] Col(x,x,y).
78 [] -T(xa,xb,xd)| -T(xa,xc,xd)|T(xa,xb,xc)|T(xa,xc,xb).
95 [] -sameside(xa,xp,xb)|T(xp,xa,xb)|T(xp,xb,xa).
98 [] -sameside(xa,xp,xb)|xa!=xp.
99 [] -sameside(xa,xp,xb)|xb!=xp.
104 [] -sameside(xa,xp,xb)|Col(xa,xp,xb).
106 [] -Col(xa,xp,xb)|T(xa,xp,xb)|sameside(xa,xp,xb).
108 [] xp=xq|Col(xp,xq,xa)|Col(xp,xq,xb)| -T(xa,xt,xb)| -Col(xp,xq,xt)|opposite(xa,xp,xq,xb).
109 [] -opposite(xa,xp,xq,xb)| -Col(xp,xq,xa).
111 [] -opposite(xa,xp,xq,xb)|T(xa,il(xa,xb,xp,xq),xb).
112 [] -opposite(xa,xp,xq,xb)|Col(xp,xq,il(xa,xb,xp,xq)).
114 [] -opposite(xa,xp,xq,xc)| -Col(xp,xq,xr)| -sameside(xa,xr,xb)|opposite(xb,xp,xq,xc).
118 [] d=il(a,b,p,q).
119 [] t=ip(p,c,a,b,d).
120 [] T(a,c,p).
121 [] T(b,q,c).
122 [] -T(a,x,b)| -T(p,q,x).
247 [para_from,118.1.2,112.2.3] -opposite(a,p,q,b)|Col(p,q,d).
248 [para_from,118.1.2,111.2.2] -opposite(a,p,q,b)|T(a,d,b).
253 [para_from,119.1.2,9.3.2] -T(p,c,a)| -T(b,d,a)|T(d,t,p).
261 [binary,120.1,58.2] Col(c,p,a).
279 [binary,120.1,22.1] T(p,c,a).
344 [binary,121.1,22.1] T(c,q,b).
393 [binary,122.1,23.1] -T(p,q,a).
395 [binary,122.1,21.1] -T(p,q,b).
407 [binary,122.2,21.1] -T(a,q,b).
422 [binary,248.2,122.1] -opposite(a,p,q,b)| -T(p,q,d).
449 [binary,248.2,22.1] -opposite(a,p,q,b)|T(b,d,a).
485 [binary,253.2,22.2,unit_del,279] T(d,t,p)| -T(a,d,b).
503 [binary,261.1,106.1] T(c,p,a)|sameside(c,p,a).
579 [binary,344.1,57.2] Col(b,c,q).
658 [binary,393.1,22.2] -T(a,q,p).
668 [ur,30,279,393] -T(p,q,c).

669 [ur,30,23,393] -T(p,q,p).
672 [ur,27,23,393] -T(q,p,q).
673 [ur,26,279,393] -T(c,q,a).
694 [binary,395.1,22.2] -T(b,q,p).
706 [ur,27,344,395] -T(c,p,q).
728 [binary,407.1,22.2] -T(b,q,a).
740 [ur,27,344,407] -T(c,a,q).
747 [binary,422.2,106.2] -opposite(a,p,q,b)| -Col(p,q,d)|sameside(p,q,d).
761 [binary,422.2,22.2] -opposite(a,p,q,b)| -T(d,q,p).
798 [binary,449.2,9.2] -opposite(a,p,q,b)| -T(x,y,a)|T(d,ip(x,y,a,b,d),x).
800 [binary,449.2,8.2] -opposite(a,p,q,b)| -T(x,y,a)|T(y,ip(x,y,a,b,d),b).
832 [binary,485.1,22.1] -T(a,d,b)|T(p,t,d).
854 [binary,832.1,248.2] T(p,t,d)| -opposite(a,p,q,b).
957 [binary,503.1,24.2,unit_del,279] sameside(c,p,a)|p=c.
980 [binary,579.1,106.1] T(b,c,q)|sameside(b,c,q).
1019 [ur,26,120,658] -T(c,q,p).
1049 [ur,27,121,668] -T(b,p,q).
1050 [ur,26,121,668] -T(p,b,c).
1051 [ur,26,21,668] -T(c,q,c).
1057 [binary,669.1,106.2] -Col(p,q,p)|sameside(p,q,p).
1080 [binary,672.1,106.2] -Col(q,p,q)|sameside(q,p,q).
1125 [ur,30,344,673] -T(c,b,a).
1158 [ur,30,121,694] -T(b,c,p).
1248 [ur,30,121,728] -T(b,c,a).
1269 [ur,78,344,673,740] -T(c,a,b).
1292 [binary,747.2,247.2,factor_simp] -opposite(a,p,q,b)|sameside(p,q,d).
1338 [para_into,800.3.2,119.1.2,unit_del,279] -opposite(a,p,q,b)|T(c,t,b).
1387 [para_from,957.2.1,694.1.3,unit_del,121] sameside(c,p,a).
1404 [binary,980.1,22.1] sameside(b,c,q)|T(q,c,b).
1427 [ur,78,344,706,1019] -T(c,p,b).
1489 [binary,1050.1,55.4,unit_del,1158,1427] -Col(b,c,p).
1536 [ur,27,121,1051] -T(b,c,q).
1539 [ur,25,344,1051] -T(q,c,b).
1543 [binary,1057.1,63.2,unit_del,64] sameside(p,q,p).
1548 [binary,1080.1,63.2,unit_del,64] sameside(q,p,q).
1554 [binary,1125.1,95.2,unit_del,1269] -sameside(b,c,a).
1611 [ur,27,23,1158] -T(c,b,c).
1639 [binary,1248.1,106.2,unit_del,1554] -Col(b,c,a).
1722 [binary,1404.1,99.1,unit_del,1539] q!=c.
1814 [binary,1536.1,106.2,unit_del,579] sameside(b,c,q).
1838 [binary,1543.1,104.1] Col(p,q,p).
1841 [binary,1543.1,99.1] p!=q.
1870 [binary,1548.1,104.1] Col(q,p,q).
1893 [ur,32,121,1722,694] -T(q,c,p).
1934 [binary,1814.1,98.1] b!=c.
1998 [binary,1870.1,62.1] Col(p,q,q).
2010 [binary,1893.1,55.3,unit_del,668,706] -Col(p,q,c).
2114 [hyper,108,344,1998,unit_del,1841,2010] Col(p,q,b)|opposite(c,p,q,b).
2587 [binary,1338.2,57.2] -opposite(a,p,q,b)|Col(b,c,t).
2809 [binary,2114.1,55.1,unit_del,395,1049] opposite(c,p,q,b)|T(q,b,p).
3270 [hyper,32,344,2809,unit_del,1019] opposite(c,p,q,b)|q=b.
3306 [para_from,3270.2.1,694.1.2,unit_del,23] opposite(c,p,q,b).
3311 [hyper,114,3306,1838,1387] opposite(a,p,q,b).

3337 [binary,3311.1,2587.1] Col(b,c,t).
3348 [binary,3311.1,1292.1] sameside(p,q,d).
3353 [binary,3311.1,854.2] T(p,t,d).
3356 [binary,3311.1,798.1] -T(x,y,a)|T(d,ip(x,y,a,b,d),x).
3361 [binary,3311.1,761.1] -T(d,q,p).
3411 [binary,3337.1,109.2] -opposite(t,b,c,x).
3735 [para_into,3356.2.2,119.1.2,unit_del,279] T(d,t,p).
4114 [ur,30,3735,3361] -T(d,q,t).
5912 [binary,1611.1,106.2] -Col(c,b,c)|sameside(c,b,c).
6014 [binary,5912.1,63.2,unit_del,64] sameside(c,b,c).
6018 [binary,6014.1,104.1] Col(c,b,c).
6026 [binary,6018.1,62.1] Col(b,c,c).
6039 [hyper,108,279,6026,unit_del,1934,1489,1639] opposite(p,b,c,a).
6133 [hyper,114,6039,579,3348] opposite(d,b,c,a).
6141 [ur,114,6133,579,3411] -sameside(d,q,t).
6143 [binary,6141.1,106.3,unit_del,4114] -Col(d,q,t).
6154 [binary,6143.1,58.1] -T(t,d,q).
6155 [binary,6143.1,57.1] -T(q,t,d).
6280 [ur,27,3353,6154] -T(p,d,q).
6327 [ur,26,3353,6155] -T(q,p,d).
6350 [binary,6280.1,22.2] -T(q,d,p).
6392 [binary,6327.1,95.2,unit_del,3348,6350] $F.

I was not able to find a shorter proof, although I would bet a shorter proof exists.

I then turned to assumption reduction, whose goal is to reduce the number of items, taken from the input, that are used to still find a proof that inner Pasch implies outer. With fewer assumptions, axioms, and proved theorems, I would expect that the program would present to me, when a proof was found, one or more of greater length than 90. However, in contrast to proof shortening where I could run with a few strokes many jobs in sequence, for assumption reduction I had to run one job at a time. Each job would consist of my choosing one of the items used to obtain the inner-outer proof and comment it out. My method for making such choices was simply guessing. If a proof was obtained, whether a bit longer or so, then I would take the amended input file and choose another item to comment out. The plan was indeed simple; the exercising of it was totally up to me, in contrast to what might be termed the program telling me what to do in proof shortening by citing proof steps that, if demodulated, produced a shorter proof. You might try your hand at reducing the number of items used, thirty-seven, and experience a small amount of research and, perhaps, a large amount of excitement as you find that various axioms and proved theorems can indeed be omitted and still yield a proof. If that is your choice, perhaps you might pause here before seeing what happened in my sojourn.

With the knowledge that the 111-step proof relied on thirty-seven input axioms and proved theorems, I felt certain I could reduce that number. With this iterative approach, I eventually had a proof, from OTTER, in which but twenty-eight items were sufficient. Perhaps many guesses among the thirty-seven would have sufficed; I do not know. The following nine were omitted, and still a proof was found.

-T(xa,xb,xc) | -T(xb,xa,xc) | xa = xb.
-Col(x,y,z) | Col(y,z,x).
-Col(x,y,z) | Col(z,y,x).
-Col(x,y,z) | Col(x,z,y).
Col(x,x,y).
-T(xa,xb,xd) | -T(xa,xc,xd) | T(xa,xb,xc) | T(xa,xc,xb).
-sameside(xa,xp,xb) | T(xp,xa,xb) | T(xp,xb,xa).
xa = xp | sameside(xa,xp,xa).
Col(b,p,q) | -Col(b,p,q).

A glance at the 90-step proof shows that it relies on thirty-three axioms and previously proved theorems.

After many additional experiments, I learned that Beeson had a proof that inner implies outer that depends on the same twenty-eight.

Among those experiments I found a proof that depends on 28 but differs by the following six from the proof that agreed with Beeson's proof.

-T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xc,xd).
Col(xa,xb,xc)| -T(xa,xb,xc).
-Col(x,y,z)|Col(z,x,y).
xa=xp|xb=xp|xc=xp| -T(xa,xp,xc)|T(xb,xp,xc)| -sameside(xa,xp,xb).
Col(p,c,q)| -Col(p,q,c).
-Col(p,q,c)|Col(q,p,c).

The proof space is clearly quite rich, as you see from the fact that two different sets of twenty-eight axioms and proved theorems, differing by six items, still lead to a proof. The proof lengths that were offered me during this sojourn usually ranged from 101 to 108; however, one experiment led to a 246-step proof after quite a long time, a proof, if memory serves that depends on at least thirty-four items. Eventually, I found an input file that yielded a proof that in turn depended on but twenty-six axioms and previously proved theorems. Perhaps because of a type of redundancy, rather than being different from the thirty-seven by eleven, the file that yielded the proof relying on but twenty-six differed by avoiding the use of the following fourteen.

-T(xa,xb,xc) | -T(xb,xa,xc) | xa = xb.
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xb,xc).
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd).
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd).
Col(xa,xb,xc) | -T(xb,xc,xa).
-Col(x,y,z) | Col(y,z,x).
-Col(x,y,z) | Col(z,y,x).
-Col(x,y,z) | Col(y,x,z).
-Col(x,y,z) | Col(x,z,y).
Col(x,x,y).
-T(xa,xb,xd) | -T(xa,xc,xd) | T(xa,xb,xc) | T(xa,xc,xb).
-sameside(xa,xp,xb) | T(xp,xa,xb) | T(xp,xb,xa).
xa=xp | sameside(xa,xp,xa).
Col(b,p,q) | -Col(b,p,q).

To my surprise, I discovered along the way that in addition to dispensing with previously proved theorems, a proof can be found by omitting clauses that are part of various definitions. For example, if you avoid including in the input the first four of the following seven that were relied upon in a proof using twenty-eight items before making deductions, you can replace the four by the last three of the following seven to get a proof that relies on twenty-seven items.

-T(xa,xb,xc)| -T(xa,xc,xd)|T(xb,xc,xd).
-T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xb,xd).
Col(xa,xb,xc)| -T(xb,xc,xa).
-Col(x,y,z)|Col(y,x,z).
-T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xc,xd).
Col(xa,xb,xc)| -T(xa,xb,xc).
-Col(x,y,z)|Col(z,x,y).

I was also pleased to find that the clause for transitivity of betweenness can be dispensed with, and I enjoyed the discovery of what I think of as redundancy. For example, I have two proofs of inner implies outer, each relying on 26 items, which differ by the exchange of the first (of the following) for the second and conversely.

-sameside(xa,xp,xb)|xa!=xp.
-sameside(xa,xp,xb)| -T(xa,xp,xb).

I have not made a more detailed examination to see which, if any, of the axioms was unneeded. However, the obtaining of a long, long proof when some item is dispensed with does indeed suggest how important that axiom, part of a definition, or previously proved theorem is.

In the next section, with invaluable guidance and suggestions from Beeson, in part based on Szmielew and other sources (that include Coxeter's book titled *Introduction to Geometry*), I can turn to two proofs of great interest. During this part of the trip, you will see how tautologies can be used effectively, and, again, how a diagram proves crucial. I return, in the following section, to the 1959 Tarski axiom set, that given in Section 1, but using a notation consistent with Szmielew. In the first theorem, the goal is to prove connectivity of betweenness dependent on the other axioms of the set I gave first in this notebook. In the second theorem, the goal is to prove inner Pasch from that set, which, as you can see, relies on two clauses for outer Pasch.

20. Two Promised Theorems

In this section, you will learn, thanks to Beeson, how demodulation can be used directly to find a proof. Ordinarily, demodulation is used for canonicalization and simplification. Of course, you have been treated to its use in the context of proof shortening as well. I was surprised at his use of demodulation and its substantial value in proof finding, which is discussed when the theorem that proves inner Pasch from outer Pasch takes center stage.

Here you will see how the hot list usage made the difference, the difference between completing a proof and obtaining no proof at all. As you will learn, I turned to using the hot list strategy in place of using demodulation. Tautologies come into play in this section. Why their presence enables OTTER to find a proof, in contrast to their absence that leads to no proof, I cannot say; I have not analyzed the situation. You will also share my surprise when I found that level saturation carried the day, whereas its replacement with complexity-preference (ratio) did not. The approaches taken here, to prove that inner Pasch is implied by outer Pasch (with the use of the 1959 Tarski axioms given in Section 1), and that, in the presence of those axioms, the axiom of connectivity of betweenness is dependent—although not proved with OTTER, I regret to say—may indeed provide you with additional research ideas.

At one point in the research reported in this notebook, some doubt was cast regarding the dependence of connectivity of betweenness on the remaining axioms offered in Section 1. Nevertheless, based on nothing that occurs to me, I still held out hope that such dependence was a fact. Eventually, at my request, Beeson sent me an input file that provided me with a beginning for my attack on this promised theorem, although its use, as Beeson certainly knew, did not come close to proving what I wished to prove. Its hints2 list contained 655 items, including proof steps from a proof, perhaps found jointly by Beeson and me, of connectivity based on the use of inner Pasch; as noted, I was after a proof in which outer Pasch was used. Fortunately, as you will learn, I concentrated my efforts for quite a while on the theorem that deduces a proof of inner Pasch from outer Pasch in the presence of the 1959 Tarski axiom set. If I hadn't, at least for now in early May 2013, I would be stuck without a proof of the dependence of connectivity.

Beeson, in his usual kindness, sent me an input file to begin my study of proving inner Pasch from outer Pasch; and, if memory serves and moderate scrutiny seems to support, the file does include the axiom of connectivity of betweenness, an axiom that I later avoided using. Of course, he was certain that much was still needed. For but one aspect, the file was oriented toward a case analysis, focusing on a != c and, then, on a = c. Because I ordinarily accept input files from colleagues without reading them fully, I note, so important, that I was unaware that Beeson (in effect) suggested using demodulators to sharply aid OTTER in finding proofs. Clearly, as I learned so many, many experiments later, he had effectively employed demodulators based on an appropriate diagram to make substantial progress. But, being unaware of the role of demodulation, I turned to what I believe, if memory serves, was an assignment from Beeson.

He wished to find a proof that avoided the approach based on case analysis, in particular, based on the two cited cases, a != c and a = c. He had each of the two proofs, based on this case analysis, already. I conducted simultaneously two experiments, one based on the ratio strategy and one based on level saturation; in both, I commented out the cited cases and replaced them with the tautology a != c | a = c. The expectation is that level saturation will take a long time when compared with the use of ratio; after all, with level saturation, no retained clause is skipped because of its weight. Indeed, to initiate inference-rule applications, level saturation has the program consider the items in the order they are retained. By its nature, you thus see that the use of sos(queue), level saturation, has at the same time possible advantages and possible disadvantages.

What occurred was quite unexpected. Specifically, the level-saturation approach found the desired proof that outer Pasch implies inner Pasch in the presence of the 1959 axiom set. With ratio, the program eventually stopped, noting that the sos went empty, and with no proof found. Now how could that occur, you might ask. Well, subsumption may be the key.

In the following trivial example, you see that paramodulation does not behave as, say, binary resolution does in the context of generalizing a parent.

a = b.
Q(f(a)).
Q(F(b)).
Q(x).

With paramodulation, from the first two given items, you deduce the third. However, if the fourth item is substituted for the second and paramodulation is applied, the conclusion is *not* more general than that yielded from the first two items. Therefore, and I am simply conjecturing, OTTER may have deduced a more general item with the ratio strategy, subsumed a less general item, and been prevented from making a needed deduction.

The following 60-step proof was obtained.

A 60-Step Proof of Inner Pasch from Outer Pasch

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Fri Apr 19 14:00:14 2013
The command was "otter". The process ID is 10322.
----> UNIT CONFLICT at 835.70 sec ----> 19943 [binary,19942.1,19230.1] $F.

Length of proof is 60. Level of proof is 23.

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

2 [] T(x,y,ext(x,y,w,v)).
3 [] E(y,ext(x,y,w,v),w,v).
10 [] -E(xa,xb,xc,xd)|E(xc,xd,xa,xb).
11 [] -E(xa,xb,xc,xd)|E(xb,xa,xc,xd).
15 [] -E(u,v,x,x)|u=v.
16 [] T(x,y,y).
17 [] -T(xa,xb,xc)|T(xc,xb,xa).
18 [] T(xa,xa,xb).
21 [] -T(xa,xb,xd)| -T(xb,xc,xd)|T(xa,xc,xd).
22 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xb,xc,xd).
23 [] -T(xa,xb,xc)| -T(xa,xc,xd)|T(xa,xb,xd).
25 [] -T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xb,xd).
26 [] xa=xb| -T(xa,xb,xc)| -T(xa,xb,xd)|T(xa,xc,xd)|T(xa,xd,xc).
27 [] xa=xb| -T(xa,xb,xc)| -T(xa,xb,xd)|T(xb,xc,xd)|T(xb,xd,xc).
28 [] -T(xa,xb,xd)| -T(xa,xc,xd)|T(xa,xb,xc)|T(xa,xc,xb).
30 [] op(q,b,p,d,c)=j.
31 [] op(j,b,a,d,p)=e.
32 [] op(j,q,a,b,e)=f.
```

33 [] T(a,p,c).
34 [] T(b,q,c).
35 [] -T(p,x,b)| -T(q,x,a).
36 [] d=ext(a,c,a,c).
40 [] g=op(p,b,q,a,f).
41 [] -T(x,v,u)| -T(y,u,z)|T(x,op(v,x,y,z,u),y).
42 [] -T(x,v,u)| -T(y,u,z)|T(z,v,op(v,x,y,z,u)).
43 [] c!=d|c=d.
44 [] a!=c|a=c.
650 [] -T(xa,xb,xc)|T(xc,xb,xa).
654 [binary,33.1,17.1] T(c,p,a).
657 [hyper,23,33,2] T(a,p,ext(a,c,x,y)).
658 [hyper,22,33,2] T(p,c,ext(a,c,x,y)).
679 [binary,34.1,17.1] T(c,q,b).
704 [binary,35.1,18.1] -T(q,p,a).
705 [binary,35.2,18.1] -T(p,q,b).
706 [binary,35.2,16.1] -T(p,a,b).
708 (heat=1) [binary,706.1,650.2] -T(b,a,p).
709 [para_from,36.1.2,3.1.2] E(c,d,a,c).
710 [para_from,36.1.2,2.1.3] T(a,c,d).
750 [para_into,657.1.3,36.1.2] T(a,p,d).
752 [para_into,658.1.3,36.1.2] T(p,c,d).
753 (heat=1) [binary,752.1,650.1] T(d,c,p).
862 [ur,22,654,704] -T(c,q,p).
863 [ur,21,654,704] -T(q,c,a).
865 (heat=1) [binary,863.1,650.2] -T(a,c,q).
866 [ur,22,679,705] -T(c,p,q).
868 [binary,709.1,11.1] E(d,c,a,c).
1502 [hyper,42,34,752,demod,30] T(d,q,j).
1508 [hyper,41,34,752,demod,30] T(b,j,p).
1576 (heat=1) [binary,1502.1,650.1] T(j,q,d).
1582 (heat=1) [binary,1508.1,650.1] T(p,j,b).
1838 [para_into,863.1.2,44.2.2,unit_del,16] a!=c.
1846 [ur,28,654,862,866] -T(c,q,a).
1853 (heat=1) [binary,1846.1,650.2] -T(a,q,c).
1860 [binary,868.1,10.1] E(a,c,d,c).
2892 [hyper,42,1508,750,demod,31] T(d,j,e).
2901 [hyper,41,1508,750,demod,31] T(b,e,a).
2995 (heat=1) [binary,2901.1,650.1] T(a,e,b).
3076 [ur,21,1508,708] -T(j,a,p).
3568 [binary,1838.1,15.2] -E(a,c,x,x).
3750 [para_into,1860.1.3,43.2.2,unit_del,3568] c!=d.
5222 [hyper,22,1502,2892] T(q,j,e).
7188 [ur,25,710,3750,865] -T(c,d,q).
7215 (heat=1) [binary,7188.1,650.2] -T(q,d,c).
8420 [hyper,42,5222,2995,demod,32] T(b,j,f).
8427 [hyper,41,5222,2995,demod,32] T(q,f,a).
10326 [hyper,26,1508,8420] b=j|T(b,p,f)|T(b,f,p).
10669 [binary,8427.1,35.2] -T(p,f,b).
10670 (heat=1) [binary,10669.1,650.2] -T(b,f,p).
12824 [binary,10326.2,17.1,unit_del,10670] b=j|T(f,p,b).
13766 [para_from,12824.1.1,34.1.1] T(j,q,c)|T(f,p,b).
14362 [hyper,27,1576,13766,unit_del,7215] T(f,p,b)|j=q|T(q,c,d).

```
14624 [para_from,14362.2.1,1582.1.2,unit_del,705] T(f,p,b)|T(q,c,d).
14626 (heat=1) [binary,14624.2,650.1] T(f,p,b)|T(d,c,q).
14753 [hyper,27,753,14626,unit_del,866,862] T(f,p,b)|d=c.
14835 [para_from,14753.2.1,7215.1.2,unit_del,16] T(f,p,b).
14836 (heat=1) [binary,14835.1,650.1] T(b,p,f).
15472 [hyper,42,14836,8427] T(a,p,op(p,b,q,a,f)).
15480 [hyper,41,14836,8427] T(b,op(p,b,q,a,f),q).
16129 [para_into,15472.1.3,40.1.2] T(a,p,g).
16190 [hyper,22,15480,34] T(op(p,b,q,a,f),q,c).
16648 [hyper,26,33,16129] a=p|T(a,c,g)|T(a,g,c).
17230 [para_into,16190.1.1,40.1.2] T(g,q,c).
18265 [para_from,16648.1.1,3076.1.2,unit_del,16] T(a,c,g)|T(a,g,c).
19229 [ur,22,17230,863] -T(g,c,a).
19230 [ur,21,17230,1853] -T(a,g,c).
19231 (heat=1) [binary,19229.1,650.2] -T(a,c,g).
19941 [binary,18265.2,17.1,unit_del,19231] T(c,g,a).
19942 (heat=1) [binary,19941.1,650.1] T(a,g,c).
19943 [binary,19942.1,19230.1] $F.
```

I now give the initial segment of an input file that, with appropriate hints, will yield the just-given 60-step proof.

Initial Segment of an Input File for Proving Inner Pasch

```
% Tarski's 1959 axiom system.
% That consists of axioms A1-A6, outer pasch instead of A7, A8-A11,
% transitivity of betweenness (Satz 3.5, aka A15), and
% connectivity of betweeenness (Satz 5.1, aka A18).
% In this file we try to prove inner Pasch from outer Pasch.

set(hyper_res).
set(para_into).
set(para_from).
set(ur_res).
set(binary_res).
set(unit_deletion).
set(order_history).
assign(report,5400).
% assign(max_seconds, 60000).
assign(max_mem,840000).
clear(print_kept).
set(input_sos_first).
set(sos_queue).
set(back_sub).
assign(bsub_hint_wt,-1).
set(keep_hint_subsumers).
assign(max_weight,11).
assign(max_distinct_vars,4).
% assign(pick_given_ratio,2).
assign(max_proofs,4).
assign(neg_weight,8).

list(usable).
```

```
x=x.
T(x,y,ext(x,y,w,v)).                    % A4, first half
E(y,ext(x,y,w,v),w,v).                    % A4, second half
E(x,x,y,y).  % Satz 2.8
-T(x,y,x) | x=y.                    % A6
% following two clauses are outer Pasch.
-T(x,v,u) | -T(y,u,z) | T(x,op(v,x,y,z,u),y).
-T(x,v,u) | -T(y,u,z) | T(z,v,op(v,x,y,z,u)).
% following is the transitivity of betweenness, Satz 3.5, Axiom A15
-T(x,y,u) | -T(y,z,u) | T(x,y,z).
E(x,y,x,y).  % Satz 2.1
-E(xa,xb,xc,xd) | E(xc,xd,xa,xb).  % Satz 2.2
-E(xa,xb,xc,xd) | E(xb,xa,xc,xd).  % Satz 2.4
-E(xa,xb,xc,xd) | -E(xc,xd,xe,xf) | E(xa,xb,xe,xf).  % Satz 2.3
-E(xa,xb,xc,xd) | E(xa,xb,xd,xc).  % Satz 2.5
E(x,x,y,y).  % Satz 2.8
-E(u,v,x,x) | u=v.   % Not one of Szmielew's theorems but we proved it.
T(x,y,y).    % Satz 3.1
-T(xa,xb,xc) | T(xc,xb,xa).  % Satz 3.2.
T(xa,xa,xb).  % Satz 3.3
-T(xa,xb,xc) | -T(xb,xa,xc) | xa = xb.  % Satz 3.4.
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xb,xc).  % Satz 3.51.
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xc,xd).  % Satz 3.52.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd).  % Satz 3.61.
-T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd). % Satz 3.62.
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd).  % Satz 3.71
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd).  % Satz 3.72
% Following is Satz 5.1
xa = xb | -T(xa,xb,xc) | -T(xa,xb,xd) | T(xa,xc,xd) | T(xa,xd,xc).
% Following is Satz 5.2
xa = xb | -T(xa,xb,xc) | -T(xa,xb,xd)| T(xb,xc,xd) | T(xb,xd,xc).
% Following is Satz 5.3
-T(xa,xb,xd) | -T(xa,xc,xd) | T(xa,xb,xc) | T(xa,xc,xb).
end_of_list.
list(passive).
end_of_list.
list(demodulators).
ext(a,c,a,c)=d.
op(q,b,p,d,c)=j.
op(j,b,a,d,p)=e.
op(j,q,a,b,e)=f.
end_of_list.

list(sos).  % negated form of inner Pasch
T(a,p,c).
T(b,q,c).
-T(p,x,b) | -T(q,x,a).
% Now we define the diagram for the proof
d = ext(a,c,a,c).
j = op(q,b,p,d,c).
e = op(j,b,a,d,p).
f = op(j,q,a,b,e).
g = op(p,b,q,a,f).
```

```
% following two clauses are outer Pasch.
-T(x,v,u) | -T(y,u,z) | T(x,op(v,x,y,z,u),y).
-T(x,v,u) | -T(y,u,z) | T(z,v,op(v,x,y,z,u)).
c!=d | c = d.  % works with this tautology present, but not without!
a != c | a = c.
%  b=j | b != j.  % works with either case, but not with the disjunction.
end_of_list.
```

I have omitted the large set of hints that were used because the key aspect at this point is the list of axioms and theorems that were used. Some of the theorems that are listed and that were used to obtain the given 60-step proof were not proved by me. I leave that for you as a challenge. As you will shortly see, I turned to experiments after finding the 60-step proof, designed to see how many of the items (from the input) that were used in the proof that I could do without, experiments reminiscent of some discussed earlier in this notebook. As for crucial differences between the given initial segment and its correspondent in the input file Beeson sent to me, I included the use of unit_deletion and the use of level saturation with set(sos_queue).

A few lines earlier, I noted that I was fortunate to begin with an emphasis on proving inner Pasch from outer. Indeed, that choice naturally led me (as just noted) to a study similar to a study reported earlier in this notebook, namely, the experimentation with removing various items chosen from the input and used in a proof. You might think of this activity as a cousin to proof shortening. Among the experiments that I conducted, one of them proved, much, much later, to be fortuitous and indeed extremely relevant to the study of the possible dependence of connectivity. However, history and science each and both require that I report that the relevance of an early experiment to the connectivity theorem did not in any way occur to me until I was seeking a so-called direct proof—and finding the venture difficult and hazardous. As I describe my blocking of one after another of input items, you, in contrast to me, may immediately see the relevance to the dependence of connectivity.

From among the possible choices to block, to avoid using, from those theorems in the input file, I chose Satz 5.3, the following.

```
%  -T(xa,xb,xd) | -T(xa,xc,xd) | T(xa,xb,xc) | T(xa,xc,xb).  %  Satz 5.3
```

(By the way, for the person who enjoys following the geometry, 5.3 asserts the totally believable, namely, if points b and c are each between points a and d, then either b is between a and c or c is between a and b.) You could naturally guess that such a statement is not needed for proving this important and powerful theorem, that outer Pasch implies inner Pasch. In approximately the same CPU time, around 2800 CPU-seconds, a second 60-step proof was completed. When compared with the 60-step proof you viewed a bit ago, both rely on twenty-eight items from the input file, which I did not expect, believing that the avoidance of 5.3 would lead to the use of but twenty-seven. In place of relying on 5.3, this second 60-step proof relied on 3.51, the following.

```
-T(xa,xb,xd)| -T(xb,xc,xd)|T(xa,xb,xc).  %  Satz 3.51
```

Again, for the person who enjoys pictures, you can draw a line a,b,d, then insert the point c between b and d, and see how reasonable is the fact that b is between a and c. The following two deductions were present in this second 60-step proof but not in the given 60-step proof.

```
-T(a,p,q).
-T(a,g,q).
```

I include various comparisons and statistics because they might lead a researcher to a new strategy or a new approach.

Have you been eager to receive another challenge, especially a challenge I cannot as yet meet? Well, I have such for you now. In particular, I took the amended input file, amended by commenting out 5.3, and made a third input file, a file in which I commented out Satz 5.2, the following.

```
%  xa = xb | -T(xa,xb,xc) | -T(xa,xb,xd)| T(xb,xc,xd) | T(xb,xd,xc)  %  Satz 5.2
```

An appealing claim this is, namely, if a is not equal to b, and if the point b is between a and c and also between a and d, then either c is between b and d or d is between b and c. With this second blocking, and various experiments—including some conducted weeks after my first one—I have failed to find a proof in which 5.2 is absent. If you find such a proof, an e-mail to me would indeed be of interest. This challenge seems formidable and, just perhaps, cannot be met. (Yes, I did not, as I often do, switch to a new set of hints or resonators; if I had, the new set would be based on the preceding proof or proofs, which might cause OTTER to miss the desired proof that avoids the use of 5.2.)

So, to cope with this annoying failure and, more important, to continue on my quest for finding a proof of inner from outer in which fewer, perhaps far fewer, than twenty-eight items from among axioms, theorems, and definitions were needed, I reinstated the possible use of 5.2 and, instead, commented out 5.1, the following.

```
% xa = xb | -T(xa,xb,xc) | -T(xa,xb,xd) | T(xa,xc,xd) | T(xa,xd,xc)  %  Satz 5.1
```

To me, 5.1 and 5.2 appear to be similar. Do you find them quite alike? More intriguing, especially if you could conjecture what will be presented late in this notebook, does Satz 5.1 remind you of an important property? At this juncture in my journey toward the city of inner Pasch, I was not reminded of any property. Perhaps I was too engrossed in my quest for a proof relying on fewer, maybe far fewer, than twenty-eight items.

OTTER did find a proof, one of length sixty-three, relying on twenty-seven input items—progress—omitting the use of 5.3 and 5.1, and adding the use of 3.51. The 63-step proof contains seventeen deduced items not in the first 60-step proof, and 15 not in the second. I was not surprised at finding a somewhat longer proof, although commenting out items does not guarantee the finding of longer proofs. (To get some idea of my thinking, remember that at this point I still did not understand how valuable this 63-step proof was and would prove to be; I simply continued my assault on assumption reduction.)

The story continued to unfold, one experiment after another, as I blithely experienced satisfaction, indeed unaware of the significance of having found a proof in which Satz 5.1 was not relied upon. The thought of eventually proving connectivity of betweenness dependent essentially drifted into the background. For example, when I commented out (blocked the use of) the following two, 3.62 and 3.72, OTTER produced a pleasing result.

```
% -T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd). % Satz 3.62.
% -T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd).  % Satz 3.72
```

A 65-step proof was found, relying on twenty-five, compared with twenty-eight, items from the input file. Not used were 5.3, 5.1, 3.62, and 3.72, the latter two of which I had proved many weeks ago (from the 1959 axiom system). But, rather than introducing the use of 3.51 as was the case earlier, 3.71, the following, was relied upon.

```
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd).  % Satz 3.71
```

I have not attempted to analyze the nature of these various tradeoffs, but they are, to me, entertaining. The 65-step proof that was presented to me contained eight deduced items not in the 63-step proof and contained twenty-two not present in the first (given) 60-step proof.

As my experiments proceeded, I was making real progress, still getting proofs of inner from outer but with fewer and fewer items from the input. It occurred to me, however, that the use of demodulation interferes with an accurate determination of proof length. Therefore, the proof lengths I had been quoting were a bit misleading. For example, an examination of the earlier-given 60-step proof shows that demodulation was used six times, the demodulators corresponding to the diagram that Beeson supplied to permit OTTER to find that proof. If you count the occurrences of demodulation, noting that at no time in the 60-step proof did a step rely on more than one demodulator, you would cite the proof as being one of length 66. Because I prefer that the program report to me the more accurate proof length, and because I was curious about what would occur if I did not rely on demodulation, I eventually turned to the use of the hot list strategy. And here is a move, or set of moves, that you might find not only interesting but profitable to make in various contexts in diverse studies. Specifically, in place of the demodulator list you find in the Segment of an input file given earlier, I made two changes. I inserted a command, the following, and I included a hot list with the items that provide the diagram.

```
assign(heat,1).
list(hot).
ext(a,c,a,c)=d.
op(q,b,p,d,c)=j.
op(j,b,a,d,p)=e.
op(j,q,a,b,e)=f.
end_of_list.
```

I, of course, commented out the demodulator list and its contents, rather than removing those lines. I also copied the elements in the given hot list to list(usable). They could have been copied to list(sos). With the cited moves, if and when a proof is found, the quoted proof length will reflect accurately the number of deduced items found in the corresponding proof. If memory serves, in almost all the remaining experiments reported here that pertain to assumption-shortening, I relied on the hot list strategy rather than on demodulation.

Instead of detailing the remainder of this journey, I will simply focus on the best of my results, a proof that relies on but nineteen, rather than twenty-eight, items from the input file. As you examine the proof I am about to display, I note that when the hot list strategy is in use, an input item can be cited more than once.

A Proof of Inner Pasch from Outer with a Minimal Input Set

```
----- Otter 3.3g-work, Jan 2005 -----
The process was started by wos on vanquish,
Mon Apr 29 16:28:37 2013
The command was "otter".  The process ID is 7855.
----> UNIT CONFLICT at  36.53 sec ----> 57028 [binary,57027.1,1488.1] $F.

Length of proof is 105.  Level of proof is 27.

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

1 [] T(x,y,ext(x,y,w,v)).
2 [] E(y,ext(x,y,w,v),w,v).
4 [] -T(x,y,u)| -T(y,z,u)|T(x,y,z).
5 [] -E(xa,xb,xc,xd)|E(xc,xd,xa,xb).
7 [] -E(u,v,x,x)|u=v.
8 [] -T(xa,xb,xc)|T(xc,xb,xa).
10 [] -T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xc,xd).
11 [] xa=xb| -T(xa,xb,xc)| -T(xa,xb,xd)|T(xb,xc,xd)|T(xb,xd,xc).
13 [] T(a,p,c).
14 [] T(b,q,c).
15 [] -T(p,x,b)| -T(q,x,a).
16 [] g=op(p,b,q,a,f).
17 [] -T(x,v,u)| -T(y,u,z)|T(x,op(v,x,y,z,u),y).
18 [] -T(x,v,u)| -T(y,u,z)|T(z,v,op(v,x,y,z,u)).
19 [] c!=d|c=d.
23 [] op(j,q,a,b,e)=f.
887 [] ext(a,c,a,c)=d.
888 [] op(q,b,p,d,c)=j.
889 [] op(j,b,a,d,p)=e.
890 [] op(j,q,a,b,e)=f.
892 [binary,5.1,2.1] E(x,y,z,ext(u,z,x,y)).
```

893 (heat=1) [para_into,892.1.4,887.1.1] E(a,c,c,d).
895 [binary,7.1,2.1] x=ext(y,x,z,z).
896 [binary,8.1,4.3] T(x,y,z)| -T(z,y,u)| -T(y,x,u).
897 [binary,8.1,1.1] T(ext(x,y,z,u),y,x).
898 [binary,8.2,4.2] -T(x,y,z)| -T(u,z,x)|T(u,z,y).
900 (heat=1) [para_into,897.1.1,887.1.1] T(d,c,a).
902 [binary,10.2,8.2] -T(x,y,z)|y=z|T(x,z,u)| -T(u,z,y).
911 [binary,13.1,8.1] T(c,p,a).
933 [binary,14.1,8.1] T(c,q,b).
948 [binary,15.1,10.4] -T(q,x,a)| -T(p,y,x)| -T(y,x,b)|y=x.
951 [binary,15.2,10.4] -T(p,x,b)| -T(q,y,x)| -T(y,x,a)|y=x.
961 [binary,17.1,14.1] -T(x,c,y)|T(b,op(q,b,x,y,c),x).
1064 [para_from,23.1.2,16.1.2.5] g=op(p,b,q,a,op(j,q,a,b,e)).
1082 [para_into,893.1.3,19.2.1] E(a,c,d,d)|c!=d.
1095 [binary,896.2,8.2] T(x,y,z)| -T(y,x,u)| -T(u,y,z).
1098 [binary,896.3,8.2] T(x,y,z)| -T(z,y,u)| -T(u,x,y).
1142 [binary,898.2,8.2] -T(x,y,z)|T(u,z,y)| -T(x,z,u).
1146 [hyper,898,13,897] T(ext(a,c,x,y),c,p).
1147 (heat=1) [para_into,1146.1.1,887.1.1] T(d,c,p).
1236 [binary,902.3,8.1] -T(x,y,z)|y=z| -T(u,z,y)|T(u,z,x).
1292 [hyper,896,900,911] T(p,c,d).
1360 [binary,961.1,8.2] T(b,op(q,b,x,y,c),x)| -T(y,c,x).
1361 [binary,961.2,8.1] -T(x,c,y)|T(x,op(q,b,x,y,c),b).
1362 (heat=1) [para_into,1360.1.2,888.1.1,unit_del,1147] T(b,j,p).
1478 [binary,1095.2,911.1] T(p,c,x)| -T(a,c,x).
1486 [para_into,895.1.2,895.1.2] x=x.
1488 [para_from,895.1.2,897.1.1] T(x,x,y).
1489 [para_from,895.1.2,1.1.3] T(x,y,y).
1493 [binary,1098.2,933.1] T(x,q,c)| -T(b,x,q).
1514 [binary,1147.1,1098.2] T(x,c,d)| -T(p,x,c).
1528 [binary,1147.1,11.3] d=c| -T(d,c,x)|T(c,x,p)|T(c,p,x).
1576 [hyper,902,1147,13] c=p|T(d,p,a).
1748 [binary,1292.1,18.2] -T(x,y,c)|T(d,y,op(y,x,p,d,c)).
1855 [binary,1361.2,15.1] -T(p,c,x)| -T(q,op(q,b,p,x,c),a).
1860 (heat=1) [para_into,1855.2.2,888.1.1,unit_del,1292] -T(q,j,a).
2143 [binary,1489.1,1098.2] T(x,y,z)| -T(y,x,y).
2579 [para_from,1576.1.1,900.1.2,factor_simp] T(d,p,a).
2589 (heat=1) [para_into,2579.1.1,887.1.2] T(ext(a,c,a,c),p,a).
2706 [binary,1748.2,8.1] -T(x,y,c)|T(op(y,x,p,d,c),y,d).
2718 (heat=1) [para_into,2706.2.1,888.1.1,unit_del,14] T(j,q,d).
3042 [binary,1860.1,8.2] -T(a,j,q).
3354 [binary,2579.1,8.1] T(a,p,d).
3496 [binary,2589.1,951.3,unit_del,1488] -T(q,ext(a,c,a,c),p)|ext(a,c,a,c)=p.
3950 [binary,2718.1,11.3] j=q| -T(j,q,x)|T(q,x,d)|T(q,d,x).
4183 [binary,3354.1,18.2] -T(x,y,p)|T(d,y,op(y,x,a,d,p)).
4185 [binary,3354.1,17.2] -T(x,y,p)|T(x,op(y,x,a,d,p),a).
4207 (heat=1) [para_into,4183.2.3,889.1.1,unit_del,1362] T(d,j,e).
4502 [para_from,3950.1.1,3042.1.2,unit_del,1489] -T(j,q,x)|T(q,x,d)|T(q,d,x).
4733 [binary,4185.2,8.1] -T(x,y,p)|T(a,op(y,x,a,d,p),x).
4746 (heat=1) [para_into,4733.2.2,889.1.1,unit_del,1362] T(a,e,b).
4944 [hyper,1095,2718,4207] T(q,j,e).
5441 [binary,4746.1,18.2] -T(x,y,e)|T(b,y,op(y,x,a,b,e)).
5443 [binary,4746.1,17.2] -T(x,y,e)|T(x,op(y,x,a,b,e),a).

5469 (heat=1) [para_into,5441.2.3,890.1.1,unit_del,4944] T(b,j,f).
5474 (heat=1) [para_into,5443.2.2,890.1.1,unit_del,4944] T(q,f,a).
5977 [hyper,11,1362,5469] b=j|T(j,p,f)|T(j,f,p).
6094 [binary,5474.1,15.2] -T(p,f,b).
6100 [binary,5474.1,8.1] T(a,f,q).
6469 [binary,5977.2,8.1] b=j|T(j,f,p)|T(f,p,j).
7017 [binary,6469.2,8.1] b=j|T(f,p,j)|T(p,f,j).
7538 [binary,1488.1,948.1] -T(p,x,q)| -T(x,q,b)|x=q.
7545 [binary,1488.1,15.2] -T(p,q,b).
7546 [binary,1488.1,15.1] -T(q,p,a).
7820 [binary,7538.2,933.1] -T(p,c,q)|c=q.
7895 [ur,1098,7545,14] -T(c,p,q).
7940 [binary,7546.1,8.2] -T(a,p,q).
7944 [ur,1098,7546,3354] -T(d,q,p).
7945 [ur,1098,7546,13] -T(c,q,p).
7948 [ur,1095,7546,2579] -T(p,q,d).
8180 [para_from,7820.2.1,1292.1.2,unit_del,7948] -T(p,c,q).
8206 [binary,7895.1,2143.1] -T(p,c,p).
8207 [binary,7895.1,1528.4,unit_del,7945] d=c| -T(d,c,q).
8338 [ur,1142,7944,2718] -T(j,p,q).
8552 [binary,8180.1,1478.1] -T(a,c,q).
8565 [binary,8180.1,8.2] -T(q,c,p).
8673 [para_into,8206.1.1,3496.2.2,unit_del,1146] -T(q,ext(a,c,a,c),p).
8675 (heat=1) [para_into,8673.1.2,887.1.1] -T(q,d,p).
8902 [binary,8338.1,2143.1] -T(p,j,p).
9166 [ur,898,14,8552] -T(a,c,b).
9417 [ur,1095,8902,1362] -T(j,p,b).
26823 [binary,1082.1,7.1] c!=d|a=c.
30755 [para_from,26823.2.1,9166.1.1,unit_del,1488] c!=d.
30826 [ur,1236,1292,30755,8675] -T(q,d,c).
30906 [para_into,30755.1.1,8207.1.2,unit_del,1486] -T(d,c,q).
31185 [binary,30906.1,8.2] -T(q,c,d).
31298 [binary,31185.1,4502.2,unit_del,30826] -T(j,q,c).
49340 [para_into,31298.1.1,7017.1.2,unit_del,14] T(f,p,j)|T(p,f,j).
49966 [hyper,1236,5469,49340,unit_del,6094] T(f,p,j)|j=f.
52685 [para_from,49966.2.1,3042.1.2,unit_del,6100] T(f,p,j).
53043 [hyper,1236,1362,52685] j=p|T(f,p,b).
53809 [para_from,53043.1.1,9417.1.1,unit_del,1488] T(f,p,b).
54690 [binary,53809.1,8.1] T(b,p,f).
55261 [hyper,18,54690,5474] T(a,p,op(p,b,q,a,f)).
55271 [hyper,17,54690,5474] T(b,op(p,b,q,a,f),q).
55420 [para_into,55261.1.3,16.1.2] T(a,p,g).
55421 [binary,55271.1,1493.2] T(op(p,b,q,a,f),q,c).
55585 [hyper,11,13,55420] a=p|T(p,c,g)|T(p,g,c).
55818 [ur,1098,8565,55421] -T(p,c,op(p,b,q,a,f)).
55844 (heat=1) [para_into,55818.1.3.5,890.1.2] -T(p,c,op(p,b,q,a,op(j,q,a,b,e))).
55855 [para_into,55421.1.1,16.1.2] T(g,q,c).
56083 [para_into,55585.2.3,1064.1.1,unit_del,55844] a=p|T(p,g,c).
56391 [ur,1142,55855,30906] -T(g,c,d).
56691 [binary,56083.2,1514.2,unit_del,56391] a=p.
57027 [para_from,56691.1.1,7940.1.1] -T(p,p,q).
57028 [binary,57027.1,1488.1] $F.

You see, in this given proof, items with heat cited, meaning that the hot list strategy played a role in the corresponding deduction. You also see that one input item, the following,is present twice, once because of being in list(sos) and once from being in list(hot).

op(j,q,a,b,e)=f.

The following three items from the input were now used but not used in the first 60-step proof, the third actually being used but with its arguments interchanged.

```
-T(x,y,u)| -T(y,z,u)|T(x,y,z).  $  Transitivity of betweenness
-T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xc,xd).  %  Satz 3.71
ext(a,c,a,c)=d.  %  from the Beeson diagram
```

I like the resulting 105-step proof in part because none of the theorems relied upon, from the input, were unproved by me from the 1959 axiom system. Finally, the following eleven items, taken from the Input Fragment, were used in the given 60-step proof but are not used in a 145-step proof.

```
-E(xa,xb,xc,xd)|E(xb,xa,xc,xd).
T(x,y,y).
T(xa,xa,xb).
-T(xa,xb,xd)| -T(xb,xc,xd)|T(xa,xc,xd).
-T(xa,xb,xc)| -T(xa,xc,xd)|T(xb,xc,xd).
-T(xa,xb,xc)| -T(xa,xc,xd)|T(xa,xb,xd).
-T(xa,xb,xc)| -T(xb,xc,xd)|xb=xc|T(xa,xb,xd).
xa=xb| -T(xa,xb,xc)| -T(xa,xb,xd)|T(xa,xc,xd)|T(xa,xd,xc).
-T(xa,xb,xd)| -T(xa,xc,xd)|T(xa,xb,xc)|T(xa,xc,xb).
d=ext(a,c,a,c).
a!=c|a=c.
```

I was able to dispense with the tautology a != c | a = c. I was not able to dispense with the tautology c != d | c = d.

If you might enjoy probing further, trying to eliminate more items from the input, searching for a shorter proof, or seeking to satisfy some other aspect, the following input file might serve well; its use will give a 105-step proof of inner from outer, and with connectivity of betweenness commented out.

A Minimized Input File for Proving Inner Pasch from Outer

```
%  Tarski's 1959 axiom system.
%  That consists of axioms A1-A6, outer pasch instead of A7, A8-A11, transitivity of betweenness
%(Satz 3.5, aka A15), and connectivity of betweeenness (Satz 5.1, aka A18).
%  In this file we try to prove inner Pasch from outer Pasch.
% This file works, except for some case distinctions we can't yet remove.  See list(sos).


set(hyper_res).
set(para_into).
set(para_from).
set(ur_res).
set(binary_res).
set(unit_deletion).
set(order_history).
assign(report,5400).
%  assign(max_seconds, 60000).
assign(max_mem,840000).

clear(print_kept).
% set(very_verbose).
```

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


assign(max_weight,15).
assign(max_distinct_vars,4).
assign(pick_given_ratio,4).
assign(max_proofs,4).
%  assign(neg_weight,8).
assign(heat,1).

list(sos).
%  x=x.
T(x,y,ext(x,y,w,v)).                          % A4, first half
E(y,ext(x,y,w,v),w,v).                         % A4, second half
%  E(x,x,y,y).  % Satz 2.8
%x = y | ext(u,v,x,y) != v.
%  -T(x,y,x) | x=y.                           % A6
%  -T(xa,xp,xc) | -T(xb,xq,xc) | T(xp,ip(xa,xp,xc,xb,xq),xb).
%A7, first part (inner Pasch)
%  -T(xa,xp,xc) | -T(xb,xq,xc) | T(xq,ip(xa,xp,xc,xb,xq),xa).
%A7, second part (inner Pasch)
% following two clauses are outer Pasch.
%  -T(x,v,u) | -T(y,u,z) | T(x,op(v,x,y,z,u),y).
-T(x,v,u) | -T(y,u,z) | T(z,v,op(v,x,y,z,u)).
% following is the transitivity of betweenness, Satz 3.5, Axiom A15
-T(x,y,u) | -T(y,z,u) | T(x,y,z).

%  E(x,y,x,y).  % Satz 2.1
-E(xa,xb,xc,xd) | E(xc,xd,xa,xb).  % Satz 2.2
%  -E(xa,xb,xc,xd) | E(xb,xa,xc,xd).  % Satz 2.4
-E(xa,xb,xc,xd) | -E(xc,xd,xe,xf) | E(xa,xb,xe,xf).  % Satz 2.3
%  -E(xa,xb,xc,xd) | E(xa,xb,xd,xc).  % Satz 2.5
%  E(x,x,y,y).  % Satz 2.8
-E(u,v,x,x) | u=v.   % Not one of Szmielew's theorems but we proved it.


%  T(x,y,y).    % Satz 3.1
-T(xa,xb,xc) | T(xc,xb,xa).  % Satz 3.2.
%  T(xa,xa,xb).  % Satz 3.3
%  -T(xa,xb,xc) | -T(xb,xa,xc) | xa = xb.  % Satz 3.4.
-T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xb,xc). % Satz 3.51.
%  -T(xa,xb,xd) | -T(xb,xc,xd) | T(xa,xc,xd). % Satz 3.52.
%  -T(xa,xb,xc) | -T(xa,xc,xd) | T(xb,xc,xd). % Satz 3.61.
%  -T(xa,xb,xc) | -T(xa,xc,xd) | T(xa,xb,xd). % Satz 3.62.
-T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xc,xd).  % Satz 3.71
%  -T(xa,xb,xc) | -T(xb,xc,xd) | xb = xc | T(xa,xb,xd).  % Satz 3.72

%  % Following is Satz 5.1
%  xa = xb | -T(xa,xb,xc) | -T(xa,xb,xd) | T(xa,xc,xd) | T(xa,xd,xc).
```

```
% Following is Satz 5.2
xa = xb | -T(xa,xb,xc) | -T(xa,xb,xd)| T(xb,xc,xd) | T(xb,xd,xc).
%  % Following is Satz 5.3
%  -T(xa,xb,xd) | -T(xa,xc,xd) | T(xa,xb,xc) | T(xa,xc,xb).



end_of_list.

list(passive).
%-T(p,c,d) | $ANS(0).
%-T(d,c,p) | $ANS(1).
%-T(a,c,d) | $ANS(2).
%-T(p,c,d) | $ANS(3).
%-T(d,c,p) | $ANS(4).
%-T(c,q,b) | $ANS(5).
%-T(d,q,j) | $ANS(6).
%-T(j,q,d) | $ANS(7).
%-T(p,j,b) | $ANS(8).
%-T(b,j,p) | $ANS(9).
-T(a,c,d) | $ANS(10).
%-T(d,j,e) | $ANS(11).
%-T(a,e,b) | $ANS(12).
%-T(q,j,e) | $ANS(13).
%-T(a,f,q) | $ANS(14).
%-T(q,f,a) | $ANS(15).
%-T(f,j,b) | $ANS(16).
%-T(f,p,b) | $ANS(17).
%-T(q,g,b) | $ANS(18).
%-T(a,p,d) | $ANS(19).
%-T(a,p,g) | $ANS(20).
%c !=op(p,b,q,a,f) | $ANS(21).
%c != g | $ANS(22).

%-T(b,q,d) | $ANS(23).
%-T(b,q,c) | $ANS(24).
%-T(d,c,q) | $ANS(25).
%-T(d,c,b) | $ANS(26).
%-T(d,c,p) | $ANS(27).
%-T(p,b,b) | $ANS(28).
%-T(d,c,a) | $ANS(29).
%-T(a,b,q) | $ANS(30).
%-T(b,j,f) | $ANS(31).
%-T(f,j,b) | $ANS(32).
%-T(b,j,p) | $ANS(33).
%-T(p,j,b) | $ANS(34).
%-T(b,q,d) | $ANS(35).
%-T(b,q,c) | $ANS(36).

%-T(j,q,d | $ANS(37)).
%-T(b,q,d) | $ANS(38).
%-T(b,q,c) | $ANS(39).
%b = q | $ANS(40).
```

```
%-T(q,c,d) | $ANS(41).
%-T(p,c,d) | $ANS(42).
%-T(a,c,q) | $ANS(43).
%-T(p,c,b) | $ANS(44).
%-T(p,c,q) | $ANS(45).
%-T(p,c,d)| $ANS(46).
%-T(d,c,a) | $ANS(47).
%-T(d,c,q) | $ANS(48).
%d=c | $ANS(49).

%-E(a,a,c,d) | $ANS(50).
%-E(a,c,c,d) | $ANS(51).
%-E(c,d,a,a) | $ANS(52).
%-E(c,d,a,c) | $ANS(53).
%-E(a,a,a,c) | $ANS(54).
%a != c | $ANS(55).

end_of_list.

%  list(demodulators).
%  ext(a,c,a,c)=d.
%  op(q,b,p,d,c)=j.
%  op(j,b,a,d,p)=e.
%  op(j,q,a,b,e)=f.
%  end_of_list.

list(sos).  % negated form of inner Pasch
T(a,p,c).
T(b,q,c).
-T(p,x,b) | -T(q,x,a).
% Now we define the diagram for the proof
%  d = ext(a,c,a,c).
%  j = op(q,b,p,d,c).
%  e = op(j,b,a,d,p).
%  f = op(j,q,a,b,e).
g = op(p,b,q,a,f).
% following two clauses are outer Pasch.
-T(x,v,u) | -T(y,u,z) | T(x,op(v,x,y,z,u),y).
-T(x,v,u) | -T(y,u,z) | T(z,v,op(v,x,y,z,u)).
c!=d | c = d.   % works with this tautology present, but not without!

% T(p,f,j) | T(f,p,j).  % This works.

% a=c. %gives a proof, so we can assume

%  a != c.
%  a != c | a = c.
%  b = j | b != j.   % works with either case, but not with the disjunction.

%q = b.   % That works too, but that's irrelevant.


op(j,b,a,d,p)=e.
```

```
op(j,q,a,b,e)=f.
end_of_list.

list(hints2).
%  following 133 prove inner from outer, temp.beeson.outer.inner.new.out8d6q2,
%  relying on many fewer input items than 8a, for ex
E(x,y,z,u)| -E(z,u,y,x).
E(x,ext(y,x,z,u),u,z).
E(c,d,c,a).
-T(x,y,x)|T(z,x,y).
-T(x,y,x)|T(y,x,z).
T(x,y,z)| -T(z,y,u)| -T(y,x,u).
T(ext(x,y,z,u),y,x).
-T(x,y,z)| -T(u,z,x)|T(u,z,y).
T(x,y,z)| -T(y,x,u)| -T(u,y,z).
T(x,y,z)| -T(z,y,u)| -T(u,x,y).
T(d,c,a).
-T(x,y,z)|T(u,z,y)| -T(x,z,u).
-T(x,y,z)|y=z|T(x,z,u)| -T(u,z,y).
-T(x,y,z)|y=z| -T(u,z,y)|T(u,z,x).
T(c,p,a).
T(c,q,b).
-T(x,c,y)|T(b,op(q,b,x,y,c),x).
-T(x,c,y)|T(x,op(q,b,x,y,c),b).
-T(q,x,a)| -T(p,y,x)| -T(y,x,b)|y=x.
-T(q,p,a).
-T(p,x,b)| -T(q,y,x)| -T(y,x,a)|y=x.
-T(p,q,b).
-T(p,a,b).
-T(a,p,q).
-T(b,q,p).
-T(b,a,p).
g=op(p,b,q,a,op(j,q,a,b,e)).
T(d,c,p).
d=c| -T(d,c,x)|T(c,x,p)|T(c,p,x).
T(p,c,d).
-T(p,x,q)| -T(x,q,b)|x=q.
-T(p,q,c).
-T(b,x,a)| -T(x,a,p)|x=a.
-T(a,p,a).
c=p|T(d,p,a).
-T(x,y,c)|T(d,y,op(y,x,p,d,c)).
-T(x,y,c)|T(x,op(y,x,p,d,c),p).
T(d,q,j).
T(b,j,p).
T(b,op(q,b,p,d,c),p).
T(p,op(q,b,p,d,c),b).
a!=c.
T(j,q,d).
j=q| -T(j,q,x)|T(q,x,d)|T(q,d,x).
T(p,j,b).
-E(a,c,x,x).
-T(q,j,a).
```

-T(a,j,q).
-E(x,x,c,a).
-T(q,p,c).
-T(c,p,q).
-T(p,q,x)| -T(q,c,x).
-T(q,c,x)| -T(x,q,p).
-T(q,c,q).
-T(op(q,b,p,d,c),x,p)|T(b,op(q,b,p,d,c),x).
-T(op(q,b,p,d,c),x,b)|T(p,op(q,b,p,d,c),x).
c!=d.
c!=ext(a,c,a,c).
-T(p,c,p).
-T(q,c,b).
-T(b,c,q).
-T(c,p,d).
-T(d,p,c).
T(p,c,ext(a,c,x,y)).
T(x,c,d)| -T(p,x,c).
T(x,q,c)| -T(b,x,q).
T(d,p,a).
T(a,p,d).
T(ext(a,c,a,c),p,a).
-T(d,q,p).
-T(x,y,p)|T(d,y,op(y,x,a,d,p)).
-T(x,y,p)|T(x,op(y,x,a,d,p),a).
T(d,j,e).
T(b,e,a).
T(q,j,e).
T(a,e,b).
-T(x,e,y)|T(y,j,op(j,q,x,y,e)).
-T(x,e,y)|T(q,op(j,q,x,y,e),x).
T(b,j,f).
T(q,f,a).
T(f,j,b).
b=j|T(j,p,f)|T(j,f,p).
b=j|T(j,f,p)|T(f,p,j).
-T(p,f,b).
T(a,f,q).
-T(j,f,b)|j=f.
b=j|T(f,p,j)|T(p,f,j).
-T(j,p,q).
-T(p,j,p).
-T(j,f,b).
-T(f,j,f).
-T(j,f,j).
-T(j,f,op(q,b,p,d,c)).
-T(p,c,q)|c=q.
-T(q,ext(a,c,a,c),p)|ext(a,c,a,c)=p.
-T(p,c,q).
-T(q,c,p).
-T(q,ext(a,c,a,c),p).
-T(q,ext(a,c,a,c),c).
-T(q,d,c).

d=c| -T(d,c,q).
-T(d,c,q).
-T(q,c,d).
-T(op(q,b,p,d,c),q,b)|op(q,b,p,d,c)=q.
-T(op(q,b,p,d,c),q,b).
-T(j,q,b).
-T(c,j,q).
-T(op(q,b,p,d,c),a,p)|op(q,b,p,d,c)=a.
-T(op(q,b,p,d,c),a,p).
-T(j,a,p).
-T(p,a,j).
j=q| -T(b,j,q).
-T(b,j,q).
T(f,p,j)|T(p,f,j).
T(f,p,j)|j=f.
T(f,p,j).
j=p|T(f,p,b).
-T(p,j,f).
-T(p,j,op(j,q,a,b,e)).
T(f,p,b).
T(b,p,f).
T(a,p,op(p,b,q,a,f)).
T(b,op(p,b,q,a,f),q).
T(a,p,g).
T(op(p,b,q,a,f),q,c).
a=p|T(p,c,g)|T(p,g,c).
-T(p,c,op(p,b,q,a,f)).
-T(p,c,op(p,b,q,a,op(j,q,a,b,e))).
T(g,q,c).
a=p|T(p,g,c).
-T(g,c,d).
a=p.
-T(p,p,j).
% following 65 should prove inner from outer, temp.beeson.outer.inner.new.out8d6
T(c,p,a).
T(p,c,ext(a,c,x,y)).
T(c,q,b).
-T(q,p,a).
-T(p,q,b).
-T(p,a,b).
-T(b,a,p).
E(c,d,a,c).
T(a,c,d).
T(d,c,a).
T(p,c,d).
T(d,c,p).
-T(c,q,p).
-T(q,c,a).
-T(c,p,q).
E(d,c,a,c).
T(d,p,a).
T(a,p,d).
T(d,q,j).

T(b,j,p).
T(j,q,d).
T(p,j,b).
-T(q,d,a).
-T(a,d,q).
a!=c.
E(a,c,d,c).
T(d,j,e).
T(b,e,a).
T(a,e,b).
-T(j,a,p).
-E(a,c,x,x).
c!=d.
T(q,j,e).
-T(c,d,q).
-T(q,d,c).
T(b,j,f).
T(q,f,a).
b=j|T(j,p,f)|T(j,f,p).
b=j|T(j,f,p)|T(f,p,j).
-T(p,f,b).
-T(b,f,p).
b=j|T(f,p,j)|T(p,f,j).
-T(j,f,p).
-T(p,f,j).
b=j|T(j,p,f).
T(c,q,j)|T(j,p,f).
T(c,q,j)|T(b,p,f).
T(b,p,f)|T(j,q,c).
T(b,p,f)|j=q|T(q,c,d).
T(b,p,f)|T(q,c,d).
T(b,p,f)|T(d,c,q).
T(b,p,f)|d=c.
T(b,p,f).
T(a,p,op(p,b,q,a,f)).
T(b,op(p,b,q,a,f),q).
T(a,p,g).
T(op(p,b,q,a,f),q,c).
a=p|T(p,c,g)|T(p,g,c).
T(g,q,c).
T(p,c,g)|T(p,g,c).
-T(g,c,a).
-T(a,c,g).
T(p,g,c).
T(p,q,c).
T(c,q,p).
% Following 60 prove inner from outer, temp.beeson.outer.inner.new.out8d, using demod six times.
T(c,p,a).
T(a,p,ext(a,c,x,y)).
T(p,c,ext(a,c,x,y)).
T(c,q,b).
-T(q,p,a).
-T(p,q,b).

-T(p,a,b).
-T(b,a,p).
E(c,d,a,c).
T(a,c,d).
T(a,p,d).
T(p,c,d).
T(d,c,p).
-T(c,q,p).
-T(q,c,a).
-T(a,c,q).
-T(c,p,q).
E(d,c,a,c).
T(d,q,j).
T(b,j,p).
T(j,q,d).
T(p,j,b).
a!=c.
-T(c,q,a).
-T(a,q,c).
E(a,c,d,c).
T(d,j,e).
T(b,e,a).
T(a,e,b).
-T(j,a,p).
-E(a,c,x,x).
c!=d.
T(q,j,e).
-T(c,d,q).
-T(q,d,c).
T(b,j,f).
T(q,f,a).
b=j|T(b,p,f)|T(b,f,p).
-T(p,f,b).
-T(b,f,p).
b=j|T(f,p,b).
T(j,q,c)|T(f,p,b).
T(f,p,b)|j=q|T(q,c,d).
T(f,p,b)|T(q,c,d).
T(f,p,b)|T(d,c,q).
T(f,p,b)|d=c.
T(f,p,b).
T(b,p,f).
T(a,p,op(p,b,q,a,f)).
T(b,op(p,b,q,a,f),q).
T(a,p,g).
T(op(p,b,q,a,f),q,c).
a=p|T(a,c,g)|T(a,g,c).
T(g,q,c).
T(a,c,g)|T(a,g,c).
-T(g,c,a).
-T(a,g,c).
-T(a,c,g).
T(c,g,a).

```
T(a,g,c).
end_of_list.

weight_list(pick_and_purge).
weight(ext(a,c,a,c),1).
weight(op(q,b,p,d,c),1).
weight(op(j,b,a,d,p),1).
weight(op(j,q,a,b,e),1).
end_of_list.

list(hot).
ext(a,c,a,c)=d.
op(q,b,p,d,c)=j.
op(j,b,a,d,p)=e.
op(j,q,a,b,e)=f.
end_of_list.
```

With an examination of the given input file, you might find entertainment when you see how many of the earlier Satz theorems are commented out and, therefore, are not needed. Be careful about using neg_weight, for its use can lead to very, very long clauses in the output. As an aside, I have two minimized proofs that differ only by exchanging the use of 3.51 with transitivity of betweenness.

At this point in my research, I was through with a study of the theorem that asserts the deducibility of inner Pasch from outer Pasch in the presence of the 1959 Tarski system given in Section 1. I turned immediately to a direct attempt to prove, with OTTER, that connectivity of betweenness is dependent on the remaining axioms of the 1959 system. Many experiments did not produce a proof of the theorem now in focus. Yes, I thought I was getting quite close, and I may include the approach I was using. After those many attempts, however, finally I followed a path that you may already have traveled. Specifically, I realized that a proof of the theorem was already in hand, if I was satisfied to have such in which OTTER played only an indirect role. The following argument can be made; and, as noted, you may have made this argument paragraphs ago.

First, I had a proof in hand, from OTTER, that inner Pasch is deducible from outer Pasch in the presence of the 1959 Axiom system. Actually, I had many proofs for this relationship. Second, I had a proof, from OTTER, that connectivity is deducible from an axiom system strongly related to the 1959 system but with outer Pasch (two clauses) replaced by inner Pasch (two clauses). Therefore, provided an important condition is met, I had, by transitivity (in effect), a proof that connectivity is deducible from the 1959 axiom system. Have you written down the crucial condition? Yes, regarding the condition, if one of my proofs of inner Pasch from outer avoids the use of connectivity, then all is in order, with OTTER playing the indirect role of supplying the cited first and second items. Earlier I used the word "fortuitous", and earlier I hinted that you might, if you had been witnessing my successes with blocking various input items, already have seen what I was totally unaware of at the time. Indeed, one of the proofs I found in my study of the theorem asserting that outer Pasch implies inner Pasch avoids the use of connectivity. Conclusion: I had, from the viewpoint of mathematics in contrast to the viewpoint of automated reasoning, proved that connectivity of betweenness is dependent on the remaining axioms of the Tarski 1959 system, the axioms that included outer Pasch rather than inner Pasch, and, of course, that excluded connectivity of betweenness.

The remaining nontrivial challenge for me was to find, with OTTER (as opposed to relying on the just-given mathematical argument), a proof of the dependence of connectivity, using the 1959 axiom system offered in Section 1. By looking so-to-speak ahead with the aid of many experiments that did not succeed, I note that various obstacles must be overcome. The two proofs that I could try to rely on, inner from outer and the dependence of connectivity from inner Pasch, are each bidirectional. Put another way, if both proofs were forward proofs, then I could begin by placing as hints their proof steps and have a fair chance of success. Indeed, OTTER could begin with the axiom system based on outer Pasch, with connectivity removed and with various theorems proved from that system, deduce inner Pasch (were it a unit clause, for example), and continue on to prove connectivity (were it also a unit clause).

Well, not so hidden in this last sentence, you find the obstacle that concerns the fact that the clauses for inner Pasch, two of them, and that for connectivity are nonunit clauses. You also note that OTTER will not deduce inner Pasch; the proof I have uses its negation in a bidirectional proof. In review, I have a proof that A implies B and a proof that B implies C and, in the A implies B proof, very important, connectivity is not used. But, in contrast to many cases in the past, I have not as yet, with OTTER, been able to find a proof that A implies C. Even with the input file Beeson sent me to use for a start, although I seem to be close, the prize eludes me. Unless magic is found, I must leave to you the challenge of proving, with OTTER or some other automated reasoning program, the dependence of connectivity within the Tarski 1959 axiom system.

I now discuss how close I have come to winning the game and, just perhaps, provide you with an approach you might use to complete the desired proof. I began by choosing as intermediate targets the unit clauses, seventy-seven of them, that are present among the deduced steps of the 122-step proof that deduced, from inner Pasch and axioms and proved theorems, connectivity. I amended the file Beeson sent me with these negated items in list(passive). The approach was one of iteration. In particular—omitting some of the less interesting experiments—I made a run that proved more than half of the seventy-seven intermediate targets; and, for the next run, I took the proof steps of those proofs and adjoined them to the set of support list. More of the seventy-seven intermediate targets were proved, whose proof steps were further adjoined for the next interesting experiment. After not too long, no additional targets were reached. Now, what might you do? I ask in case you wish to participate in my research.

My reasoning, if that is not too flattering—okay, my guess—was to assume that the proof, if found, might resemble the proof of connectivity that relied on inner Pasch. Since my efforts at proving all of the seventy-seven intermediate targets did not succeed, perhaps it was time to focus on some nonunit clauses. I chose from the Beeson hints2 list three of them, the following, and adjoined them in list(sos) for the next experiment.

```
-T(c1,d,b) | -T(d1,c,b) | T(c,e,c1).
-E(c,d,x,y) | E(c,p,x,y).
-T(d1,c,b)  T(d,e,d1).
```

Rather than detailing the unsatisfactory results, I merely note that I did not find the desired proof. Further experiments led essentially nowhere. If doubt exists about the provability, in the context of automated reasoning, of the theorem under study, I note that Gupta in the 1950s provided a proof that is first-order. When I asked Beeson for a new input file, one based (in effect) on Gupta's proof, fine colleague that he is, he supplied such with a warning that some typos might be present, some perhaps from Gupta's work. That file offers in list(passive) targets that correspond to the steps of the Gupta proof. My experiments with the file yields proofs of quite a few of the intermediate targets. However, the proof of connectivity remained out of reach, even after a variety of attacks.

I therefore provide you with what might be termed an unfinished notebook. To finish it in a manner that I prefer would require my supplying an input file and proof, obtainable from it, that connectivity of betweenness is dependent on the remaining 1959 Tarski axioms. The task—the implied challenge—is, perhaps, overwhelming. I suspect that another notebook will be written that follows this one, with more results concerning Tarskian geometry. For now, I pause and await more joint work with Beeson.