# THE PHILOSOPHY OF AUTOMATED THEOREM PROVING

Francis Jeffry Pelletier
Departments of Philosophy and Computing Science
University of Alberta
Edmonton, Alberta, Canada T6G 2E1

Different researchers use "the philosophy of automated theorem proving" to cover different concepts, indeed, different *levels* of concepts. Some would count such issues as how to efficiently index databases as part of the philosophy of automated theorem proving. Others wonder about whether formulas should be represented *as* strings or as trees or as lists, and call this part of the philosophy of automated theorem proving. Yet others concern themselves with what kind of search should be embodied in any automated theorem prover, or to what degree any automated theorem prover should resemble Prolog. Still others debate whether natural deduction or semantic tableaux or resolution is "better", and call this a part of the philosophy of automated theorem proving. Some people wonder whether automated theorem proving should be "human oriented" or "machine oriented" — sometimes arguing about whether the internal proof methods should be "human-like" or not, sometimes arguing about whether the generated proof should be output in a form understandable by people, and sometimes arguing about the desirability of human intervention in the process of constructing a proof. There are also those who ask such questions as whether we should even be concerned with completeness or with soundness of a system, or perhaps we should instead look at very efficient (but incomplete) subsystems or look at methods of generating models which might nevertheless validate invalid arguments. And all of these have been viewed as issues in the philosophy of automated theorem proving.

Here, I would like to step back from such implementation issues and ask: "What do we really think we are doing when we write an automated theorem prover?" My reflections are perhaps idiosyncratic, but I do think that they put the different researchers* efforts into a broader perspective, and give us some kind of handle on which directions we ourselves might wish to pursue when constructing (or extending) an automated theorem proving system.

A *logic* is defined to be (i) a vocabulary and formation rules (which tells us what strings of symbols are well-formed formulas in the logic), and (ii) a definition of 'proof in that system (which tells us the conditions under which an arrangement of formulas in the system constitutes a proof). Historically speaking, definitions of 'proof have been given in various different manners: the most common have been Hilbert-style (axiomatic), Gentzen-style (consecution, or sequent), Fitch-style (natural deduction), and Beth-style (tableaux). The fact that there are different styles of proof brings up an issue I would like to discuss for a while, and I think the best way to proceed would be to look at a very simple example. Let System A be a classical two-valued propositional logic whose only connectives are -> and -i, having punctuation symbols ) and (, with variables p,q,r,...., and with the usual formation rules for formulas. In such a system we might be given the following (Hilbert) definition of a proof:

axioms: $(p\rightarrow(q\rightarrow p))$
$((p\rightarrow(q\rightarrow r))\rightarrow((p\rightarrow q)\rightarrow(p\rightarrow r)))$
$((\neg p\rightarrow\neg q)\rightarrow(q\rightarrow p))$

Rule 1: From $(\Phi\rightarrow\Psi)$ and $\Phi$, infer $\Psi$

Rule 2: From $\Phi$, infer $\Phi'$ (where $\Phi'$ comes from $\Phi$ by uniform substitution of a sentence for a variable)

A proof of $\Psi$ is a finite sequence of lines $\Phi_1, \Phi_2, \ldots\Phi_n$, where each member of the sequence is either an axiom or follows from earlier lines in the sequence by a rule, and where $\Phi_n=\Psi$.

Now, if one is going to construct a proof *in this system*, one must construct such a sequence of formulas. Anything else does not count as a proof in this system. For example, were THINKER (a natural deduction system, Pelletier 1982) to produce a proof of $\Psi$, it would not be in this system. If one were to generate a truth table for $\Psi$ and show that it contained all T's, that would not be a proof in this system. If one showed that the conjunctive normal form of $\neg\Psi$ yielded the null clause by use of propositional resolution, this would not be a proof in this system. Though the Logic Theorist operated with a slightly different set of axioms, nonetheless any automated proof generator for our system A would have to look very much like it. Nothing else even comes close. But what, then, do we think we are doing when we present the output of THINKER, of a semantic tableau, or of a resolution-based system in response to the query "Is formula $\Psi$ a theorem of system A?"

One answer might be that we believe that if $\vdash_X\Psi'$ then $\vdash_A\Psi$. ($\vdash_Y\Phi$ indicates that there is a proof of $\Phi$ in system Y. 'A' is the axiom system given above and X is one's favourite theorem proving system -- tableaux, resolution, natural deduction, etc. $\Psi'$ is the representation of $\Psi$ within system X.) That is, our generation of $\vdash_X\Psi'$ ensures that *there is a proof* of $\Psi$ in system A -- even if we are not going to produce it. I think it would be worthwhile to keep in mind that if the original goal were to produce a proof in system A, we have not succeeded. Furthermore, such a view surely calls for a demonstration which would convince us of the truth of this background claim: we would want to know that $\vdash_X\Psi$ iff $\vdash_Y\Psi'$ for different proof systems X and Y. Unless one takes the Realist Approach (see below), proving this seems to require some effective method of showing how to convert X-proofs into Y-proofs and conversely. For some simple cases this can be done, but it is not possible to convert automatically proofs from one system into proofs of another. (Consider for example whether there is an effective way to convert propositional resolution proofs into Logic Theorist proofs, and conversely). Besides this, sometimes even the representation in system Y (i.e., $\Psi'$) is not equivalent to the original $\Psi$.

For example skolemized formulas are not equivalent to the original formulas which gave rise to them. (Of course, if the skolemized formula can be proved, then so can the original; but once again this does not constitute a proof of the original, but merely an assurance that *there exists some proof or other* of the original.) So what *are* we doing when we present the output of our automated theorem prover in response to the query of whether some formula is a theorem?

One answer is the Realist answer, according to which there exists an (abstract) object, Propositional Logic, consisting perhaps of a set of valid formulas or perhaps of a set of valid arguments (and maybe there are even other choices....just what one believes an abstract logic to contain depends upon one's philosophic beliefs about abstract objects in general). With such an outlook, the various proof systems merely become different ways for us to discover whether some particular formula (or argument) is a part of Propositional Logic. We start, within this Realist framework, with some idea of what Propositional Logic is; perhaps we think it is the set of theorems, and perhaps we would designate this set by $\models\Phi$. And now, any sound way of convincing ourselves that $\Psi$ is a member of this set is as good as any other — we might construct a proof of $\Psi$ in system A, or with THINKER, or give a tableau, or indeed we might just write out a truth table for $\Psi$. The problem then becomes merely a matter of convincing ourselves that if $\vdash_Y\Psi$ for whatever method Y we happen to use, then $\models\Psi$, that is, of convincing ourselves that system Y is sound. Generally this is pretty straightforward to prove; indeed, too easy, since even if system Y produced *no* proofs it would satisfy this. More difficult is to show completeness: that if not $\vdash_Y\Psi$ then not $\models\Psi$ — i.e., that system Y captures all of the abstract system, Propositional Logic.

The remarks 1 have been making in fact suggest two distinctions: on the one hand (i) a distinction between a particular proof system, such as resolution or system A above, and some abstract object such as Propositional Logic, and on the other hand (ii) a distinction between a particular proof system, such as resolution or system A, and a program written which intends to manifest or emulate that proof system. I would like to dwell a bit on the second of these distinctions. THINKER, for example, is an emulation of Kalish & Montague's (1964) proof system (henceforth KM). It was constructed in such a way that if $\vdash_{th}\Psi$ then $\vdash_{km}\Psi$. But the converse is not true. There are various proofs in KM that will never be produced by THINKER — it just wouldn't make such-and-so inference in that situation or it just wouldn't set so-and-so subgoal at this point even though it was legal to do so in KM; and there are even formulas $\Psi$ for which $\vdash_{km}\Psi$ but not $\vdash_{th}\Psi$ THINKER is, after all, bound by certain emulation/ implementation considerations which dictate finiteness to an extreme degree. For example, all individual variables are amongst e..z together with a subscript 0..9. This means that there are only 220 individual variables available to THINKER; therefore any theorem requiring more than that number of variables cannot be expressed, much less proved. (For example "If there are at least 222 objects then there are at least 221 objects".) Besides that, my computer times THINKER out after a certain period of time — even if THINKER is only one step away from completing a proof. It is for reasons such as these that I would prefer to say that THINKER arid KM are different proof systems. KM is sound and complete: $\vdash_{km}\Psi$ iff $\models\Psi$. But THINKER is not complete; indeed, *no* computer emulation is complete...or so I would prefer to use the term 'complete'. This would mean that no computer emulation really is the same as any of the well-known systems of logic. There is no automated theorem prover which is ("really") resolution, or semantic tableaux, etc.; for these are all complete proof systems. Thus, proving that (say) negative hyperresolution is a complete proof system in no way indicates that any particular system should embody it — for the emulation is bound to be incomplete anyway. There needs to be a different *type* of proof to the effect that such-and-such emulation is better as an emulation of negative hyperresolution than (say) THINKER is as an emulation of KM. There thus seem to be three "Levels" of objects here, according to the Realist: Propositional Logic, different proof systems, and different emulations of some proof system.

Of course, some people wish to abstract away from these "mudane considerations" of finiteness of emulations and talk about idealized emulations. This would be to posit a fourth Level of objects — an idealized emulation somewhere "between" a proof system and any particular emulation of that system. In such a setting the question of completeness becomes more delicate. Here we want to know whether the *strategy* employed in the emulation will (eventually and without considerations of space, memory, or representability) generate everything that the emulated system allows. This is the sort of question posed (and answered negatively) by the authors of the Logic Theorist when they remark that the Logic Theorist could have proved $(\neg p\rightarrow q)\rightarrow(\neg q\rightarrow p)$ even though it never succeeded (because, they say, of time considerations), but its strategies never would have proved $(p\vee\neg\neg p)$, even though the system they were emulating, *Principia Mathematica*, does yield a proof of it. Would THINKER, for example, generate all the proofs allowed by KM? The answer to this is no: as remarked above, THINKER'S strategy makes certain legal moves of KM be unavailable. A more interesting question is whether, for any proof in KM of $\Phi$, is there a proof in THINKER of $\Phi$ (ignoring considerations of finiteness, etc.). The answer to this is unknown, but there is no particular reason to think so. I'm sure that precisely similar remarks could be made for any automated theorem proving system. Suppose system X is "resolution with set of support strategy." We know that X is complete: there is a proof that $\vdash_X\Psi$ iff $\models\Psi$. But whether any particular system which emulates this is complete is another matter, even ignoring issues of "finiteness". For surely this depends on whether the program can always find the correct set of support and then generate all possible resolvants from it.

The Realist position with regards to the automated theorem proving of propositional logic can be illustrated as in Figure 1. For each different logical system, he will have a similar hierarchy. In the hierarchy there are four different "Levels": 1, the abstract object; 2, the various proof

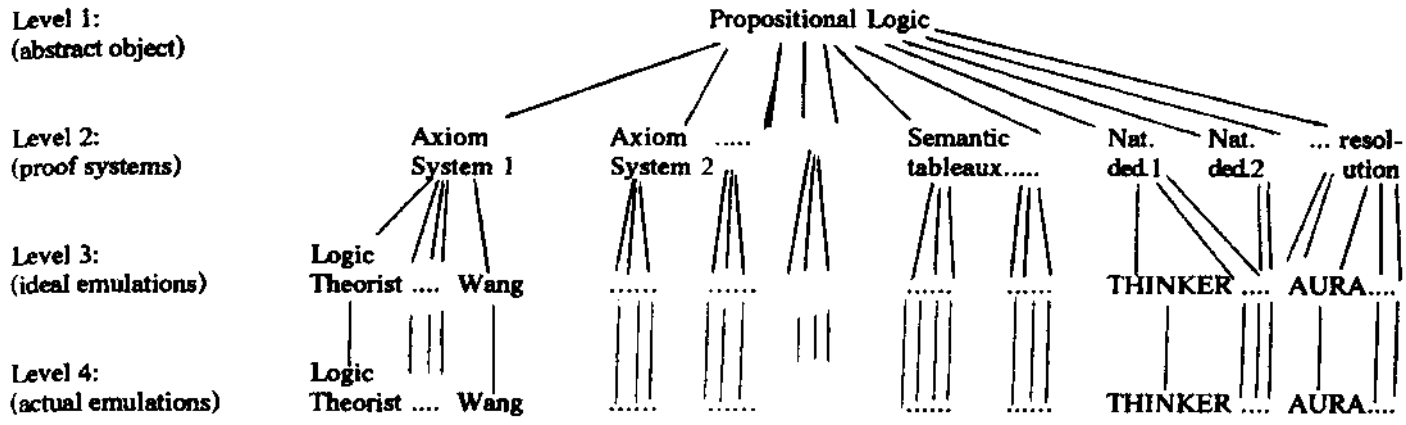


Figure 1: The Realist's View of Automated Proof Systems

systems; 3, the various "idealized emulations" of a proof system; and 4, the various actual emulations of which the Level 3 objects are idealizations. For the Realist, questions of "completeness" can be raised between any two adjacent Levels: one can always ask of something at Level n whether it is "complete for" the object at Level n-1. The Realist believes that logic has answered the question between Levels 2 and 1 — all of the well-known systems of proof are complete. And of course, from the very definition of the relation between a Level 4 object and the corresponding Level 3 object, it follows that they are not complete. The only interesting question then is between Level 3 and Level 2. That is, given the specific strategies used in any particular emulation, but abstracting away from questions of finiteness, will the system "match" the proof theory. As indicated before, the notion of "match" here is a bit tricky. If one meant: will it generate all the proofs (with each step included) that are legal according to the proof theory, then the answer is almost surely no since every particular emulation (except the British Museum Algorithm) embodies some search strategy which eliminates some possible proofs. But the better question is: for every argument $\Gamma \vdash \Phi$ that the proof theory allows, is there a proof of $\Gamma \vdash \Phi$ in the idealized emulation? The idea is that the proof theory allows many (an infinite number) of proofs of $\Phi$ from $\Gamma$; all we require of the ideal emulation is that it find one of them. But this is a question that seems not to be addressed very often in the literature. Researchers do not ask whether such-and-so *actual* system with its *actual* proof-generating strategy that embodies negative hyperresolution (for example) could, considerations of finiteness aside, find a proof of any valid argument of the resolution proof theory. Instead, they choose to investigate whether, for everything that can be proved using ordinary resolution, there is a proof using negative hyperresolution — ignoring the question of whether their particular strategy will succeed in finding it. Writers in the area may have been clear in their own minds what precisely they are proving about completeness, but this does not always seep through to the reader. I myself often wonder whether they aren't in reality inventing a new Level 2 proof system and showing that it is complete with respect to Level 1. But if so, this says nothing about any particular Level 3 system. (There has also been little study of precise conditions under which a particular emulation is incomplete for the idealized emulation. Perhaps this is because people find the specific conditions under which we today compute not very interesting — they will change, after all. But it might nonetheless be interesting to study it, since it is actual implementations — Level 4 objects— which exist in the world and with which we are required to compute and which we are required to use for all those applications that automated theorem proving is claimed to be good.)

As remarked above, one basic attitude someone might take towards logic is a Realist position. As with all philosophical positions, there is an opposing position; here the opposing position is a Nominalism. The Realist believes that there is an antecedently existing system of logic and that all the various proof systems are but different ways to "get at" this antecedently existing system. The Nominalist, on the other hand, believes that there are only the various proof systems. Of course, even a Nominalist could believe that there can be "something in common" amongst various proof systems, or even "something they are striving toward" (in the case of incomplete systems). They may even call this "something" *a semantics or a semantic interpretation* and produce completeness and soundness results for the various systems. But to a Nominalist, this semantics is just *yet* another system — it has no ontological (or epistemological) priority over the various proof systems - even though it might perhaps have been singled out for special consideration because of some interesting properties it was perceived to have (such as essential use having been made of such concepts as "true", "satisfies", etc.).

The Nominalist, then, denies the existence of the Realist's Level 1 abstract logic and instead allows only the various proof theories. What makes a Realist think there is an abstract object in addition to the proof theories, the Nominalist says, is merely due to certain similarities amongst the proof theories. But the study of these similarities does not presuppose the existence of yet another object to be studied, any more than the study of "the average North American academic" (who is white, male, married, has 1.8 children,....) presupposes the existence of such a person. The Nominalist takes the position illustrated in Figure 2 (for the propositional logic, there are further hierarchies for other groupings of proof systems):

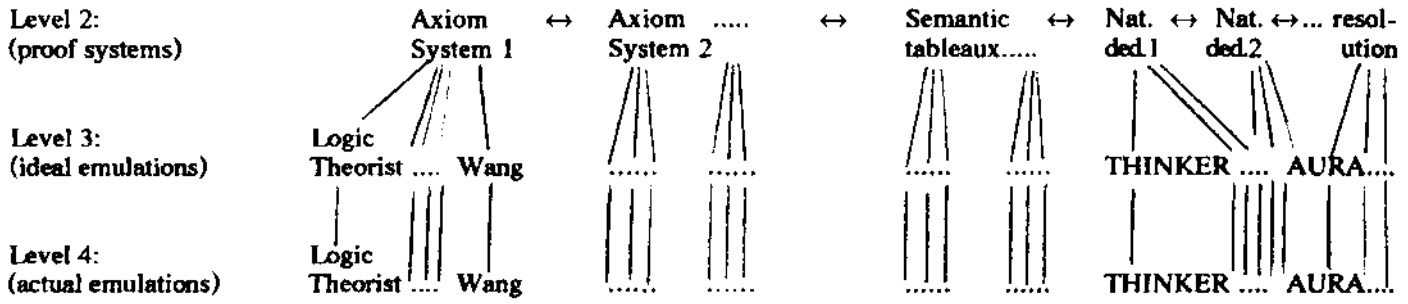


Figure 2: The Nominalist's View of Automated Proof Systems

The ↔'s here, relating the objects on Level 2 to one another, are used to indicate that the Nominalist believes that these proof systems can be compared with one another and certain similarities noted. For the Nominalist, completeness is a somewhat different matter than it is for the Realist. Of course, the Nominalist can ask precisely the same things as a Realist about the relation between a Level 4 object and a Level 3 object, or between a Level 3 object and a Level 2 object. But the notion of completeness of a Level 2 object must be framed not in terms of some abstract object at Level 1, but rather with respect to some special Level 2 system. As remarked above, this is, by his own lights, "just another system". But it is viewed as special due to its unique use of certain concepts -- the so-called "semantic concepts". Still, according to a Nominalist, there is no special *ontological* status to be accorded this system.

What does all this have to do with the business of actually writing an automated theorem proving system, you might be impatient to know at this point. Perhaps not too much, practically; but it seems to me that on a more abstract, philosophical level it addresses the question of whether various practitioners of automated theorem proving have a coherent position. Consider a person who argues that natural deduction is better than resolution-based systems, e.g., Bledsoe (1971) or Pelletier (1982). Whyever would a person be even tempted to argue this unless he thought that the goal of automated theorem proving was to approach "the real system" — that is, unless he were a Realist. After all, it is patently obvious that THINKER is better at emulating proofs in the KM system than any resolution-emulating system; and that any resolution-emulating system is better than THINKER at producing proofs of a resolution system. The same sort of Realist reasoning must have been in Wang's mind when he inveighed against the Logic Theorist and in favour of his consecution-style method (Wang 1960, p.246). But not everyone seems to be a Realist in this sense. These who, like the designers of the Logic Theorist, believe that they are investigating how people reason in the context of a given system (studying substitution into particular axioms, reasoning backward from certain goals dictated by particular axioms, and the like) can be taken to be Nominalists — at least insofar as they would be quite unconcerned with the results of a system where proofs take a different form. And I have spoken with various practitioners in the field who would vehemently deny the existence of any Level 1 object.

As I see it, proof *systems* come in four basic flavours: axiomatic (with a pedigree involving Hilbert 1927 and Whitehead & Russell 1910), resolution-based (with a pedigree of Robinson 1965 and Herbrand $1930_1$ tableau-based (with the pedigree of Beth 1959 and Gentzen 1934/5), and natural deduction (with a pedigree of Fitch 1952 and Gentzen $1934/5^2$). One can write an automated theorem proving program which emulates any particular manifestation of one of these flavours. So long as one concerns oneself with nothing more than the question of how good one's emulation is, one can quite happily be a Nominalist. But should one attempt to argue that their system is better than one of a different flavour, then there is some standard presupposed. And this implies Realism with respect to the standard. Without the standard, there simply are no grounds on which to perform the comparison.

But if you were a Realist interested in automated theorem proving, what would an argument against one or another of these types of systems look like? Against emulating an axiomatic system, one might say that generating proofs which involve arbitrary substitution of sentences for propositional variables just cannot be done efficiently. (Of course there has never been a proof of this, but I think it fair to say that this is the collective wisdom of the automated theorem proving community, as witnessed by the fact that none of the major automated theorem proving systems is an emulation of the axiomatic approach.) The point, for -he Realist, is that *if* you are really interested in "getting at" the abstract logic, you would not try to emulate an axiomatic system, but would rather try to "get at" the logic through another route. The Nominalist of course doesn't have this option: since he thinks there is no abstract logic to "get at", his whole goal is to emulate some particular proof theory. Why would he choose one rather than another proof theory to emulate? For the Nominalist, this must be a matter merely of practicality: which is more interesting/challenging/will get a bigger grant? But there can be no appeal to the Realist's belief that there is an abstract logic and "any way *of* investigating it is legitimate," so we should choose the most efficient way.

[1] Davis (1983, pp.9-12) says that Herbrand (1930) is incorrectly credited here, and that credit should instead be accorded Skolem (1928). What we call 'the Herbrand universe', he says, should be called 'the Skolem universe'. (But he does accord Herbrand with discovery of a version of unification in which a system of equations needs be solved.)

[2] Actually, Gentzen systems (consecution or sequent systems) are a very general format. Although both tableaux and natural deductions might justifiably be said to derive from this format, some authors take pains to keep the three separated. (E.g., Anderson & Bclnap, 1973; see p. xxiv).

We have seen the kind of argument a Realist would give against emulating axiomatic systems — it is too hard and since there are other equally good routes to the abstract logic we should investigate them. But this type of argument is difficult to use against the other types of proof theory since they all seem equally capable of emulation. To give arguments of a similar nature against the other flavours seems to call for our casting out nets a bit wider, and asking whether such systems can be reasonably extended to other areas, e.g., to non-classical logics such as modal logics. If they cannott be, this might count against pursuing such a formalization even in the case of classical logic.

Let us then consider modal logics, for here we can find difficulties both for the Realist and the Nominalist — at least within the framework so far considered. In the logic literature, modal logics are generally presented in an axiomatic form, and it is in only a few special cases that we have alternative formulations. The Realist can at least use these alternative formulations, when they exist, as a means of "getting at" the abstract logic (assuming that he is not interested in trying to emulate axiomatic systems). But a Nominalist who is not willing to emulate an axiomatic system is in a difficult position. Of course he too can emulate any other proof theory which happens to exist, just as the Realist can; but in his own mind he is not "getting at" the same thing that the axiomatic system was describing. And since all the modal systems are given at least an axiomatic formulation, there seems to be an important sense in which the Nominalist is missing some generalization about modal logic in not having some uniform method to deal with the field. The problem, both for the Realist and the Nominalist, is one of converting the axiomatic systems into some other format and then emulating it. But there may be no such transformation -- or at least, no such transformation which is easily emulated. For example, Fitting (1988) reports on the difficulty of transforming the modal system S5 (and by implication, the Brower system KTB, and weaker systems such as KB) into a tableau format; Abadi & Manna (1986) remark that the system KG (as they call it; it is often called KGrz) seems unamenible to a resolution-style format. The resolution systems of Farinas del Cerro (1985) are restricted to a quite small subset of the formulas of any particular modal logic, due to the lack of a normal form representation with which he can apply resolution to these other formulas. Various "doubly indexed" resolution strategies have been adopted in the resolution literature by Wallen (1987), Jackson & Reichgelt (1987), and Olbach (1988); but as Fitting (1988, p. 191) puts it, "it has required considerable coercion" of the underlying logic to implement it. Bonevac (1987) gives a natural deduction format for systems including the T axiom (in particular for KT, KT4, and KT5 ~ the systems often called T, S4, and S5), but it is not at all obvious that any system without this axiom can be given a "natural" natural deduction formulation.

As can be seen, Realist arguments against the use of one or another of the basic proof formats take the form: one shouldn't try to emulate proof system X2 (subscript 2 indicating a "Level 2 object") because there are so many difficulties with doing this that the relation between X4 and X3, or between $X_3$ and $X_2$, will be bad (3 and 4 subscripts indicating the Level 3 or 4 item of proof system $X_2$). But since we are just trying to "get at" $X_1$, the Realist will believe that it does not matter whether sometimes one uses one kind of Level 2 system and at another time we use a different type of Level 2 system. Indeed we can even mix the different types up and use them simultaneously. The Realist believes these are all good ways to investigate the abstract Level 1 object. It is these kinds of strategies that a Realist can pursue with a clear conscience but a Nominalist cannot, bound as he is to particular proof systems. So, if one is to be a Nominalist with respect to modal logics generally, it looks like one's range is quite narrowed — unless one is going directly to emulate axiomatic systems. But there are two more strategies that haven't yet been mentioned, and at least one of them could be used with equinimity by a Nominalist. These methods involve an "ascent to the metalanguage", and were (I believe) first mentioned in the automated theorem proving literature in Morgan (1976) where they were called "syntactic" and "semantic" methods.

As described, a Nominalist believes that there are proof systems -- he denies only the independent existence of some further, abstract object to which any proof system is merely a means of access. Thus a Nominalist would have no ontological qualms with reasoning *about* such a proof system. Of course, this reasoning must be done within a system, according to his Nominalistic views; but this raises no particular problem — he can use any of the various proof systems (which he does believe to exist) to reason about any particular system. For example, our Nominalist might wish to reason about our system A, and wish to do so by invoking the use of some quantified resolution system (say). Here our Nominalist views the various formulas of A as being objects, and views the claim $\vdash_A \Phi$ as expressing that the object denoted by the formula $\Phi$ has a certain property, which he may choose to express as $\mathbf{ThmA(\Phi)}$. And in system A, formulas have a structure. This structure is represented in our Nominalist's system by function symbols. For example, the unary sentential operator -1 of system A is perhaps represented as the unary function symbol n, so that formula '$\neg p$' of system A becomes the term *n(p)' of the Nominalist's system. The binary sentential operator '$\rightarrow$' of A is perhaps represented by the binary function symbol i, so that '$(p \rightarrow q)$' of system A becomes 'i(p,q)\ The axioms of system A merely become universally quantified premises of each argument in the Nominalist's system. Thus axiom 1 of system A becomes

$$\mathbf{ThmA(i(x,i(y,x)))}$$

And this is used as a premise whenever we wish to determine $\mathbf{ThmA(\Phi)}$ for some $\Phi$. Rules of inference of system A, such as our Rule 1, are expressed as saying that every pair of formulas that are related to one another in a certain way and to which the predicate 'ThmA' can be applied, give rise to another formula to which 'ThmA' can also be applied. That is, Rule 1 gets represented as

$$\mathbf{\neg ThmA(x) \vee \neg ThmA(i(x,y)) \vee ThmA(y)}$$

For an automated theorem proving Nominalist, then, this is the strategy to pursue when faced with a proof system for which there is no transformation into some other proof

system for which he has an automated theorem proving emulator. Examples of systems which have adopted this strategy are: Morgan (1985), Paulson (1988), Quaife (1988). Pelletier (1986) had rated this kind of strategy as being very difficult to pursue (see the discussion surrounding problems #66-70), but the results reported in Quaife (1988) seem to show that this was much too pessimistic.

Since the Realist too acknowledges the existence of these proof systems, he too can use this "syntactic method." But since he in addition acknowledges the existence of an abstract system, he can feel free to investigate it directly. Let us ask ourselves again, just *what* is this abstract system? According to the Realist, it is the set of valid formulas (or arguments) towards which the various proof systems are a means of access. This way of looking at the abstract system makes it be characterized by certain "semantic" notions, such as 'True' and 'False' (for our system A) or True in a given possible world with a specific accessibility relation R' (for a modal system). This means that the Realist would find it acceptable to write a truth table for a formula of system A in order to determine whether it belonged to Propositional Logic. But a Nominalist ought not to be able to take this route, since he denies the very existence of Propositional Logic. It means that a Realist ought to be happy to write a "transitive, reflexive possible world diagram" for a sentence in the language of $S_4$ to find out whether it really belonged to the abstract system, S4. But again, a Nominalist ought not do this — he refuses to believe that there is an abstract S4 to do this to, but rather thinks there are only the various proof systems which are presentations of S4. Indeed, since a Realist believes that all these things exist, he could quantify over the possible worlds and could make one-place predicates P(x), Q(y),... mean "(proposition) p is true at world x". To say that the formula Lp is true at a given possible world x is to say that p is true at every possible world y which is accessible from x. All this can be represented by the realist as

$$(\forall y)(W(y)\&R(x,y)\rightarrow P(y))$$

("W" means that y is a possible world; "R" is the accessibility relation saying that y is accessible from x). A formula is a theorem of any normal modal logic just in case it is true at every possible world. The different (normal) modal logics differ only on what extra requirements they impose upon the accessibility relation R. And these requirements can be added as premises to the representation of an argument of any such modal system. This is Morgan's (1976) "semantic method". It has been incorporated into THINKER with some considerable success for standard modal systems.

Of course, to use the semantic method correctly one must know what the correct underlying logic of the abstract system is. The examples of constructing truth tables for Propositional Logic and of quantifying over possible worlds in S4 have assumed that their logic is truth functional and classical first order quantification theory. In these cases that assumption is correct, but some other cases are not so clear and there are yet others for which this is quite false. For example, some proponents of many-valued logics have argued that "the metalanguage of their proof systems" (or as I would prefer to say: the abstract object) it itself many-valued, not classically two-valued. And demonstrably there are modal logics - even normal modal logics — for which there is no characteristic first-order condition on the accessibility relation amongst possible worlds. For example, although the axiom (Grz)

$$L(L\Phi\rightarrow\Phi)\rightarrow L\Phi$$

has a first-order model, the closely-related axiom

$$L(L\Phi\leftrightarrow\Phi)\rightarrow L\Phi$$

does not. Therefore, the specific "semantic method" given above, which assumed that the abstract system has a first-order underlying logic, cannot be used here. Some other underlying logic system would have to be developed and implemented to investigate directly this abstract system of this modal logic. (For further considerations on this peculiar state of affairs, see Hughes & Cresswell 1986, under the heading "What do we mean by 'completeness'.")

Of course, the Nominalistic-oriented "syntactic method" suffers no such shortcoming, since it is not trying to *use* the "underlying logic of the abstract system" but only generates proofs that are in accord with it. Some might think that this greater flexibility in the realm of non-standard logics is precisely what the theorem-proving community needs. But others may be attracted to the view that for any logic worthy of the name there "really is" an abstract system; and that unless we have a characterization of this abstract system and can therefore work directly with it, the advantage afforded by the "syntactic method" is only a chimera. For, until we know a logical system is "really there", there is no reason to waste time studying it.

Who is right in this matter I leave it to the reader to judge.

## Acknowledgments

The author is a member of the Institute for Robotics and Intelligent Systems and wishes to acknowledge the support of the Networks of Centres of Excellence Program of the Government of Canada, NSERC, and the participation of PRECARN Associates Inc.

## References

Abada, M. & Manna, Z. (1986) "Modal Theorem Proving" *CADES.* (Springer-Verlag).pp. 172-179.

Anderson, A. & Belnap, N. (1973) *Entailment* . Princeton Univ. Press.

Bledsoe,W. (1971) "Splitting and Reduction Heuristics in Automated Theorem Proving" *Artificial Intelligence* pp.57-78.

Beth, F. (1959) *The Foundations of Mathematics.* North-Holland.

Bonevac, D. (1987) *Deduction* Mayfield Pub. Co.

Davis, M. (1983) "The Prehistory and Early History of Automated Deduction", in Siekmann & Wrightson, pp. 1-28.

Farinas del Cerro, L. (1985) "Resolution Modal Logic" *Logique et Analyse* .pp. 153-172.

Fitch, F. (1952) *Symbolic Logic.* Roland Press.

Fitting, M. (1988) "First Order Modal Tableaux" *Journal of Automated Reasoning* pp. 191-213.

Gentzen, G. (1934/5) "Investigations into Logical Deduction". Translation printed in M. Szabo *The Collected Papers of Gerhard Gentzen* (North-Holland) 1969, pp. 68-131.

Herbrand, J. (1930) "Investigations in Proof Theory". Translation printed in W. Goldfarb *Jacques Herbrand -- logical Writings (Harvard* Univ. Press) 1971.

Hilbert, D. (1927) "The Foundations of Mathematics". Reprinted in J. van Heijenoort *From Frege to GO del* (Harvard Univ. Press) 1967.

Hughes, G. & Cresswell,M. (1986)-A *Companion to Modal Logic.* Methuen.

Jackson, P. & Reichgelt, H. "A General Proof Method for First Order Modal Logic" *UCAI-10.* pp.942-944.

Kalish, D. & Montague, R. (1964) Lo*gic* (Hartcourt, Brace, Janovich).

Konolige, K. (1986) "Resolution and Quantified Epistemic Logics" *CADES.* (Springer-Verlag). pp. 199-208.

Morgan, C. (1976) "Methods for Automated Theorem Proving in Non-Classical Logics" *IEEE Trans, on Computers* pp.852-862.

Morgan, C. (1985) "Autologic" *Logique et Analyse* pp. 257-282.

Olbach, H. (1988) "A Resolution Calculus for Modal Logics". *CADE-9* (Springer-Verlag). pp.500-516.

Paulson, L. (1988) "1SABELLE: The Next 700 Theorem Provers" *CADE-9.* (Springer-Verlag). pp.772-773.

Pelletier, F.J. (1982) *Completely Non-Clausal Completely Heuristic-Driven, Automatic Theorem Proving.* Tech. Report TR82-7, Dept. Computing Science, Univ. Alberta.

Pelletier, F.J. (1986) "Seventy-Five Problems for Testing Automated Theorem Provers" *Journal of Automated Reasoning,* pp.l91-216,and "Errata"*ibid.* 1988 pp.235-236.

Pelletier, F.J. (1987) *Further Developments in THINKER, an Automated Theorem Prover.* Tech. Report TR-ARP-16/87 Automated Reasoning Project, Australia National University.

Quaife, A. (1988) "Automated Proofs of Lob's Theorem and Godel's Two Incompleteness Theorems" *Journal of Automated Reasoning,* pp.219-23L

Robinson, J. (1965) "A Machine Oriented Logic Based on the Resolution Principle" *J ACM.* 23-41.

Siekmann,J. & Wrightson,G. (1983) *The Automation of Reasoning, Vol 1.* Springer-Verlag.

Skolem, Th. (1928) "On Mathematical Logic". Translation printed in J. van Heijenoort *From Frege to Godel* (Harvard Univ. Press) 1967.

Thistlewaite, P., M.McRobbie, & R.Meyer (1988) *Automated Theorem Proving in Non-Classical Logics.* Pitman.

Wallen, L. (1987) "Matrix Proof Methods for Modal Logics". *UCAI-10.* pp.917-923.

Wang, H. (1960) "Toward Mechanical Mathematics". Reprinted in Siekmann & Wrightson 1983, pp. 244-264.

Whitehead,A. & Russell,B. (1910) *Principia Mathematica, Vol I.* Cambridge Univ. Press.

Wos, L. (1987) *Automated Reasoning: 33 Basic Problems.* Prentice-Hall.