INTEREST DRIVEN SUPPOSITIONAL REASONING

John L. Pollock
Department of Philosophy
University of Arizona
Tucson, Arizona 85721
(e-mail: pollock@ccit.arizona.edu)

Abstract. The aim of this paper is to investigate two related aspects of human reasoning, and use the results to construct an automated theorem prover for the predicate calculus that at least approximately models human reasoning. The result is a non-resolution theorem prover that does not use Skolemization. It involves two central ideas. One is the interest constraints that are of central importance in guiding human reasoning. The other is the notion of suppositional reasoning, wherein one makes a supposition, draws inferences that depend upon that supposition, and then infers a conclusion that does not depend upon it. Suppositional reasoning is involved in the use of conditionals and reductio ad absurdum, and is central to human reasoning with quantifiers. The resulting theorem prover turns out to be surprisingly efficient, beating most resolution theorem provers on some hard problems.

Keywords. Automated theorem proving, reasoning, natural deduction.

1. Introduction

The typical automated theorem prover is based on resolution and Skolemization.[1] The original appeal of these two procedures was that they took advantage of the unique data processing capabilities of computers. They were not intended to model ordinary human reasoning, and they clearly do not. My investigation of theorem proving began as an attempt to model certain aspects of human reasoning, and my expectation was that the resulting theorem prover, OSCAR, would be quite slow in comparison to theorem provers designed to take advantage of the special abilities of computers. Instead, OSCAR has turned out to be surprisingly efficient, doing some hard problems faster than most resolution-based theorem provers.

[1] For instance, Lusk et al [9], [10], Stickel [19], [20], Wos and Winker [21], and others too numerous to mention.

The deductive reasoner described here is part of a more general reasoner that performs both deductive and defeasible reasoning. The original reason for exploring non-resolution reasoners in the OSCAR project was that it is not clear that the completeness of resolution refutation carries over in any form to defeasible reasoning. Whether it does can only be established once we have a better understanding of how defeasible reasoning works. Accordingly, I set about building a deductive reasoner that could be extended to a defeasible reasoner, and this paper describes the result.[2]

I approach the the topic of reasoning from the perspective of the epistemologist, who is primarily interested in understanding the structure of human rational thought. Human rational architecture is described by rules for updating one's set of beliefs. A feature of these rules that has recently been emphasized by Gilbert Harman [9] is that they do not mandate the adoption of every belief for which we have good reasons. For example, each belief *P* gives us a conclusive reason for any disjunction (*P* ∨ *Q*) containing *P* as a disjunct, but we do not arbitrarily adopt such disjunctions. Our epistemic rules must honor the fact that we are information processors with a limited capacity for processing and storage. If belief adoption is not mandated simply by the possession of good reasons, what more is required? It seems clear that it has something to do with interests. If we are interested in knowing the value of 23+57, we reason differently than if we are interested in knowing what time it is in Moscow. We do not just reason randomly until we happen upon the answers to our questions. In short, our rules for belief formation involve interest driven reasoning.

In an important sense, any functioning theorem prover is interest driven. What the theorem prover does is determined at least in part by what it is trying to prove. The only alternative is the totally impractical one of generating proofs at random until the system happens on a proof of the desired theorem. However, the ways in which specific theorem provers are interest driven tends to be determined by *ad hoc* solutions to particular problems of cognitive engineering rather than by general principles applicable to all reasoning. This paper will make some general proposals regarding the interest driven features of human reasoning and use these proposals as a guide in the construction of OSCAR. No attempt will be made to muster psychological evidence for the claim that human reasoning works this way, because that is really beside the point. The observations about human reasoning serve only to motivate the construction of OSCAR.

2. Two Kinds of Reasons

In trying to prove a theorem in mathematics, or trying to construct a derivation in logic, we sometimes cast about more or less at random, drawing whatever conclusions we can until we eventually see some way to put them together to get to our desired conclusion. At other times we proceed in a much more deliberate fashion, in effect reasoning backwards from our desired conclusion. We can think of such backwards reasoning as deriving

---

[2] The first incarnation of OSCAR is described in [17]. It was not interest driven, and the present investigation began as an attempt to incorporate interest constraints into OSCAR. A more general account of OSCAR can be found in [20].

interests from interests. In more conventional terminology, it is a species of goal chaining. Although goal chaining is the standard paradigm in much of AI, for reasons that are unclear to me, it plays little explicit role in contemporary work on automated theorem proving. It can be regarded as implicit in set-of-support strategies, but in human reasoning it seems to play an explicit and central role.

Our reasoning usually involves a combination of backwards reasoning and random forwards reasoning. It is natural to suppose that backward reasoning is just forward reasoning done in reverse, and that in any given instance we *could* just reason in one direction or the other without combining the two. But I will claim that this view of reasoning is fundamentally mistaken. There are profound differences in the structure of random forward reasoning and backwards reasoning, and neither is replaceable by the other. They are both absolutely essential for reasoning, and there is a definite logic governing when to use one and when to use the other. The choice is not just a matter of convenience. We literally could not arrive at our conclusions in any but the simplest cases without using both kinds of reasoning. The first intimation of this comes from considering the simplest rules for backwards and forwards reasoning.

In the general case, a reason for *q* is a set of propositions. Then, at least in the simplest cases, random forward reasoning proceeds straightforwardly in terms of a rule like:

R-DEDUCE:

> If *X* is a reason for *q*, and you adopt some member of *X* and already believe the others, then adopt *q*.

Backwards reasoning proceeds in terms of a pair of rules roughly like the following:

INTEREST-ADOPTION:

> If you are interested in *q*, and *X* is a reason for *q*, then become interested in the members of *X*. If you already believe all the members of *X* then adopt *q*.

DISCHARGE-INTEREST:

> If you are interested the members of *X* as a way of getting *q*, and you adopt some member of *X* and already believe the others, then adopt *q*.

These must be supplemented with rules telling us to cancel interest in propositions under various circumstances. I will pursue the details of these various rules in the next section. But before doing that, let us reflect upon various schemas of conclusive (i.e., deductive) reasons that ought to be useable in a system of deductive reasoning. The following simple entailments are all plausible candidates for conclusive reasons:

Backwards reasons:

adjunction:	$\{p,q\} \vdash (p \,\&\, q)$
addition:	$p \vdash (p \vee q)$
	$q \vdash (p \vee q)$
~~ introduction:	$p \vdash {\sim\sim}p$

Forwards reasons:

simplification:	$(p \,\&\, q) \vdash p$
	$(p \,\&\, q) \vdash q$
~~ elimination:	$\sim\sim p \vdash p$
modus ponens:	$\{(\sim p \vee q), p\} \vdash q$

What should be observed is that the reasons listed under "backwards reasons" are *only* of use in backwards reasoning, and those listed under "forwards reasons" are only of use in random forward reasoning. For instance, consider addition. Suppose we could use addition in random forward reasoning. Then if we adopted p, we would be led to adopt every disjunction containing p as one disjunct. But there are infinitely many such disjunctions, and most are useless in any given problem. In point of fact, we only use addition when we have some reason to be interested in the resulting disjunction. Much the same point can be made about adjunction. Suppose we could use it in random forward reasoning, and we adopted p and q. Then we would be led to adopt $(p \,\&\, q)$. That does not seem so bad, but it would not stop there. We would go on to adopt $[(p \,\&\, q) \,\&\, p]$, $(q \,\&\, p)$, $[(q \,\&\, p) \,\&\, (p \,\&\, q)]$, $[(q \,\&\, p) \,\&\, [(p \,\&\, q) \,\&\, p]]$, $(p \,\&\, [(q \,\&\, p) \,\&\, [(p \,\&\, q) \,\&\, p]])$, and so on without limit. Obviously, we do not do that, and a reasoning system that performed in this way would be crippled. This largely useless reasoning would continually get in its way, taking up its resources and preventing it from making more useful inferences.

The use of simplification in backward reasoning would be even more catastrophic. Suppose we are interested in q. Backwards reasoning with simplification would lead us to adopt interest in $(p \,\&\, q)$, for *every* p, and then backwards reasoning with adjunction (which is presumably permissible) would lead us to adopt interest in p. Thus interest in anything would automatically lead to interest in everything, which would completely vitiate the interest restrictions in interest driven reasoning. Similar reflection on the other reasons indicates that in each case, they can only function in the category in which they are listed.

The somewhat surprising conclusion to be drawn from this is that some reasons are of use in backward reasoning, and others are of use in random forward reasoning, and individual reasons may not play both roles. Reasons can be classified as *forwards reasons*, which can be used in random forward reasoning in accordance with *R-DEDUCE*, and *backwards reasons*, which can be used for backwards reasoning in accordance with *INTEREST-ADOPTION* and *DISCHARGE-INTEREST*.

As described, a backwards reason is an entailment $p_1,...,p_n \vdash q$ such that adopting interest in q leads us to adopt interest in $p_1,...,p_n$. There can also be "partial backwards reasons" where adopting interest in q leads us to adopt interest in $p_{i+1},...,p_n$ *provided* we have already established $p_1,...,p_i$. Modus ponens is an example of this. Adopting interest in q should not lead us to adopt interest in p and $(p \supset q)$ *for every* p. But if we have already established $(p \supset q)$, then adopting interest in q can reasonably lead us to adopt interest in p.

To put things in perspective, notice that resolution-based systems use resolution as a random forward reason and combine it with a form of reductio ad absurdum (see below).

3. Linear Interest Driven Reasoning

OSCAR will consist of a set of rules for interest driven deductive reasoning proceeding in terms of forwards reasons and backwards reasons. I will begin by focusing on rules for linear reasoning (i.e., reasoning that does not involve rules like *reductio ad absurdum* or *conditionalization*, which require subsidiary arguments and suppositional reasoning). Then I will extend the system to encompass suppositional reasoning. This will culminate in a system of interest driven reasoning for the propositional and predicate calculi.

We come to a deductive reasoning situation with a set of premises that are somehow "given" and a set of propositions in which we are interested. These comprise the set *input* from which reasoning begins, and the set *ultimate* of ultimate interests. As the various rules operate, OSCAR will acquire a backlog of propositions waiting to be adopted as beliefs and other propositions in which it is to adopt interest. These will be stored in the *adoption-queue* and the *interest-queue*, respectively. The elements of *adoption-queue* will have the form $\langle p,X\rangle$, signifying that p is to be adopted on the basis of X. The elements of the *interest-queue* will have the form $\langle p,X_0,X,q\rangle$. This will signify that p is of interest as a way of deriving q from the set X of propositions. X_0 is the set of all propositions in X other than p that have not yet been adopted. The general architecture of OSCAR is as follows:

(linear-process-beliefs)

(1) for all p in *ultimate*, insert $\langle p,\varnothing,\varnothing,\varnothing\rangle$ into *interest-queue*;[3]
(2) adopt all members of *input*;
(3) do the following repeatedly until *interest-queue* and *adoption-queue* are empty:
    (a) if *interest-queue* is nonempty then (process-interest-queue);
    (b) if *interest-queue* is empty but *adoption-queue* is nonempty then (process-adoption-queue).

The idea behind steps 2 and 3 is that OSCAR first adopts the propositions given as input, and adopts interest in the propositions that are to comprise the ultimate interests. These initial operations will typically lead to other propositions being inserted into *adoption-queue* and *interest-queue*. OSCAR continues adopting interest in members of *interest-queue* until it is empty. Then it turns to *adoption-queue* and begins adopting its members. Whenever this leads to something new being inserted into *interest-queue*, OSCAR adopts interest in that before continuing processing *adoption-queue*.

process-interest-queue is as follows:

(process-interest-queue)

If $\langle p,X_0,X,q\rangle$ is the first member of *interest-queue* then:

(1) if $p\in$*adoptions*, let X^* be the set of all elements of X_0 not in *adoptions*, and:
    (a) if $X^* = \varnothing$ then insert $\langle X,q\rangle$ into *adoption-queue*;
    (b) otherwise, where r is the first member of X^*:
        (i) insert $\langle r,(X^*-\{r\}),X,q\rangle$ into *forset*;
        (ii) (adopt-interest r);

---

[3] For p in *ultimate*, the last three elements of the quadruple that is inserted into *interest-queue* are never used for anything, so we arbitrarily let them be .

(2) if $p \notin$ *adoptions* and $\neg p$ *adoptions* then:
  (a) insert $\langle p,X_0,X,q\rangle$ into *forset*;
  (b) (adopt-interest *p*).

The set *forset* is a bookkeeping device allowing OSCAR to keep track of why it is interested in particular propositions. It will be used by discharge-interest. Adopted propositions (beliefs) will be put in the set *adoptions*, and propositions in which OSCAR has adopted interest will be put in the set *interest*. Adopting interest in a proposition will consist of the following:

(adopt-interest *p*)
(1) if for some *X*, $\langle p,X\rangle$ *adoption-queue* then (adopt *p*);
(2) otherwise:
  (a) insert *p* into *interest*.
  (b) (interest-adoption *p*)
(3) delete the last member of *interest-queue*.

(interest-adoption *p*)
Given any set *X* of propositions, if $\langle X,p\rangle$ is an instance of a backwards reason:
(1) if all members of *X* are in *adoptions* then insert $\langle p,X\rangle$ into *adoption-queue*;
(2) if not all members of *X* are in *adoptions* and no member of *X* is such that its negation is in *adoptions*, then where X_0 is a list of all elements of *X* not in *adoptions* and *q* is the first member of X_0, insert $\langle q,(X_0\text{-}\{q\}),X,p\rangle$ into *interest-queue*.

In turn, process-adoption-queue is as follows:

(process-adoption-queue)
If $\langle p,X\rangle$ is the first member of *adoption-queue* then:
(1) insert $\langle p,X\rangle$ into *basis*;
(2) (adopt *p*).

The set *basis* keeps a record of the reasoning. This is used in turning the reasoning into a deductive proof when the reasoning is completed.

(adopt *p*)
(1) insert *p* into *adoptions*;
(2) delete *p* from *ultimate*;
(3) if this results in *ultimate* being empty, terminate processing;
(4) (discharge-interest *p*);
(5) (r-deduce *p*);
(6) (cancel-interest *p*);

(7) if $\neg p$ *ultimate*, then (cancel-interest $\neg p$);[4]
(8) delete the first member of *adoption-queue*.

(r-deduce *p*)

Where *p* is newly adopted, if *X* is a forwards reason for *q*, *p X*, and all other members of *X* are already members of *adoptions*, then insert $\langle p,X\rangle$ into *adoption-queue*.

Rather than inserting every $\langle r,X_0,X,p\rangle$ for *r* in X_0 into *interest-queue* and subsequently into *forset*, it is more efficient to insert them one at a time. If OSCAR succeeds in deducing the first member of X_0, then it begins looking for the next member, and so on. In that way if OSCAR is unable to obtain some element of X_0, it does not waste time looking for the other members. Accordingly, discharge-interest is as follows (where the first member of *adoption-queue* has the form $\langle p,Z\rangle$):

(discharge-interest *p*)

If $p\in$*interest*, then for any $\langle p,X_0,X,q\rangle$ in *forset*, delete $\langle p,X_0,X,q\rangle$ from *forset*, and where X^* is the set of all members of X_0 not in *adoptions*:
(1) if $X^* = \varnothing$ then insert $\langle q,X\rangle$ into *adoption-queue*;
(2) otherwise, where *r* is the first member of X^*, insert $\langle r,X^*-\{r\},X,q\rangle$ into *interest-queue*.

In addition to the rules for interest adoption, there must be rules for interest cancellation. If *p interest* and either *p* or $\neg p$ is adopted, the truth value of *p* has thereby been established and OSCAR should cease to be interested in *p*. In addition, when interest in *p* is cancelled, then if $\langle q,X_0,X,p\rangle$ *interest-queue* it should be deleted from *interest-queue*; and if $\langle q,X_0,X,p\rangle$ *forset*, it should be deleted from *forset* and OSCAR should cancel interest in *q* if there is no other reason to be interested in *q*. This is captured by having (adopt *p*) perform (cancel-interest *p*) and (cancel-interest $\neg p$), where cancel-interest is defined recursively as follows:

(cancel-interest *p*)

(1) delete *p* from *interest* and *ultimate*;
(2) for any σ in *interest-queue* such that *p* is the first or the last element of , delete σ from *interest-queue*;
(3) for every q,X_0,X such that $\langle q,X_0,X,p\rangle$ *forset*:
    (a) delete $\langle q,X_0,X,p\rangle$ from *forset*;
    (b) if *q ultimate* and there is no σ in *forset* such that *q* is the first member of , then (cancel-interest *q*).

With these definitions and an array of conclusive reasons, OSCAR becomes a system of interest driven deductive reasoning. It can actually perform simple reasoning tasks. However, it is grossly inadequate as a general deductive reasoner, for the reasons that will

[4] If $p = \sim q$ for some *q* then $\neg p$ is *q*; otherwise $\neg p$ is $\sim p$.

be explained in the next section.

4. Non-linear Reasoning and Conditionalization

Thus far I have talked exclusively about rules for linear reasoning. Linear reasoning proceeds in terms of arguments that do not involve subsidiary arguments. One of the most important features of human deductive reasoning is that it is not exclusively linear. The simplest way to see that this must be the case is to note that deductive reasoning can lead to *a priori* knowledge of "truths of reason". For instance, we can obtain conclusions like [(*p* & *q*) ⊃ *q*] or (*p* ∨ ~*p*) that do not depend upon any premises. Linear reasoning can only draw conclusions from previously adopted beliefs, so such *a priori* knowledge cannot be obtained by linear reasoning. Instead, we employ several kinds of rules for the construction and use of subsidiary arguments, most notably, *conditionalization*, *reductio ad absurdum*, *dilemma*, and *universal generalization*. In this section, I will introduce *conditionalization* as the simplest form of nonlinear reasoning. In subsequent sections I will discuss *reductio ad absurdum*, *dilemma*, and quantifier rules.

The employment of subsidiary arguments comprises *suppositional reasoning*, wherein we suppose something "for the sake of argument", reason using the supposition in the same way we reason about beliefs and interests nonsuppositionally, and then on the basis of conclusions drawn using the supposition we draw further conclusions that do not depend upon the supposition. For instance, in conditionalization, in order to establish a conditional (*p* ⊃ *q*), we "suppose" the antecedent, try to derive the consequent from that supposition, and then "discharge" the supposition to infer the conditional. Similarly, in reductio ad absurdum, if we are trying to establish *p*, we may suppose ¬*p*, show that that leads to a contradiction, and then discharge the supposition to infer *p*. In both of these kinds of reasoning, what is crucial is the use of suppositions to get conclusions related to our desired conclusions, and their subsequent discharge. Within the supposition, we reason as if the supposed propositions were beliefs, using all the rules for adoption and interest adoption that were discussed in connection with linear reasoning.[5]

In implementing suppositional reasoning, we must have separate sets of adoptions, interests, ultimate interests, adoption-queues, interest-queues, and so forth, for each supposition. I will take a supposition to be a data structure with slots for each of these lists. The "empty supposition", where we suppose nothing, will be given the name `*beliefs*'.

When OSCAR makes a new supposition, it is always for some specific purpose, and hence the circumstances dictate what is of ultimate interest relative to the supposition. Suppositions can be nested, because OSCAR may suppose something more extensive in order to establish something relative to a smaller supposition. OSCAR may also make a supposition *A* at some point with a set of ultimate interests *C*, and then later make the same supposition for the purpose of establishing some different conclusions. In this case OSCAR

---

[5] Some theorem provers that utilize varieties of suppositional reasoning, although without exploring the general structure of suppositional reasoning, are described in [3], [5], [6], [13], and [21].

just adds these new interests to (ultimate *A*) and goes on reasoning. In general, we can define the notion of *making a supposition with supposed propositions A and ultimate interests C* as follows:

(make-supposition *A C*)

(1) If there isn't already a supposition *X* with (suppose *X*) = *A*, create one and set:
(ultimate *X*) = *C*
(interest-queue *X*) = *C*
(adoption-queue *X*) = (*A* ∪ *input*)
and leave all the other slots empty.

(2) If instead there is such a supposition *X*, set:
(ultimate *X*) = *C* ∪ (ultimate *X*)
(interest-queue *X*) = *C* ∪ (interest-queue *X*)
(adoption-queue *X*) = *A* ∪ (adoption-queue *X*)
and leave all the other slots unchanged.

The functions adopt, adopt-interest, discharge-interest, and so on, must now take a second variable. For instance, (adopt *p A*) means "adopt *p* in the supposition *A*". Similarly, (linear-process-beliefs) is replaced by (linear-process *A*), which can be applied to any supposition.

Conditional reasoning involves two basic procedures over and above those for linear reasoning. One is for making suppositions, and the other is for discharging suppositions. The basic rule for making a supposition tells us that if interest is adopted in a conditional (*p* ⊃ *q*) in a supposition *A*, then make a new supposition *A** wherein *p* is added to what is supposed in *A*, and put *q* into (ultimate *A**). OSCAR must also keep track of why it wants *q*. We can use (forset *A**) for this purpose, but in order to do so we must give its members a more complex structure. We will take them to be sextuples $\langle q,\alpha,B_0,B,p,S\rangle$. This signifies that OSCAR is trying to derive *p* in the supposition *S* from *B* in accordance with a rule of inference identified by the keyword . For conditionalization, α will be *condit*, and for linear reasoning in accordance with interest-adoption α will be *con* (for "conclusive reason"). *B* can no longer be a set of propositions, because in suppositional reasoning we may infer *p* in *S* on the basis of obtaining conclusions in other suppositions. Accordingly, *B* will be taken to be a set of pairs $\langle r,S^*\rangle$ of propositions and suppositions. B_0 will be the set of all $\langle r,S^*\rangle$ in *B* other than $\langle q,S\rangle$ such that *r* (adoptions *S**). The rule for making suppositions in conditional reasoning is then:

(c-suppose *p A*)

If *p* is a conditional (*q* ⊃ *r*) then (make-supposition (suppose *A*) {*q*} {*r*}), and where *A** is the resulting supposition:

(1) insert $\langle r,condit,\varnothing,\{\langle r,A^*\rangle\},(p \supset q),A\rangle$ into (interest-queue *A**);
(2) jump to *A** and linear-process it.

Step (2) will be discussed below. We add (c-suppose *p A*) to the definition of (adopt-interest *p A*). Suppositions can then be discharged by making discharge-interest a bit more complex:

(discharge-interest *p A*)

If *p interest*, then for any $\langle p,\alpha,B_0,B,q,S\rangle$ in (forset *A*), delete $\langle p,\alpha,B_0,B,q,S\rangle$ from (forset *A*), and where *B** is the set of all $\langle r,S^*\rangle$ in B_0 such that *r* (adoptions *S**):

(1) if $B^* = \varnothing$ then insert $\langle q,\alpha,B\rangle$ into (adoption-queue *S*), and if $S \neq A$ then jump to *S* and linear-process it;

(2) otherwise, where $\langle r,S^*\rangle$ is the first member of *B**, insert $\langle r,(B^*-\{\langle r,S^*\rangle\}),B,q,S\rangle$ into (interest-queue *S**), and if $S^* \neq A$ then jump to *S* and linear-process it.

Notice that OSCAR now inserts $\langle q,\alpha,B\rangle$ into (adoption-queue *S*) rather than just inserting $\langle q,B\rangle$. This is so that upon adopting *q*, this triple can be put into (basis *A*) to maintain a record of the reasoning. When a proposition *p* is deduced from a set *X* in a supposition *A* in accordance with r-deduce, what will be inserted into (adoption-queue *A*) is $\langle q,con,X\text{x}\{A\}\rangle$.

cancel-interest must be made more complicated than before, because now when OSCAR cancels interest in a proposition, it should also cancel suppositions made for the purpose of obtaining that proposition. If *p* is deleted from (interest *Y*), OSCAR also deletes it from (ultimate *Y*) if it is there. If that empties (ultimate *Y*), OSCAR cancels interest in the whole supposition *Y* (by setting the cancelflag to 1). OSCAR then recursively cancels interest in suppositions adopted in the service of the cancelled supposition.

What remains is to explain the control structure that combines these rules to form OSCAR. OSCAR must now have a more complex architecture than when it involved only linear reasoning. At any given time, we let *HOME* be the supposition within which OSCAR is working. OSCAR begins by setting *HOME* equal to *beliefs*. It then does linear processing within *HOME* (some of which will generate new suppositions), until some condition is satisfied that leads it to move-on to another supposition, which is then named "*HOME*". move-on can be triggered by three sorts of circumstances. First, OSCAR may simply run out of things to do in *HOME*. This happens whenever (interest-queue *HOME*) and (adoption-queue *HOME*) are both emptied. Second, OSCAR may adopt some proposition *p* in (ultimate *HOME*). If *HOME* ≠ *beliefs*, *p* was placed in (ultimate *HOME*) for the purpose of obtaining a conditional in some other supposition, so the reasonable thing for OSCAR to do is jump to that other supposition, making it *HOME*, and adopting the conditional. Third, upon adopting interest in a conditional ($p \supset q$) in *A*, (c-suppose *p A*) will have OSCAR make a supposition *B* in which it supposes the antecedent *p* and adopts ultimate interest in *q*. It turns out that OSCAR is much faster if we have it jump immediately to *B*, renaming it "*HOME*", and process *B* before continuing to process *A*. If OSCAR is unable to get *q* within *B*, then it returns to *A*. This must all be made recursive, because in the process of exploring *B*, OSCAR may jump to another supposition *C*. If OSCAR gets stuck in *C*, it must return to *B*, and then if it is stuck there too, it must return to *A*.

This control structure can be imposed by defining OSCAR to be the following process (where *input* and *interests* are supplied at the start):

(oscar)

(1) let *beliefs* be the supposition resulting from (make-supposition ∅ *interests*), and set (adoption-queue *beliefs*) = *input*;

(2) set *HOME* = *beliefs*;

(3) perform the following loop until instructed otherwise:

(a) (linear-process *HOME*);
(b) (move-on).

To illustrate the functioning of conditionalization, suppose we give OSCAR the input $(p \supset (q \supset r))$ and the ultimate interest $[(p \supset q) \supset (p \supset r)]$. OSCAR will begin by making a supposition A in which it supposes $(p \supset q)$ with ultimate interest $(p \supset r)$, putting $\langle (p \supset r), condit, \varnothing, \{\langle (p \supset r), \text{A}\rangle\}, [(p \supset q) \supset (p \supset r)], beliefs\rangle$ into (forset A). Then because it wants $(p \supset r)$ in A, it will adopt a supposition B in which it supposes $\{(p \supset q), p\}$, with ultimate interest r, putting $\langle r, condit, \varnothing, \{\langle r, B\rangle\}, (p \supset r), A\rangle$ into (forset B). OSCAR will immediately get r in B by r-deduce. Then by discharge-interest, it will get $(p \supset r)$ in A, and then $[(p \supset q) \supset (p \supset r)]$ in *beliefs*.

5. Reductio ad absurdum

The systems of reasoning described in sections three and four illustrate many general features of interest driven suppositional reasoning, but they also suffer from a crucial defect - the deducibility relation is not transitive. Examples to illustrate this will always depend upon precisely what forwards and backwards reasons we give the system, but the general phenomenon is the same in all cases. As we have seen, backwards reasoning is not the same thing as forwards reasoning in reverse, because different reasons are available for each. Accordingly, the following possibility can arise:

$$p \text{ - - - - } > s_1 < \text{ - - - - } q \text{ - - - - } > s_2 < \text{ - - - - } r$$

Here we can deduce q from p by reasoning forwards from p and backwards from q until the two strands connect up at s_1, and we can deduce r from q by reasoning forwards from q and backwards from r until the two strands connect up at s_2. But it may not be possible to deduce r from p by reasoning forwards from p and backwards from q. We may not be able to get beyond s_1 by random forward reasoning from p, or beyond s_2 by backwards reasoning from r. This will leave us with a gap in between that cannot be filled by the system described. This is because it has no way to get interested in q, and hence no way to deduce q from p and then deduce r from q. To illustrate, suppose we use just the reasons listed at the end of section two, and we set the system the task of deducing $(s \vee t)$ from the premises $(p \,\&\, q)$ and $[\sim(q \vee r) \vee s]$. Then we have the following reasoning patterns:

$$(p \,\&\, q) \text{ - - - -> } q \text{ <- - - - } (q \vee r)$$
$$\text{- - - -> } s \text{ <- - - - } (s \vee t)$$
$$[\sim(q \vee r) \vee s]$$

The system has no difficulty deducing $(q \vee r)$ from $(p \,\&\, q)$, and it has no difficulty deducing $(s \vee t)$ from $[\sim(q \vee r) \vee s]$ and $(q \vee r)$, but it has no way to put all this together and deduce $(s \vee t)$ from $(p \,\&\, q)$ and $[\sim(q \vee r) \vee s]$. Of course, any particular example of this sort can always be circumvented by adding reasons to the system, but there seems to be no plausible array of reasons that will overcome this difficulty in general.

In the propositional calculus, I have found two ways to make OSCAR complete and hence ensure that the deducibility relation is transitive. This can be accomplished either by adding a rule of reductio ad absurdum, or by adding a rule for reasoning by cases (dilemma). When we go on to the predicate calculus, a rule of dilemma will no longer be sufficient, but a rule of reductio ad absurdum will still secure transitivity. It is not clear why reductio should be required for transitivity, but I have not found any alternative. Accordingly, I will describe reductio ad absurdum in this section, and give a belief description of dilemma in the next.

Like any kind of suppositional reasoning, reductio ad absurdum involves two basic procedures, one for making suppositions, and one for discharging suppositions. The basic rule for making a supposition is the following:

(reductio-suppose p A)

> Where A is a supposition and p is a proposition in which we have newly adopted interest relative to A, if $\neg p$ is not in either (adoptions A) or (adoption-queue A), let A^* be the supposition resulting from (make-supposition A $\{\neg p\}$ $\{p\}$ A), and:
> (1) insert $\langle p,\textit{reductio},\varnothing,\{\langle p,A^*\rangle\},p,A\rangle$ into (interest-queue A^*);
> (2) set (reductioflag A^*) = 1.

The reductioflag is used to keep track of whether a supposition was introduced for use with conditionalization or for use with reductio ad absurdum. (reductio-suppose p A) is added to the definition of (adopt-interest p A). Inferences by reductio ad absurdum are then accomplished by discharge-interest, without having to change that rule in any way.

There are two kinds of reductio ad absurdum reasoning. In one you establish p by supposing $\neg p$ and deriving p from it. In the other you establish p by supposing $\neg p$ and deriving an arbitrary pair q and $\neg q$ from it. The latter includes the former as a special case. As formulated, reductio-suppose corresponds to the former variety of reductio ad absurdum. It turns out that that is inadequate. In order to make the deducibility relation transitive, we need the stronger variety of reductio ad absurdum. This cannot be implemented by simply instructing the system to adopt interest in every contradiction, because that would require the system to adopt interest in infinitely many propositions. However, we can capture a non-explosive form of the strong reductio rule by instructing the system to adopt interest in the negation of anything it adopts:

(reductio-interest-adoption p A)

> Where p is newly adopted relative to the supposition A, if (reductioflag A) = 1 and $\neg p$ is not already in either (interest A) or (interest-queue A) then for every σ in (forset A), if q,A^* are the last two members of σ then affix $\langle\neg p,\varnothing,\{\langle p,A\rangle,\langle\neg p,A\rangle\},q,A^*\rangle$ to the end of (interest-queue A).

The addition of (reductio-interest-adoption p A) to the definition of (adopt p A) has the result that, within a reductio-supposition, whenever OSCAR adopts anything it adopts-interest in its negation, and if OSCAR is able to adopt that negation then it can discharge the

supposition and infer the proposition whose negation it originally supposed.[6]

In conditional reasoning, when OSCAR adopts interest in a conditional ($p \supset q$), it supposes p and immediately jumps to that supposition and begins exploring it. However, reductio-suppose must proceed differently. Otherwise, every time OSCAR adopted interest in a proposition, it would immediately suppose its negation. If in the course of exploring that reductio-supposition OSCAR adopted interest in something else, it would suppose its negation, and jump to a new reductio-supposition. The result would be an infinite loop in which OSCAR never get back to *beliefs*. Instead, human mathematicians and logicians employ reductio ad absurdum in a more controlled fashion. For the most part, they attempt a reductio only when nothing else seems to work. We can build a control structure into OSCAR that makes it behave similarly.

6. Refinements for the Propositional Calculus

Different arrays of backwards and forwards reasons can be incorporated into OSCAR. The reasons actually used by the running system are not those of section two, but are instead as follows:

Forwards Reasons:

$(p \ \& \ q) \vdash p$	$\sim\sim p \vdash p$
$(p \ \& \ q) \vdash q$	$\sim(p \supset q) \vdash p, \neg q$
$p_1,...,p_n, [(p_1 \ \&...\& \ p_n) \supset q] \vdash q$	$(p \supset q), \neg q \vdash \neg p$
$(p \vee q), \neg p \vdash q$	$(p \supset q), \neg q \vdash p$
$\sim(p \ \& \ q) \vdash (\neg p \vee \neg q)$	$(p \equiv q) \vdash (p \supset q), (q \supset p)$
$\sim(p \vee q) \vdash \neg p$	$\sim(p \equiv q) \vdash \sim[(p \supset q) \ \& \ (q \supset p)]$
$\sim(p \vee q) \vdash \neg q$	

Backwards Reasons:

$p, q \vdash (p \ \& \ q)$	$p \vdash \sim\sim p$
$\sim p, \sim q \vdash \sim(p \vee q)$	$p, \neg q \vdash \sim(p \supset q)$
$(p \supset q), (q \supset p) \vdash (p \equiv q)$	$(\neg p \vee q) \vdash (p \supset q)$
$(\neg q \vee p) \vdash (p \supset q)$	$(\neg p \vee \neg q) \vdash \sim(p \ \& \ q)$

If $(p \supset q)$ is already established: $p, (p \supset q) \vdash q$

These reasons were chosen largely as a result of experiment. They make the system faster than any other combinations tried.

With these additions, OSCAR becomes a powerful system for reasoning in the propositional calculus. OSCAR's purpose is to reason to a conclusion, which is not the

---

[6] The restriction in process-interest-queue against adopting interest in propositions whose negations have been adopted must be relaxed for reductio-suppositions.

same thing as producing a proof of the conclusion. Because it includes backwards reasoning, the structure of interest driven reasoning is more complicated than the structure of a proof. Nevertheless, it is always possible to distill a proof out of OSCAR's reasoning. OSCAR keeps a record of the reasoning in the slots (basis *A*) and it is straightforward to construct an algorithm for producing proofs from that stored information. Adding such a proof module to OSCAR, and building in display capabilities so that we can observe OSCAR's reasoning, the following is a sample run of OSCAR:

input: *P*
interests: ((*P* & *Q*) ∨ (*P* & ~*Q*))

SUPPOSITION: 1
ultimate interests:
((*P* & *Q*) ∨ (*P* & ~*Q*))
adoptions: interests:
P given
((*P* & *Q*) ∨ (*P* & ~*Q*)) ultimate interest
(~(*P* & ~*Q*) ∨ (*P* & *Q*)) for disjunction introduction
jump to 2

MOVING TO SUPPOSITION: 2; origins: 1)
~(*P* & ~*Q*)
ultimate interests:
(*P* & *Q*)
adoptions: interests:
P given
~(*P* & ~*Q*) by supposition
(*P* & *Q*) ultimate interest
Q for adjunction
(~*P* ∨ *Q*) De Morgan
Q disjunctive syllogism
(*P* & *Q*) adjunction
jump back to 1

MOVING TO SUPPOSITION: 1;
ultimate interests:
((*P* & *Q*) (*P* & ~*Q*))
adoptions: interests:
((*P* & *Q*) ∨ (*P* & ~*Q*))
(~(*P* & ~*Q*) ∨ (*P* & *Q*))
P given
(~(*P* & ~*Q*) ∨ (*P* & *Q*)) conditionalization
((*P* & *Q*) ∨ (*P* & ~*Q*)) disjunction introduction

given: P
to prove: $((P \ \& \ Q) \vee (P \ \& \sim Q))$

suppose: $\sim(P \ \& \sim Q)$

(1)	$\sim(P \ \& \sim Q)$	by supposition
(2)	$(\sim P \vee Q)$	De Morgan from 1
(3)	P	given
(4)	Q	disjunctive syllogism from 3 2
(5)	$(P \ \& \ Q)$	adjunction from 3 4

suppose: -

(6)	$(\sim(P \ \& \sim Q) \vee (P \ \& \ Q))$	conditionalization from 5
(7)	$((P \ \& \ Q) \vee (P \ \& \sim Q))$	disjunction introduction from 6

Although I will not go into the details here, it is not difficult to show that OSCAR is complete in the sense that it can, in principle, infer every tautological consequent from a finite set of premises. The proof proceeds by showing that if all else fails, OSCAR can always construct a proof by reductio that looks rather like a proof by resolution refutation. In fact, I have rarely seen OSCAR resort to such an inelegant proof. OSCAR frequently produces proofs which, if constructed by a student in elementary logic, would be considered ingenious.

It is worth noting that OSCAR does not require reasoning by conditionalization to be complete. Any such reasoning can be subsumed under reasoning by reductio ad absurdum. For example, without conditionalization OSCAR would do the above proof by reductio, and the result would be exactly the same as a proof by resolution refutation. This suggests deleting conditionalization from OSCAR. However, without conditionalization, OSCAR is *much* slower, and the proofs OSCAR produces are less natural.

Although OSCAR is complete as described, the running system also incorporates a pair of rules for reasoning by cases (dilemma). Whenever OSCAR adopts a disjunction ($p \vee q$) in a supposition A, dilemma-suppose leads OSCAR to make a new supposition A_1 in which p is supposed separately. A_1 is given the same ultimate interests as A. Whenever one of those ultimate interests r is adopted in A_1, OSCAR makes a new supposition A_2 with ultimate interest $\{r\}$ and in which q is supposed. If r is subsequently obtained in A_2, then discharge-interest instructs the system to adopt it in A as well.

It turns out that reductio-interest-adoption and reductio-suppose can be replaced by dilemma-suppose combined with the forwards reason $(\neg p \vee q) \vdash (p \supset q)$, and c-suppose combined with the (already incorporated) backwards reason $(p \supset q) \vdash (\neg p \vee q)$. Experiment indicates, however, that the result is a significant loss in efficiency. The most efficient use of dilemma involves distinguishing between conditionals and disjunctions (of course, the equivalence is provable, but they are not processed in the same way), and applying dilemma only to the former. In effect, different ways of encoding the same information amount to different instructions regarding how to process it. This is lost in resolution based systems,

because they eliminate all other connectives in favor of 'v' and '~'.[7] Much of OSCAR's efficiency (and correspndingly, the efficiency of human reasoning) seems to result from this encoding of processing information.

7. First-Order Reasoning

The next step is to extend the theory of interest driven reasoning to reasoning with quantifiers. With the exception of Pelletier's THINKER [16] and Oppacher and Suen's HARP ([14] and [15]), all theorem provers with which I am familiar use Skolemization and unification for this purpose, but nothing like Skolemization is involved in human reasoning. Instead, human reasoning involves something like a rule of universal generalization. If we want to establish $(\forall x)(F\ x)$, we "consider an arbitrary object x", reason to the conclusion that it is F, and then infer $(\forall x)(F\ x)$. This involves a new kind of supposition, one whose point is to introduce something that plays the role of a free variable. Because we can go on to use conditionalization or reductio ad absurdum to draw conclusions about the free variable, such suppositions can also involve substantive assumptions. So on this view, a supposition has two parts, a set of formulas (possibly containing free variables) and a set of free variables. This more complex notion of a supposition can be implemented by adding a slot for (variables *A*) in suppositions. make-supposition will now take three arguments, the third being the set of free variables. (make-supposition *A V C*) will be as before, except that it sets (variables *X*) = *V*, and it creates a new supposition if there is none with (suppose *X*) = *A* and (variables *X*) = *V*.

Reasoning with universal generalizations involves the use of a rule that can be stated as follows (where the system has newly adopted interest in *p* relative to *A*):

(u-suppose *p A*)

If *p* is a universal generalization $(\forall x)q$ then:

(1) let $*x$ be a variable not in (variables *A*);
(2) let $q*$ be (subst $*x$ *x q*), i.e., the result of substituting $*x$ for all occurrences of *x* in *q*;[8]
(3) let $A*$ be the supposition resulting from (make-supposition (suppose *A*) (variables *A*) $\{*x\}$ $\{q*\}$);
(4) insert $\langle q*,ug,\varnothing,\{\langle q*,A*\rangle\},p,A\rangle$ into (interest-queue $A*$);
(5) if $q*$ is not already in (interest $A*$), jump to $A*$.

---

[7] Somewhat the same effect is achieved by directed resolution, although that is in general incomplete whereas OSCAR's encoding of processing instructions does not affect completeness.

[8] To avoid collision of variables, OSCAR requires that no well formed formula can contain a quantifier within the scope of another quantifier binding the same variable.

Universal generalization will then proceed by applying discharge-interest to the entries in (forset *A*) that result from u-suppose.

In addition to universal generalization, reasoning with universal quantifiers involves a rule of universal instantiation, allowing us to infer substitution instances of universal generalizations. The question immediately arises as to *which* instances should be inferred from a given generalization. OSCAR cannot infer *all possible* instances - there are infinitely many of them. My initial (provisional) proposal is that we take ui to license the inference from $(\forall x)p$ to (subst $c\ x\ p$) only when c is a term already occurring in the current supposition. An important refinement of this strategy will be discussed below. The most efficient way to handle this is to have a slot in suppositions called (terms *A*), which is a list of all the terms to which OSCAR can instantiate by ui. Then ui can be stated as follows:

(ui *p A*)

If *p* is newly adopted relative to a supposition *A* and has the form $(\forall x)q$, and *c* (terms *A*), then insert $\langle$(subst *c x q*),*ui*,$\{\langle p,A\rangle\}\rangle$ into (adoption-queue *A*).

If we give OSCAR some appropriate reasons, it can reason with existential quantifiers by translating them into universal quantifiers. However, OSCAR is faster and produces more natural looking proofs if we instead give it rules explicitly designed for handling existential quantifiers. The required rules are rules of existential instantiation and existential generalization. Existential generalization proceeds via an interest rule to the effect that when interest is adopted in $(\exists x)p$, OSCAR should also adopt interest in (subst $c\ x\ p$) for each *c* in (terms *A*) and insert $\langle$(subst *c x p*),*eg*,$\{\langle$(subst *c x p*),*A*$\rangle\}$,$\{\langle$(subst *c x p*),*A*$\rangle\}$,$(\exists x)p$,*A*$\rangle$ into (interest-queue *A*).[9]

Existential instantiation instructs OSCAR that if $(\exists x)p$ is adopted and no instance of *p* has yet been adopted, then OSCAR is to choose a new term $@x$ not occurring in (terms *A*), insert $@x$ into (terms *A*), adjust the *eg* entries in (interest-queue *A*) and (forset *A*) appropriately, universally instantiate all universal generalizations in (adoptions *A*) with respect to $@x$, and adopt (subst $@x\ x\ p$). There is a complication, however, concerning when this is to be done. It would be natural to do it immediately upon adopting $(\exists x)p$, but that can lead to an infinite loop. For instance, if OSCAR adopts $(\exists x)(\exists y)(R\ x\ y)$, and $c\in$(terms *A*), this would lead to the adoption of $(\exists y)(R\ c\ y)$, $(R\ c\ @y)$, $(\exists y)(R\ @y\ y)$, $(R\ @y\ @@y)$, $(\exists y)(R\ @@y\ y)$, $(R\ @@y\ @@@y)$, and so on. The strategy employed in OSCAR is to allow one level of existential instantiation immediately, but if an infinite loop threatens then OSCAR postpones further applications of existential instantiation until other rules are applied. The delayed application of ei is handled by the same control structure that determines when to make reductio suppositions. It interleaves the application of ei and reductio-suppose.

OSCAR's array of forwards and backwards reasons are supplemented with the following quantificational reasons:

Forwards Reasons:

---

[9] To guarantee completeness in the predicate calculus, if (terms *A*) is empty, this rule must introduce a new dummy term ``@'' and insert it into (terms *A*).

$\sim(\exists x)p \vdash (\forall x)\neg p$ $\sim(\forall x)p \vdash (\exists x)\neg p$

$((\exists x)p \supset q) \vdash (\forall x)(p \supset q)$ (rewriting any occurrences of `x' in q)

Backwards Reasons:

$(\exists x)\neg p \vdash \sim(\forall x)p$ $(\forall x)\neg p \vdash \sim(\exists x)p$

With these rules for dealing with quantifiers, OSCAR is able to reason to and produce proofs of theorems of the predicate calculus in ways that closely mimic human reasoning. For instance, OSCAR produced the following proof:

given: $(\forall x)(\forall y)((R\,x\,y) \supset (R\,y\,x))$
$(\forall x)(\forall y)(\forall z)(((R\,x\,y) \,\&\, (R\,y\,z)) \supset (R\,x\,z))$
$(\forall x)(\forall y)(R\,x\,y)$
to prove:$(\forall x)(R\,x\,x)$

25 adoption steps; 2 interest adoptions; 2 suppositions explored
13 inference steps used in proof; 2 suppositions used in proof
48% redundant adoptions; 0% redundant suppositions

SUPPOSITION: 1
ultimate interests:
$(\forall x)(R\,x\,x)$
adoptions: interests:
1. $(\forall x)(\forall y)(R\,x\,y)$ given
2. $(\forall x)(\forall y)((R\,x\,y) \supset (R\,y\,x))$ given
3. $(\forall x)(\forall y)(\forall z)(((R\,x\,y) \,\&\, (R\,y\,z)) \supset (R\,x\,z))$ given
    1. $(\forall x)(R\,x\,x)$ ultimate interest

jump to 2

MOVING TO SUPPOSITION: 2; sup-type U; origins: 1
nil
{ *x }
ultimate interests:
$(R\,{*x}\,{*x})$
adoptions: interests:
\+ + + +
1. $(\forall x)(\forall y)(R\,x\,y)$ given
2. $(\forall x)(\forall y)((R\,x\,y) \supset (R\,y\,x))$ given
3. $(\forall x)(\forall y)(\forall z)(((R\,x\,y) \,\&\, (R\,y\,z)) \supset (R\,x\,z))$ given
    1. $(R\,{*x}\,{*x})$ ultimate interest
4. $(\forall y)(R\,{*x}\,y)$ UI from 1
5. $(R\,{*x}\,@y)$ EI from 4
6. $(\forall y)(R\,@y\,y)$ UI from 1
7. $((R\,@y\,{*x}) \supset (R\,{*x}\,@y))$ UI from 2

8. $((R\ @y\ @y) \supset (R\ @y\ @y))$ UI from 2
9. $(\forall y)((R\ *x\ y) \supset (R\ y\ *x))$ UI from 2
10. $((R\ *x\ *x) \supset (R\ *x\ *x))$ UI from 9
11. $((R\ *x\ @y) \supset (R\ @y\ *x))$ UI from 9
12. $(R\ @y\ *x)$ modus ponens from 11 5
13. $(((R\ @y\ *x)\ \&\ (R\ *x\ *x)) \supset (R\ @y\ *x))$ UI from 3
14. $(((R\ @y\ *x)\ \&\ (R\ *x\ @y)) \supset (R\ @y\ @y))$ UI from 3
15. $(R\ @y\ @y)$ modus ponens from 14 5 12
16. $(((R\ @y\ @y)\ \&\ (R\ @y\ *x)) \supset (R\ @y\ *x))$ UI from 3
17. $(((R\ @y\ @y)\ \&\ (R\ @y\ @y)) \supset (R\ @y\ @y))$ UI from 3
18. $(\forall y)(\forall z)(((R\ *x\ y)\ \&\ (R\ y\ z)) \supset (R\ *x\ z))$ UI from 3
19. $(\forall z)(((R\ *x\ *x)\ \&\ (R\ *x\ z)) \supset (R\ *x\ z))$ UI from 18
20. $(((R\ *x\ *x)\ \&\ (R\ *x\ *x)) \supset (R\ *x\ *x))$ UI from 19
21. $(((R\ *x\ *x)\ \&\ (R\ *x\ @y)) \supset (R\ *x\ @y))$ UI from 19
22. $(\forall z)(((R\ *x\ @y)\ \&\ (R\ @y\ z)) \supset (R\ *x\ z))$ UI from 18
23. $(((R\ *x\ @y)\ \&\ (R\ @y\ *x)) \supset (R\ *x\ *x))$ UI from 22
24. $(R\ *x\ *x)$ modus ponens from 23 12 5

jump back to 1

MOVING TO SUPPOSITION: 1; sup-type NIL; origins:
(nil nil)
ultimate interests:
$(\forall x)(R\ x\ x)$
adoptions: interests:
\+ + + +
4. $(\forall x)(R\ x\ x)$ UG from supposition 2

THEOREM
given: $(\forall x)(\forall y)(R\ x\ y)$
$(\forall x)(\forall y)((R\ x\ y) \supset (R\ y\ x))$
$(\forall x)(\forall y)(\forall z)(((R\ x\ y)\ \&\ (R\ y\ z)) \supset (R\ x\ z))$
to prove: $(\forall x)(R\ x\ x)$

suppose, with variables { $*x$ }
-
(1) $(\forall x)(\forall y)(R\ x\ y)$ given
(2) $(\forall x)(\forall y)((R\ x\ y) \supset (R\ y\ x))$ given
(3) $(\forall x)(\forall y)(\forall z)(((R\ x\ y)\ \&\ (R\ y\ z)) \supset (R\ x\ z))$ given
(4) $(\forall y)(R\ *x\ y)$ UI from 1
(5) $(\forall y)((R\ *x\ y) \supset (R\ y\ *x))$ UI from 2
(6) $(\forall y)(\forall z)(((R\ *x\ y)\ \&\ (R\ y\ z)) \supset (R\ *x\ z))$ UI from 3
(7) $(R\ *x\ @y)$ EI from 4
(8) $((R\ *x\ @y) \supset (R\ @y\ *x))$ UI from 5
(9) $(\forall z)(((R\ *x\ @y)\ \&\ (R\ @y\ z)) \supset (R\ *x\ z))$ UI from 6

(10) $(R\ @y\ *x)$ modus ponens from 8 7
(11) $(((R\ *x\ @y)\ \&\ (R\ @y\ *x)) \supset (R\ *x\ *x))$ UI from 9
(12) $(R\ *x\ *x)$ modus ponens from 11 10 7

suppose, with variables {}
-
(13) $(\forall x)(R\ x\ x)$ UG from 12

Because of the presence of ei and ui, OSCAR can systematically generate the entire Herbrand universe (in the form of the variables $*x$ and $@x$ rather than by using Skolem functions), so it follows from Herbrand's theorem that OSCAR is complete for the predicate calculus.

I have described OSCAR as not using Skolemization or unification, but ei and ui accomplish the same thing. The difference is that they do this without introducing Skolem functions. My original motivation for this was to build a system that more closely mimics human reasoning. One could, however, try to build a hybrid system by combining the system of suppositional reasoning in the propositional calculus with Skolemization and unification. But it is not obvious how to do that. Given Skolemized formulas, unification is not an inference rule - it is a procedure that is incorporated into truth-functional inference rules to generate first-order rules. Unification could be combined with each of OSCAR's truth-functional inference rules in the same way it is combined with resolution. However, the control structure would become problematic. I indicated above the necessity to interleave applications of ei and reductio. However, incorporating unification into the other rules has the effect of doing ei automatically, without waiting for appications of reductio. It is not clear how to solve this problem.

8. All-Detachment

In accordance with ui, if OSCAR adopts $(\forall x)(F\ x)$ relative to a supposition A, it goes on to adopt $(F\ c)$ for every c in (terms A). If a large number of constants occur in the problem, most such instantiations will normally be useless, and the processing of these instances may retard the performance of the system by a large margin. This is connected with the way "all-statements" are represented in the predicate calculus. We symbolize "All A are B" as $(\forall x)((A\ x) \supset (B\ x))$. The system described in section seven then applies ui to the latter universal generalization and infers all conditionals of the form $((A\ c) \supset (B\ c))$. But this can result in the adoption of an immense number of useless conditionals. Human reasoning is more circumspect. Instead of inferring the conditional $((A\ c) \supset (B\ c))$, we normally wait until either (1) we have inferred $(A\ c)$, in which case we infer $(B\ c)$ directly, or (2) we have inferred $\sim(B\ c)$, in which case we infer $\sim(A\ c)$ directly. These inferences illustrate the rule of all-detachment, which, in its simplest form, licenses the following inferences:

(All A are B), $(A\ c) \vdash (B\ c)$
(All A are B), $\sim(B\ c) \vdash \sim(A\ c)$.

More complicated cases of all-detachment occur when A is a conjunction, for instance:

(All C&D are B), (C c), ~(B c) $\vdash$ ~(D c).

all-detachment can also involve the simultaneous instantiation of several initial universal quantifiers. Note the similarity of all-detachment to unit resolution.

all-detachment is implemented by having a rule which converts newly adopted universal generalizations into all-statements with conjunctive antecedents whenever that is possible, and stores them in a new slot. Then whenever the system adopts appropriate instances of parts of an all-statement, it makes the appropriate inferences. More precisely, if OSCAR adopts $(\forall x_1)...(\forall x_n)p$, where p is not a universal generalization, then it converts $\sim p$ into a kind of disjunctive normal form by recursively applying the following transformations from the outside in:

$[p \,\&\, (q \vee r)]$ $\quad$ $[(p \,\&\, q) \vee (p \,\&\, r)]$
$[(p \vee q) \,\&\, r]$ $\quad$ $[(p \,\&\, r) \vee (q \,\&\, r)]$
$(p \supset q)$ $\quad$ $(\neg p \vee q)$
$(p \equiv q)$ $\quad$ $[(p \,\&\, q) \vee (\neg p \,\&\, \neg q)]$
$(\exists x)(p \vee q)$ $\quad$ $((\exists x)p \vee (\exists x)q)$ (vacuous quantifiers are dropped)
$\sim\sim p$ $\quad$ p
$\sim(p \vee q)$ $\quad$ $(\neg p \,\&\, \neg q)$
$\sim(p \,\&\, q)$ $\quad$ $(\neg p \vee \neg q)$
$\sim(p \supset q)$ $\quad$ $(p \,\&\, \neg q)$
$\sim(p \equiv q)$ $\quad$ $[(p \,\&\, \neg q)$ $(\neg p \,\&\, q)]$
$\sim(\forall x)p$ $\quad$ $(\exists x)\neg p$

Note that embedded unnegated universal generalizations are not transformed, so the q_i's need not be literals. As $\sim p$ has no true instances, each of the resulting disjuncts has all false instances. If $(q_1 \,\&...\&\, q_m)$ is such a disjunct, we store $\langle\{q_1,...,q_n\},\{x_1,...,x_n\},p\rangle$ in (all-schemes A). For instance, adoption of

$(\forall x)\{(A\ x) \supset ((\forall w)[(P\ w) \supset (E\ x\ w)] \supset (\forall y)[\{(A\ y) \,\&\, ((M\ y\ x) \,\&\, (\exists z)[(P\ z) \,\&\, (E\ y\ z)]\} \supset (E\ x\ y)])\}$

(from the Schubert steamroller problem - see below) results in the following list being inserted into (all-schemes A):

$\langle\{(A\ x), \sim(E\ x\ w), (P\ w), (E\ y\ z), (P\ z), (M\ y\ x), (A\ y), \sim(E\ x\ y)\},$
$\{x,y,z,w\},$
$(\forall x)\{(A\ x) \supset ((\forall w)[(P\ w) \supset (E\ x\ w)] \supset (\forall y)[\{(A\ y) \,\&\, ((M\ y\ x)$
$\quad \,\&\, (\exists z)[(P\ z) \,\&\, (E\ y\ z)]\} \supset (E\ x\ y)])\}\rangle$

all-detachment then instructs OSCAR to adopt an instance of $\neg q_i$ whenever it adopts the corresponding instances of $q_1,...,q_{i-1},q_{i+1},...,q_m$. We cannot *replace* ui by all-detachment,

because the resulting system will be incomplete, but what we can do instead is apply all-detachment first and delay the application of ui until we have exhausted all other possible inferences within a supposition. This makes the system much more efficient.

A similar strategy is adopted in connection with eg. Given interest in $(\exists x)(F\ x)$, instead of immediately adopting interest in every instance $(F\ c)$ for c (terms A), OSCAR simply begins checking each new adoption to see whether its addition to (adoptions A) creates an instance of $(\exists x)(F\ x)$. Only when OSCAR exhausts both (interest-queue A) and (adoption-queue A) does it execute eg.

These changes make OSCAR a great deal more efficient. To illustrate, the example given in the last section is now shortened as follows:

given: $(\forall x)(\forall y)((R\ x\ y)\ \ (R\ y\ x))$
$(\forall x)(\forall y)(\forall z)(((R\ x\ y)\ \&\ (R\ y\ z))\supset(R\ x\ z))$
$(\forall x)(\forall y)(R\ x\ y)$
to prove:$(\forall x)(R\ x\ x)$

8 adoption steps; 2 interest adoptions; 2 suppositions explored
8 inference steps used in proof; 2 suppositions used in proof
0% redundant adoptions; 0% redundant suppositions

SUPPOSITION: 1
ultimate interests:
$(\forall x)(R\ x\ x)$
adoptions: interests:
1. $(\forall x)(\forall y)(R\ x\ y)$ given
2. $(\forall x)(\forall y)((R\ x\ y)\supset(R\ y\ x))$ given
3. $(\forall x)(\forall y)(\forall z)(((R\ x\ y)\ \&\ (R\ y\ z))\supset(R\ x\ z))$ given
1. $(\forall x)(R\ x\ x)$ ultimate interest

jump to 2

MOVING TO SUPPOSITION: 2; sup-type U; origins: 1
nil
{ *x }
ultimate interests:
$(R\ *x\ *x)$
adoptions: interests:
+ + + +
1. $(\forall x)(\forall y)(R\ x\ y)$ given
2. $(\forall x)(\forall y)((R\ x\ y)\supset(R\ y\ x))$ given
3. $(\forall x)(\forall y)(\forall z)(((R\ x\ y)\ \&\ (R\ y\ z))\supset(R\ x\ z))$ given
1. $(R\ *x\ *x)$ ultimate interest
4. $(\forall y)(R\ *x\ y)$ UI from 1
5. $(R\ *x\ @y)$ EI from 4

6. (*R* @*y* **x*) all-detachment from 5 2
7. (*R* **x* **x*) all-detachment from 5 6 3
jump back to 1

MOVING TO SUPPOSITION: 1; sup-type NIL; origins:
(nil nil)
ultimate interests:
(∀*x*)(*R x x*)
adoptions: interests:
+ + + +
4. (∀*x*)(*R x x*) UG from supposition 2

THEOREM
given: (∀*x*)(∀*y*)(*R x y*)
(∀*x*)(∀*y*)((*R x y*) ⊃ (*R y x*))
(∀*x*)(∀*y*)(∀*z*)(((*R x y*) & (*R y z*)) ⊃ (*R x z*))
to prove: (∀*x*)(*R x x*)

suppose, with variables { **x* }
-
(1) (∀*x*)(∀*y*)(*R x y*) given
(2) (∀*y*)(*R* **x y*) UI from 1
(3) (∀*x*)(∀*y*)((*R x y*) ⊃ (*R y x*)) given
(4) (*R* **x* @*y*) EI from 2
(5) (∀*x*)(∀*y*)(∀*z*)(((*R x y*) & (*R y z*)) ⊃ (*R x z*)) given
(6) (*R* @*y* **x*) all-detachment from 4 3
(7) (*R* **x* **x*) all-detachment from 4 6 5

suppose, with variables {}
-
(8) (∀*x*)(*R x x*) UG from 7

Notice particularly the difference in the number of unnecessary inference steps. Using ui, 48% of the inference steps were unnecessary for the proof that was eventually found, but using all-detachment there were *no* unnecessary steps.

9. Conclusions

OSCAR implements interest driven suppositional reasoning in the predicate calculus. As my purpose has been to model human reasoning at least to a first approximation, I have avoided resolution and Skolemization on the grounds that they are not fundamental to human reasoning. They are at best technical tricks that humans can learn to use only with difficulty. Furthermore, the structure of standard resolution theorem provers precludes

their being easily integrated into my system of defeasible reasoning [19] in a way that would throw any light on the interest driven features of the latter, and that was the main purpose of building the present system. OSCAR is a natural deduction system, but it contrasts sharply with most automated theorem provers that are called "natural deduction systems". I am familiar with only one system to which OSCAR bears a strong resemblance. That is Pelletier's THINKER ([16], [17]). Other systems to which OSCAR bears a more distant resemblance are HARP (Oppacher and Suen [14] and [15]), TPS (Andrews et al [1], [2]), and the UT Natural Deduction Prover (Bledsoe [3] and [4], Bledsoe and Tyson [5]), and the systems of Murray [12] and Nevins [13].

My expectation was that OSCAR would be appreciably slower than more familiar automated theorem provers that make use of the special abilities of computers to carry out tasks that humans find difficult. Thus I have been surprised to find that OSCAR is rather fast. It is unfortunate that there is no general standard of comparison for different theorem provers, but I have several bits of evidence indicating that OSCAR is surprisingly fast.

First, I have tested OSCAR on (a corrected version of) Pelletier's problems [18]. OSCAR is written in COMMON LISP, and running on a Symbolics 3600, OSCAR takes only a few seconds (on the average) to do each of Pelletier's propositional and predicate calculus problems. Many of these are too easy to be indicative of much, but one of the harder problems is the Schubert steamroller problem:

$(\forall x)(Wx \supset Ax)$
$(\forall x)(Fx \supset Ax)$
$(\forall x)(Bx \supset Ax)$
$(\forall x)(Cx \supset Ax)$
$(\forall x)(Sx \supset Ax)$
$(\exists w)Ww$
$(\exists f)Ff$
$(\exists b)Bb$
$(\exists c)Cc$
$(\exists s)Ss$
$(\exists g)Gg$
$(\forall x)[Ax \supset [(\forall w)(Pw \supset Exw) \supset$
$(\forall y)((Ay \ \& \ (Myx \ \& \ (\exists z)(Pz \ \& \ Eyz))) \supset Exy)]]$

$(\forall x)(\forall y)[(Cx \ \& \ By) \supset Mxy]$
$(\forall x)(\forall y)[(Sx \ \& \ By) \supset Mxy]$
$(\forall x)(\forall y)[(Bx \ \& \ Fy) \supset Mxy]$
$(\forall x)(\forall y)[(Fx \ \& \ Wy) \supset Mxy]$
$(\forall x)(\forall y)[(Wx \ \& \ Fy) \supset \sim Exy]$
$(\forall x)(\forall y)[(Wx \ \& \ Gy) \supset \sim Exy]$
$(\forall x)(\forall y)[(Bx \ \& \ Cy) \supset Exy]$
$(\forall x)(\forall y)[(Bx \ \& \ Sy) \supset \sim Exy]$
$(\forall x)[Cx \supset (\exists y)(Py \ \& \ Exy)]$
$(\forall x)[Sx \supset (\exists y)(Py \ \& \ Exy)]$
$(\forall x)(Gx \supset Px)$

$(\exists x)(\exists y)[(Ax \ \& \ Ay) \ \& \ (\exists z)[Exy \ \& \ (Gz \ \& \ Eyz)]]$

This is a slightly whimsical symbolization of the following:

> Wolves, foxes, birds, caterpillars, and snails are animals, and there are some of each of them. Also, there are some grains, and grains are plants. Every animal either likes to eat all plants or all animals much smaller than itself that like to eat some plants. Caterpillars and snails are much smaller than birds, which are much smaller than foxes, which in turn are much smaller than wolves. Wolves do not like to eat foxes or grains, while birds like to eat caterpillars but not snails. Caterpillars and snails like to

eat some plants. Therefore, there is an animal that likes to eat a grain-eating animal.[10]

This problem has been attempted by a wide variety of theorem provers, and the fastest time reported is 6 seconds. That time is reported by Stickel [23], running a connection-graph resolution theorem proving program on a Symbolics 3600 and applying it to the result of already transforming the problem into clausal form. In comparison, OSCAR does this problem in 13 seconds on a Symbolics 3600.

OSCAR was not constructed with speed in mind, and is not even written very efficiently. A sustained effort at optimization would accelerate the system significantly. This makes it interesting to compare OSCAR with respected resolution theorem provers that are considered fast but do not do the Schubert steamroller problem as fast as Stickel's current system. An earlier theorem prover of Stickel was reported to do the problem in 2 hours 53 minutes, and the well known ITP theorem prover was at one time reported to do it in 11 minutes (Cohn [7]).

Most theorem provers are designed to be used somewhat interactively, allowing the human operator to choose different proof strategies for different problems. That is desirable if they are to be used as tools, but it detracts from the theoretical significance of speed results. After all, given any valid formula *P*, we could introduce a rule saying "Infer *P*". Of course, no one would actually incorporate a rule like this into a theorem prover, but this just represents the extreme end of a continuum. Fast times using strategies chosen especially for a particular problem may show little unless some theoretical rationale can be provided for the choice of strategy and the rationale built into the theorem prover itself so that it can choose its own strategies. In this connection it should be noted that OSCAR is a purely automatic theorem prover, using the same strategies on all problems. In contrast, Stickel's fast times are extremely impressive, but as he himself observes, they are very sensitive how the set of support is chosen in his theorem prover. He first tried putting only the goal clause in the set of support, but that resulted in his theorem prover taking 5,694 seconds. The 6 seconds time was achieved by eliminating the set of support restriction.

It is of interest to see exactly what OSCAR does on the steamroller problem. The following is a printout of OSCAR's performance, with the steps not used in the proof marked with `#':

53 adoption steps; 1 interest adoptions; 1 suppositions explored
42 inference steps used in proof; 1 suppositions used in proof
21% redundant adoptions; 0% redundant suppositions

SUPPOSITION: 1
ultimate interests:
$(\exists x)(\exists y)(((A\ x)\ \&\ (A\ y))\ \&\ (\exists z)((E\ x\ y)\ \&\ ((G\ z)\ \&\ (E\ y\ z))))$
adoptions: interests:
1. $(\exists w_0)(W\ w_0)$ given
2. $(W\ @w_0)$ EI from 1

[10] Pelletier [18], 203.

3. $(\exists f_0)(F\ f_0)$ given
4. $(F\ @f_0)$ EI from 3
5. $(\exists b_0)(B\ b_0)$ given
6. $(B\ @b_0)$ EI from 5
7. $(\exists c_0)(C\ c_0)$ given
8. $(C\ @c_0)$ EI from 7
9. $(\exists s_0)(S\ s_0)$ given
10. $(S\ @s_0)$ EI from 9
11. $(\exists g_0)(G\ g_0)$ given
12. $(G\ @g_0)$ EI from 11
13. $(\forall x)((W\ x) \supset (A\ x))$ given
14. $(\forall x)((F\ x) \supset (A\ x))$ given
15. $(\forall x)((B\ x) \supset (A\ x))$ given
16. $(\forall x)((C\ x) \supset (A\ x))$ given
17. $(\forall x)((S\ x) \supset (A\ x))$ given
18. $(\forall x)((G\ x) \supset (P\ x))$ given
19. $(\forall x)(\forall y)(((C\ x)\ \&\ (B\ y)) \supset (M\ x\ y))$ given
20. $(\forall x)(\forall y)(((S\ x)\ \&\ (B\ y)) \supset (M\ x\ y))$ given
21. $(\forall x)(\forall y)(((B\ x)\ \&\ (F\ y)) \supset (M\ x\ y))$ given
22. $(\forall x)(\forall y)(((F\ x)\ \&\ (W\ y)) \supset (M\ x\ y))$ given
23. $(\forall x)(\forall y)(((B\ x)\ \&\ (C\ y)) \supset (E\ x\ y))$ given
24. $(\forall x)((C\ x) \supset (\forall y)((P\ y)\ \&\ (E\ x\ y)))$ given
25. $(\forall x)((S\ x) \supset (\forall y)((P\ y)\ \&\ (E\ x\ y)))$ given
26. $(\forall x)(\forall y)(((W\ x)\ \&\ (F\ y)) \supset \sim(E\ x\ y))$ given
27. $(\forall x)(\forall y)(((W\ x)\ \&\ (G\ y)) \supset \sim(E\ x\ y))$ given
28. $(\forall x)(\forall y)(((B\ x)\ \&\ (S\ y)) \supset \sim(E\ x\ y))$ given
29. $(\forall x)((A\ x) \supset ((\forall W)((P\ W) \supset (E\ x\ W)) \supset (\forall y)(((A\ y)$
$\&\ ((M\ y\ x)\ \&\ (\exists z)((P\ z)\ \&\ (E\ y\ z)))) \supset (E\ x\ y))))$ given
    1. $(\exists x)(\exists y)(((A\ x)\ \&\ (A\ y))\ \&\ (\exists z)((E\ x\ y)\ \&\ ((G\ z)\ \&\ (E\ y\ z))))$ ultimate interest
30. $(P\ @g_0)$ all-detachment from 12 18
31. $(A\ @s_0)$ all-detachment from 10 17
32. $(A\ @c_0)$ all-detachment from 8 16
33. $(A\ @b_0)$ all-detachment from 6 15
34. $(A\ @f_0)$ all-detachment from 4 14
35. $(A\ @w_0)$ all-detachment from 2 13
36. $(E\ @b_0\ @c_0)$ all-detachment from 6 8 23
37. $(M\ @f_0\ @w_0)$ all-detachment from 2 4 22
38. $(M\ @b_0\ @f_0)$ all-detachment from 4 6 21
39. $(M\ @s_0\ @b_0)$ all-detachment from 6 10 20
40. $(M\ @c_0\ @b_0)$ all-detachment from 6 8 19
41. $\sim(E\ @b_0\ @s_0)$ all-detachment from 6 10 28
42. $\sim(E\ @w_0\ @g_0)$ all-detachment from 2 12 27
43. $\sim(E\ @w_0\ @f_0)$ all-detachment from 2 4 26

44. $\sim(E\ @f_0\ @g_0)$ all-detachment from 42 30 37 34 30 35 43 29
45. $(\exists y)((P\ y)\ \&\ (E\ @s_0\ y))$ all-detachment from 10 25
46. $((P\ @y)\ \&\ (E\ @s_0\ @y))$ EI from 45
47. $(P\ @y)$ simplification from 46
48. $(E\ @s_0\ @y)$ simplification from 46
49. $\sim(P\ @s_0)$ all-detachment from 33 47 39 31 41 41 48 29
50. $\sim(G\ @s_0)$ all-detachment from 49 18
51. $(E\ @b_0\ @g_0)$ all-detachment from 33 47 39 31 41 30 48 29
52. $(E\ @f_0\ @b_0)$ all-detachment from 34 30 38 33 44 30 51 29
53. $(\exists x)(\exists y)(((A\ x)\ \&\ (A\ y))\ \&\ (\exists z)((E\ x\ y)\ \&\ ((G\ z)\ \&\ (E\ y\ z))))$ EG from 52 33 34 51 12

THEOREM
given: $(\forall x)((S\ x) \supset (\exists y)((P\ y)\ \&\ (E\ x\ y)))$
$(\exists s_0)(S\ s_0)$
$(\exists b_0)(B\ b_0)$
$(\exists f_0)(F\ f_0)$
$(\exists w_0)(W\ w_0)$
$(\forall x)((G\ x) \supset (P\ x))$
$(\forall x)(\forall y)(((B\ x)\ \&\ (S\ y)) \supset \sim(E\ x\ y))$
$(\forall x)((S\ x) \supset (A\ x))$
$(\forall x)(\forall y)(((S\ x)\ \&\ (B\ y)) \supset (M\ x\ y))$
$(\forall x)(\forall y)(((W\ x)\ \&\ (F\ y)) \supset \sim(E\ x\ y))$
$(\forall x)((W\ x) \supset (A\ x))$
$(\forall x)(\forall y)(((F\ x)\ \&\ (W\ y)) \supset (M\ x\ y))$
$(\forall x)(\forall y)(((W\ x)\ \&\ (G\ y)) \supset \sim(E\ x\ y))$
$(\forall x)((B\ x) \supset (A\ x))$
$(\forall x)(\forall y)(((B\ x)\ \&\ (F\ y)) \supset (M\ x\ y))$
$(\forall x)((F\ x) \supset (A\ x))$
$(\exists g_0)(G\ g_0)$
$(\exists x)((A\ x)$
$((\forall W)((P\ W) \supset (E\ x\ W)) \supset (\forall y)(((A\ y)\ \&\ ((M\ y\ x)\ \&\ (\exists z)((P\ z)\ \&\ (E\ y\ z))))$
$\supset (E\ x\ y))))$
to prove: $(\exists x)(\exists y)(((A\ x)\ \&\ (A\ y))\ \&\ (\exists z)((E\ x\ y)\ \&\ ((G\ z)\ \&\ (E\ y\ z))))$

suppose, with variables {}
-
(1) $(\exists s_0)(S\ s_0)$ given
(2) $(\forall x)((S\ x) \supset (\exists y)((P\ y)\ \&\ (E\ x\ y)))$ given
(3) $(S\ @s_0)$ EI from 1
(4) $(\exists y)((P\ y)\ \&\ (E\ @s_0\ y))$ all-detachment from 3 2
(5) $(\exists g_0)(G\ g_0)$ given
(6) $(\exists b_0)(B\ b_0)$ given

(7) $(\exists f_0)(F\ f_0)$ given
(8) $(\exists w_0)(W\ w_0)$ given
(9) $((P\ @y)\ \&\ (E\ @s_0\ @y))$ EI from 4
(10) $(\forall x)((G\ x) \supset (P\ x))$ given
(11) $(G\ @g_0)$ EI from 5
(12) $(\forall x)(\forall y)(((B\ x)\ \&\ (S\ y)) \supset \sim(E\ x\ y))$ given
(13) $(B\ @b_0)$ EI from 6
(14) $(\forall x)((S\ x) \supset (A\ x))$ given
(15) $(\forall x)(\forall y)(((S\ x)\ \&\ (B\ y)) \supset (M\ x\ y))$ given
(16) $(\forall x)((B\ x) \supset (A\ x))$ given
(17) $(\forall x)(\forall y)(((W\ x)\ \&\ (F\ y)) \supset \sim(E\ x\ y))$ given
(18) $(F\ @f_0)$ EI from 7
(19) $(W\ @w_0)$ EI from 8
(20) $(\forall x)((W\ x) \supset (A\ x))$ given
(21) $(\forall x)((F\ x) \supset (A\ x))$ given
(22) $(\forall x)(\forall y)(((F\ x)\ \&\ (W\ y)) \supset (M\ x\ y))$ given
(23) $(\forall x)(\forall y)(((W\ x)\ \&\ (G\ y)) \supset \sim(E\ x\ y))$ given
(24) $(\forall x)((A\ x) \supset$
$((\forall W)((P\ W) \supset (E\ x\ W)) \supset (\forall y)(((A\ y)\ \&\ ((M\ y\ x)\ \&\ (\exists z)((P\ z)\ \&\ (E\ y\ z)))) \supset (E\ x\ y))))$ given
(25) $(E\ @s_0\ @y)$ simplification from 9
(26) $(P\ @g_0)$ all-detachment from 11 10
(27) $\sim(E\ @b_0\ @s_0)$ all-detachment from 13 3 12
(28) $(A\ @s_0)$ all-detachment from 3 14
(29) $(M\ @s_0\ @b_0)$ all-detachment from 13 3 15
(30) $(P\ @y)$ simplification from 9
(31) $(A\ @b_0)$ all-detachment from 13 16
(32) $\sim(E\ @w_0\ @f_0)$ all-detachment from 19 18 17
(33) $(A\ @w_0)$ all-detachment from 19 20
(34) $(A\ @f_0)$ all-detachment from 18 21
(35) $(M\ @f_0\ @w_0)$ all-detachment from 19 18 22
(36) $\sim(E\ @w_0\ @g_0)$ all-detachment from 19 11 23
(37) $(\forall x)(\forall y)(((B\ x)\ \&\ (F\ y)) \supset (M\ x\ y))$ given
(38) $(E\ @b_0\ @g_0)$ all-detachment from 31 30 29 28 27 26 25 24
(39) $\sim(E\ @f_0\ @g_0)$ all-detachment from 36 26 35 34 26 33 32 24
(40) $(M\ @b_0\ @f_0)$ all-detachment from 33 36 37
(41) $(E\ @f_0\ @b_0)$ all-detachment from 34 26 40 31 39 26 38 24
(42) $(\exists x)(\exists y)(((A\ x)\ \&\ (A\ y))\ \&\ (\exists z)((E\ x\ y)\ \&\ ((G\ z)\ \&\ (E\ y\ z))))$ EG from 41 31 34 38 11

Of the 11 unnecessary steps, 6 of them are among the given premises of the problem. Omitting those, OSCAR performs only 5 unnecessary inferences. By contrast, even on Stickel's fastest time his theorem prover performed 479 inferences, and when his theorem

prover was used non-interactively, it performed 245,820 inferences. Stickel also provides data on some other theorem provers doing the Schubert steamroller problem, according to which they performed between 1,106 and 26,190 inference steps. This means that between 91% and 99.98% of the inferences performed by these theorem provers were unnecessary, as opposed to a real measure of 11% for OSCAR.

To take another example, a problem of moderate difficulty that has been run on a number of theorem provers is that of showing that a group is commutative if the square of every element is the identity element. This problem is from Change and Lee [7]. Letting `$(P\ x\ y\ z)$' symbolize `$x \cdot y = z$', this problem can be formulated as follows:

given:$(\forall x)(P\ x\ e\ x)$
$(\forall x)(P\ e\ x\ x)$
$(\forall x)(\forall y)(\forall z)(\forall u)(\forall v)(\forall w)[((P\ x\ y\ u)\ \&\ ((P\ y\ z\ v)\ \&\ (P\ u\ z\ w)))\supset(P\ x\ v\ w)]$
$(\forall x)(\forall y)(\forall z)(\forall u)(\forall v)(\forall w)[((P\ x\ y\ u)\ \&\ ((P\ y\ z\ v)\ \&\ (P\ x\ v\ w)))\supset(P\ u\ z\ w)]$
$(\forall x)(P\ x\ x\ e)$
$(P\ a\ b\ c)$
To prove: $(P\ b\ a\ c)$

19 adoption steps; 1 interest adoptions; 1 suppositions explored
15 inference steps used in proof; 1 suppositions used in proof
21% redundant adoptions; 0% redundant suppositions

SUPPOSITION: 1
ultimate interests:
$(P\ b\ a\ c)$

adoptions:	interests:	
1.	$(P\ a\ b\ c)$	given
2.	$(\forall x)(P\ x\ e\ x)$	given
3.	$(\forall x)(P\ e\ x\ x)$	given
4.	$(\forall x)(P\ x\ x\ e)$	given
5.	$(\forall x)(\forall y)(\forall z)(\forall u)(\forall v)(\forall w)(((P\ x\ y\ u)\ \&\ ((P\ y\ z\ v)\ \&\ (P\ u\ z\ w)))\supset(P\ x\ v\ w))$	given
6.	$(\forall x)(\forall y)(\forall z)(\forall u)(\forall v)(\forall w)(((P\ x\ y\ u)\ \&\ ((P\ y\ z\ v)\ \&\ (P\ x\ v\ w)))\supset(P\ u\ z\ w))$	given
	1. $(P\ b\ a\ c)$	ultimate interest
# 7.	$(P\ e\ e\ e)$	UI from 2
8.	$(P\ a\ a\ e)$	UI from 4
9.	$(P\ b\ b\ e)$	UI from 4
10.	$(P\ c\ c\ e)$	UI from 4
# 11.	$(P\ e\ a\ a)$	UI from 3
# 12.	$(P\ e\ c\ c)$	UI from 3
# 13.	$(P\ c\ e\ c)$	UI from 2
14.	$(P\ a\ e\ a)$	UI from 2
15.	$(P\ c\ b\ a)$	all-detachment from 9 1 14 6
16.	$(P\ e\ b\ b)$	UI from 3
17.	$(P\ a\ c\ b)$	all-detachment from 1 8 16 5

18. $(P\ b\ c\ a)$ all-detachment from 14 10 17 6
19. $(P\ b\ a\ c)$ all-detachment from 1 15 18 5

THEOREM
given: $(\forall x)(P\ e\ x\ x)$
$(\forall x)(P\ x\ e\ x)$
$(\forall x)(P\ x\ x\ e)$
$(\forall x)(\forall y)(\forall z)(\forall u)(\forall v)(\forall w)(((P\ x\ y\ u)\ \&\ ((P\ y\ z\ v)\ \&\ (P\ x\ v\ w)))\supset(P\ u\ z\ w))$
$(\forall x)(\forall y)(\forall z)(\forall u)(\forall v)(\forall w)(((P\ x\ y\ u)\ \&\ ((P\ y\ z\ v)\ \&\ (P\ u\ z\ w)))\supset(P\ x\ v\ w))$
$(P\ a\ b\ c)$
to prove: $(P\ b\ a\ c)$

suppose, with variables {}
-
(1) $(\forall x)(P\ e\ x\ x)$ given
(2) $(\forall x)(P\ x\ x\ e)$ given
(3) $(\forall x)(\forall y)(\forall z)(\forall u)(\forall v)(\forall w)(((P\ x\ y\ u)\ \&\ ((P\ y\ z\ v)\ \&\ (P\ u\ z\ w)))\supset(P\ x\ v\ w))$ given
(4) $(P\ e\ b\ b)$ UI from 1
(5) $(P\ a\ a\ e)$ UI from 2
(6) $(P\ a\ b\ c)$ given
(7) $(\forall x)(P\ x\ e\ x)$ given
(8) $(\forall x)(\forall y)(\forall z)(\forall u)(\forall v)(\forall w)(((P\ x\ y\ u)\ \&\ ((P\ y\ z\ v)\ \&\ (P\ x\ v\ w)))\supset(P\ u\ z\ w))$ given
(9) $(P\ a\ c\ b)$ all-detachment from 6 5 4 3
(10) $(P\ c\ c\ e)$ UI from 2
(11) $(P\ a\ e\ a)$ UI from 7
(12) $(P\ b\ b\ e)$ UI from 2
(13) $(P\ b\ c\ a)$ all-detachment from 11 10 9 8
(14) $(P\ c\ b\ a)$ all-detachment from 12 6 11 8
(15) $(P\ b\ a\ c)$ all-detachment from 6 14 13 3

By contrast, Wilson and Minker [25] give performance data for this problem on four theorem provers (this is the problem they call `Group2'). Where 21% of OSCAR's inferences are unnecessary, those theorems provers performed 79%, 87%, 96%, and 96% unnecessary inferences. Furthermore, these were presumably the results after tailoring the theorem provers to the problem to get the best performance. Stickel's new PTTP (*Prolog technology theorem prover*) solves this problem in a startling .284 seconds (OSCAR requires 8 seconds), but performs 1,136 inferences in the process. In terms of sheer speed, nothing can hold a candle to Stickel's PTTP on this problem, and I do not mean for these remarks to detract from its significance, but the percentage of unnecessary inferences is perhaps indicative of what can be expected from a resolution theorem prover that does not use strategies carefully tailored to the individual problem.

These redundancy figures hold up across a wide variety of problems. In solving Pelletier's propositional calculus problems (numbers 1-17), OSCAR's average percentage

of unnecessary inferences is only 8%, and on the pure predicate calculus problems (numbers 18-47), the redundancy figure only goes up to 26%. It appears that OSCAR's interest constraints are extremely effective in eliminating unnecessary inferences. This would seem to be the key to OSCAR's surprising speed.

It has been suggested repeatedly that OSCAR's efficiency results from the fact that "the system is just doing unit resolution". This is wrong for a number of reasons. First, OSCAR is not doing unit resolution, because it is not doing resolution at all. It is true, however, that all-detachment is similar to unit resolution in various ways. But it is also different in important respects. The most important difference is that it has nothing to do with Horn clauses. The disjuncts stored in the members of all-schemes can be of arbitrary logical complexity. Furthermore, on the steamroller problem, OSCAR outperforms resolution-based theorem provers using unit resolution. For instance, Stickel [23] lists twelve variations of his theorem prover, all giving strong preference to unit resolution, and OSCAR beats all but the six-second version.

It is a mistake to put too much emphasis on all-detachment. It is just one part of the theorem prover, albeit an important part. The rules for conditionalization, reasoning by cases, etc., are all essential components. It must be acknowledged that in the Schubert Steamroller problem and the group theory problem discussed above, most of the work is done by all-detachment. In some ways, those are poor test problems, because they are too simple. In problems of greater logical complexity, all of the proof operations may be used, and it is noteworthy that the proportion of unnecessary inferences remains fairly constant. It would be good to compare OSCAR with other theorem provers running some of the more complex of Pelletier's problems, but I know of no published data on how other theorem provers do on those or other problems of similar difficulty. Most published data concerns problems involving identity and function symbols, but the version of OSCAR under discussion cannot accommodate either. Still, it may be informative to examine OSCAR's behavior on one more problem. Pelletier's problem #40 consists of establishing that if there were an "anti-Russell set" (a set which contains exactly those sets which *are members* of themselves), then not every set has a complement. Symbolizing `x is a member of y' as $(F\,x\,y)$, the problem is as follows:

PROVING $[(\exists y)(\forall x)((F\,x\,y) \equiv (F\,x\,x)) \supset \sim(\forall z)(\exists w)(\forall v)((F\,v\,w) \equiv \sim(F\,v\,z))]$

19 adoption steps; 13 interest adoptions; 5 suppositions explored
19 inference steps used in proof; 5 suppositions used in proof
0% redundant adoptions; 0% redundant suppositions

SUPPOSITION: 1
ultimate interests:
$((\exists y)(\forall x)((F\,x\,y) \equiv (F\,x\,x)) \supset \sim(\forall z)(\exists w)(\forall v)((F\,v\,w) \equiv \sim(F\,v\,z)))$
adoptions: interests:
1. $((\exists y)(\forall x)((F\,x\,y) \equiv (F\,x\,x)) \supset \sim(\forall z)(\exists w)(\forall v)((F\,v\,w) \equiv \sim(F\,v\,z)))$ ultimate interest
jump to 2

MOVING TO SUPPOSITION: 2; sup-type C; origins: 1
  $(\exists y)(\forall x)((F\ x\ y) \equiv (F\ x\ x))$
      nil
      ultimate interests:
      $\sim(\forall z)(\exists w)(\forall v)((F\ v\ w) \equiv \sim(F\ v\ z))$
adoptions: interests:
 + + + +
1. $(\exists y)(\forall x)((F\ x\ y) \equiv (F\ x\ x))$ by supposition
2. $(\forall x)((F\ x\ @y) \equiv (F\ x\ x))$ EI from 1
        1. $\sim(\forall z)(\exists w)(\forall v)((F\ v\ w) \equiv \sim(F\ v\ z))$ ultimate interest
        2. $(\exists z)\sim(\exists w)(\forall v)((F\ v\ w) \equiv \sim(F\ v\ z))$ quantifier interchange for 1
3. $((F\ @y\ @y) \equiv (F\ @y\ @y))$ UI from 2
4. $((F\ @y\ @y) \equiv (F\ @y\ @y))$ biconditional simplification from 3
        3. $\sim(\exists w)(\forall v)((F\ v\ w) \equiv \sim(F\ v\ @y))$ EG for 2
        4. $(\forall w)\sim(\forall v)((F\ v\ w) \equiv \sim(F\ v\ @y))$ quantifier interchange for 3
jump to 3

MOVING TO SUPPOSITION: 3; sup-type U; origins: 2
  $(\exists y)(\forall x)((F\ x\ y) \equiv (F\ x\ x))$
      $\{\ *w\ \}$
      ultimate interests:
      $\sim(\forall v)((F\ v\ *w) \equiv \sim(F\ v\ @y))$
adoptions: interests:
 + + + +
1. $(\exists y)(\forall x)((F\ x\ y) \equiv (F\ x\ x))$ by supposition
2. $(\forall x)((F\ x\ @y) \equiv (F\ x\ x))$ EI from 1
3. $((F\ @y\ @y) \equiv (F\ @y\ @y))$ UI from 2
4. $((F\ @y\ @y) \equiv (F\ @y\ @y))$ biconditional simplification from 3
        1. $\sim(\forall v)((F\ v\ *w) \equiv \sim(F\ v\ @y))$ ultimate interest
        2. $(\exists v)\sim((F\ v\ *w) \equiv \sim(F\ v\ @y))$ quantifier interchange for 1
        3. $\sim((F\ @y\ *w) \equiv \sim(F\ @y\ @y))$ EG for 2
        4. $((F\ @y\ *w) \equiv (F\ @y\ @y))$ biconditional refutation for 3
        5. $\sim((F\ *w\ *w) \equiv \sim(F\ *w\ @y))$ EG for 2
        6. $((F\ *w\ *w) \equiv (F\ *w\ @y))$ biconditional refutation for 5
        7. $((F\ *w\ *w)\ \supset (F\ *w\ @y))$ biconditional introduction for 6
jump to 4

MOVING TO SUPPOSITION: 4; sup-type C; origins: 3
  $(F\ *w\ *w)$
  $(\exists y)(\forall x)((F\ x\ y) \equiv (F\ x\ x))$
      $\{\ *w\ \}$
      ultimate interests:
      $(F\ *w\ @y)$
adoptions: interests:
 + + + +

1. $(\exists y)(\forall x)((F\ x\ y) \equiv (F\ x\ x))$ by supposition
2. $(\forall x)((F\ x\ @y) \equiv (F\ x\ x))$ EI from 1
3. $((F\ @y\ @y) \equiv (F\ @y\ @y))$ UI from 2
4. $((F\ @y\ @y) \equiv (F\ @y\ @y))$ biconditional simplification from 3
5. $(F\ *w\ *w)$ by supposition
6. $(F\ *w\ @y)$ all-detachment from 5 2

jump back to 3

MOVING TO SUPPOSITION: 3; sup-type U; origins: 2
$(\exists y)(\forall x)((F\ x\ y) \equiv (F\ x\ x))$
{ $*w$ }
ultimate interests:
$\sim(\forall v)((F\ v\ *w) \equiv \sim(F\ v\ @y))$

adoptions: interests:
\+ + + +

5. $((F\ *w\ *w) \supset (F\ *w\ @y))$ conditionalization from supposition 4
8. $((F\ *w\ @y) \supset (F\ *w\ *w))$ biconditional introduction for 6

jump to 5

MOVING TO SUPPOSITION: 5; sup-type C; origins: 3
$(F\ *w\ @y)$
$(\exists y)(\forall x)((F\ x\ y) \equiv (F\ x\ x))$
{ $*w$ }
ultimate interests:
$(F\ *w\ *w)$

adoptions: interests:
\+ + + +

1. $(\exists y)(\forall x)((F\ x\ y) \equiv (F\ x\ x))$ by supposition
2. $(\forall x)((F\ x\ @y) \equiv (F\ x\ x))$ EI from 1
3. $((F\ @y\ @y) \equiv (F\ @y\ @y))$ UI from 2
4. $((F\ @y\ @y) \supset (F\ @y\ @y))$ biconditional simplification from 3
5. $((F\ *w\ *w) \supset (F\ *w\ @y))$ conditionalization from supposition 4
6. $(F\ *w\ @y)$ by supposition
7. $(F\ *w\ *w)$ all-detachment from 6 2

jump back to 3

MOVING TO SUPPOSITION: 3; sup-type U; origins: 2
$(\exists y)(\forall x)((F\ x\ y) \equiv (F\ x\ x))$
{ $*w$ }
ultimate interests:
$\sim(\forall v)((F\ v\ *w) \equiv \sim(F\ v\ @y))$

adoptions: interests:
\+ + + +

6. $((F\ *w\ @y) \supset (F\ *w\ *w))$ conditionalization from supposition 5
7. $((F\ *w\ *w) \equiv (F\ *w\ @y))$ biconditional introduction from 5 6

8. $\sim((F\ {*w}\ {*w}) \equiv \sim(F\ {*w}\ @y))$ biconditional refutation from 7
9. $(\exists v)\sim((F\ v\ {*w}) \equiv \sim(F\ v\ @y))$ EG from 8
10. $\sim(\forall v)((F\ v\ {*w}) \equiv \sim(F\ v\ @y))$ quantifier interchange from 9
jump back to 2

MOVING TO SUPPOSITION: 2; sup-type C; origins: 1
$(\exists y)(\forall x)((F\ x\ y) \equiv (F\ x\ x))$
nil
ultimate interests:
$\sim(\forall z)(\exists w)(\forall v)((F\ v\ w) \equiv \sim(F\ v\ z))$
adoptions: interests:
\+ + + +
5. $(\forall w)\sim(\forall v)((F\ v\ w) \equiv \sim(F\ v\ @y))$ UG from supposition 3
6. $\sim(\exists w)(\forall v)((F\ v\ w) \equiv \sim(F\ v\ @y))$ quantifier interchange from 5
7. $(\exists z)\sim(\exists w)(\forall v)((F\ v\ w) \equiv \sim(F\ v\ z))$ EG from 6
8. $\sim(\forall z)(\exists w)(\forall v)((F\ v\ w) \equiv \sim(F\ v\ z))$ quantifier interchange from 7
jump back to 1

MOVING TO SUPPOSITION: 1
ultimate interests:
$((\exists y)(\forall x)((F\ x\ y) \equiv (F\ x\ x)) \supset \sim(\forall z)(\exists w)(\forall v)((F\ v\ w) \equiv \sim(F\ v\ z)))$
adoptions: interests:
\+ + + +
1. $((\exists y)(\forall x)((F\ x\ y) \equiv (F\ x\ x)) \supset \sim(\forall z)(\exists w)(\forall v)((F\ v\ w) \equiv \sim(F\ v\ z)))$ conditionalization from supposition 2

THEOREM
to prove: $((\exists y)(\forall x)((F\ x\ y) \equiv (F\ x\ x)) \supset \sim(\forall z)(\exists w)(\forall v)((F\ v\ w) \equiv \sim(F\ v\ z)))$

suppose, with variables { $*w$ }
$(F\ {*w}\ @y)$
$(\exists y)(\forall x)((F\ x\ y) \equiv (F\ x\ x))$
(1) $(\exists y)(\forall x)((F\ x\ y) \equiv (F\ x\ x))$ by supposition
(2) $(\forall x)((F\ x\ @y) \equiv (F\ x\ x))$ EI from 1
(3) $(F\ {*w}\ @y)$ by supposition
(4) $(F\ {*w}\ {*w})$ all-detachment from 3 2

suppose, with variables { $*w$ }
$(F\ {*w}\ {*w})$
$(\exists y)(\forall x)((F\ x\ y) \equiv (F\ x\ x))$
(5) $(\exists y)(\forall x)((F\ x\ y)\ \ (F\ x\ x))$ by supposition
(6) $(\forall x)((F\ x\ @y) \equiv (F\ x\ x))$ EI from 5
(7) $(F\ {*w}\ {*w})$ by supposition
(8) $(F\ {*w}\ @y)$ all-detachment from 7 6

suppose, *w*ith variables { **w* }
($\exists y$)($\forall x$)((*F x y*) ≡ (*F x x*))
(9) ((*F *w @y*) ⊃ (*F *w *w*)) conditionalization from 4
(10) ((*F *w *w*) ⊃ (*F *w @y*)) conditionalization from 8
(11) ((*F *w *w*) ≡ (*F *w @y*)) biconditional introduction from 10 9
(12) ~((*F *w *w*) ≡ ~(*F *w @y*)) biconditional refutation from 11
(13) ($\exists v$)~((*F v *w*) ≡ ~(*F v @y*)) EG from 12
(14) ~($\forall v$)((*F v *w*) ≡ ~(*F v @y*)) quantifier interchange from 13

suppose, *w*ith variables {}
($\exists y$)($\forall x$)((*F x y*) ≡ (*F x x*))
(15) ($\forall w$)~($\forall v$)((*F v w*) ≡ ~(*F v @y*)) UG from 14
(16) ~($\exists w$)($\forall v$)((*F v w*) ≡ ~(*F v @y*)) quantifier interchange from 15
(17) ($\exists z$)~($\exists w$)($\forall v$)((*F v w*) ≡ ~(*F v z*)) EG from 16
(18) ~($\forall z$)($\exists w$)($\forall v$)((*F v w*) ≡ ~(*F v z*)) quantifier interchange from 17

suppose, *w*ith variables {}
-
(19) (($\exists y$)($\forall x$)((*F x y*) ≡ (*F x x*)) ⊃ ~($\forall z$)($\exists w$)($\forall v$)((*F v w*) ≡ ~(*F v z*)))
conditionalization from 18

It is noteworthy how tightly focused this reasoning is. There are *no* unnecessary steps.

I remarked above that OSCAR is not written for efficiency. OSCAR's speed does not result from fancy programing, but from the general architecture of the system. This suggests that OSCAR's architecture has an inherent advantage over resolution theorem provers. OSCAR can, in effect, do proofs by resolution refutation. That is pretty much what proofs by reductio amount to. But OSCAR is much faster when it does not have to resort to proofs by reductio. Taking out the other rules and forcing OSCAR to reason by reductio makes the reasoning markedly slower. Correspondingly, human logicians tend to use reductio as a last resort strategy, to be attempted only when other strategies are inapplicable.

This suggests that resolution be viewed as just one possible inference rule among many. As I remarked above, the intuition underlying resolution-based systems is that resolution is the kind of operation computers can perform very quickly. The above performance data suggests that that intuition is sound. Even when OSCAR solves a problem faster than resolution based systems, OSCAR performs many fewer inferences per unit of time than do the resolution-based systems. OSCAR trades speed on individual inferences for inference rules that are better focused and require vastly fewer inferences. It seems that that is often more than an even trade.[11] The same information can be encoded

[11] However, this may be called into doubt by the startling speed of Stickel's PTTP system if its performance can be maintained on hard problems.

in different, logically equivalent, ways by using different logical operators. Resolution focuses on just disjunction, negation, and universal generalization. Perhaps the explanation for why human beings employ additional logical operators is that there are efficient strategies for reasoning with them.

REFERENCES

1. Andrews, P. B., F. Pfenning, S. Issar, C. P. Klapper, The TPS Theorem Proving System. *8th International Conference on Automated Deduction*, ed. J. H. Siekman. Springer, 1985.
2. Andrews, P. B., D. A. Miller, E. L. Cohen, and F. Pfenning, Automating higher order logic. *Automated Theorem Proving: After 25 Years*, ed W. W. Bledsoe and D. W. Loveland. American Mathematical Society, 1983.
3. Bledsoe, W., Non-reduction theorem proving, *AI* 9 (1977), 1-35.
4. Bledsoe, W., Some automatic proofs in analysis. *Automated Theorem Proving: After 25 Years*, ed W. W. Bledsoe and D. W. Loveland. American Mathematical Society, 1983.
5. Bledsoe, W., and Tyson, M., The UT interactive theorem prover, The University of Texas at Austin Math Dept. Memo ATP-17, 1975.
6. Boyer, R., and Moore, J., *A Computational Logic* (Academic Press, 1979).
7. Chang, C., and Lee, R., *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, 1973.
8. Cohn, A., On the solution of Schubert's steamroller in many sorted logic. *IJCAI85*.
9. Harman, G., *Change in View* (MIT press, 1986).
10. Lusk, E., and R. Overbeek, The automated reasoning system ITP. ANL-84-27, Argonne National Laboratory, 1984.
11. Lusk, E., W. McCune, R. Overbeek, ITP at Argonne National Laboratory. *8th International Conference on Automated Deduction*, ed. J. H. Siekman. Springer, 1985.
12. Murray, N. V., Completely non-clausal theorem proving. *Artificial Intelligence* 18 (1982), 67-85.
13. Nevins, A., A Human oriented logic for automatic theorem proving. *Jour. Assoc. for Comput. Mach.* 21 (1974), 603-621.
14. Oppacher, F., E. Suen, Controlling Deduction with Proof Condensation and Heuristics. *8th International Conference on Automated Deduction*, ed. J. H. Siekman. Springer, 1985.
15. Oppacher, F., E. Suen, HARP: A tableau-based theorem prover. *The Journal of Automated Reasoning 4* (1988), 69-100.
16. Pelletier, F. J., *Completely Non-Clausal, Completely Heuristically Driven, Automated Theorem Proving*. Dept. of Computing Science, University of Alberta, Tech. Report TR82-7, 1982.
17. Pelletier, F. J., THINKER. *8th International Conference on Automated Deduction*, ed. J. H. Siekman. Springer, 1985.
18. Pelletier, F. J., Seventy-five problems for testing automatic theorem provers. *Journal of Automated Reasoning* 2 (1986), 191-216. See the errata in *The Journal of Automated Reasoning 4* (1988), 235-6.
19. Pollock, J., Defeasible reasoning, *Cognitive Science* 11 (1987), 481-518.
20. Pollock, J., *How to Build a Person.* Bradford/MIT Press, 1989.
21. Reiter, R., A semantically guided deductive system for automatic theorem proving,

IJCAI 3 (1973), 41-46.
22. Stickel, M. E., The KLAUS Automated Theorem Proving System. *8th International Conference on Automated Deduction*, ed. J. H. Siekman. Springer, 1985.
23. Stickel, M. E., Schubert's steamroller problem: formulations and solutions. *Journal of Automated Reasoning* 2 (1986) 89-101.
24. Stickel, M. E., A Prolog technology theorem prover: implementation by an extended Prolog compiler. *The Journal of Automated Reasoning 4* (1988), 353-380.
25. Wilson, G. A., and Minker, J., Resolution, refinements, and search strategies: A comparative study. *IEEE Transactions on Computers C-25* (1976), 782-801.
26. Wos, L., and S. Winker, Open questions solved with the assistance of AURA. *Automated Theorem Proving: After 25 Years*, ed W. W. Bledsoe and D. W. Loveland. American Mathematical Society, 1983.