THE USE OF MODELS IN AUTOMATIC THEOREM-PROVING

Raymond Reiter

Department of Computer Science

University of British Columbia

Vancouver 8, British Columbia

Canada

September, 1972

The Use of Models in Automatic Theorem-Proving

Raymond Reiter
Department of Computer Science
University of British Columbia
Vancouver 8, B.C.
Canada

ABSTRACT

This paper is an extended argument in favour of the use of models in automatic theorem-proving. Gelernter's Geometry Machine is taken as a prototype of such a theorem-prover, and is analyzed with the objective of extracting its underlying semantic and syntactic formal system. This system is later generalized to permit, among other things, arbitrary well-formed formulae, arbitrary models, the use of models in making inferences and the dynamic modification of models as the proof unfolds. The purely logical component of this generalized system resembles natural deduction, rather than resolution. The various semantic features of the formal system are discussed and illustrated through a number of examples. In particular, the system is seen to possess a very smooth and natural interface between the semantics and the deductive syntax. These syntactic and semantic subsystems interact continuously during the search for a proof, each suggesting to the other how next to proceed. Another feature is the use of semantic information to minimize back-up due to dead-end searches. Particularly appealing is the naturalness of the system and its close correspondence with many "people oriented" techniques for proof discovery.

The notions of "model" and "truth in a model" are defined for quantifier-free formulae. It is found that the semantic theory so defined is, in some respects, too weak. Infinite models are considered and an attempt is made to deal with the problem of finitely representing such models.

Finally, relationships are indicated between the semantic approach to theorem-proving and other areas of artificial intelligence, specifically natural language processing, generalization and hypothesis formation, and the representation of new knowledge.

1. Introduction

The past dozen years or so have witnessed a great deal of research and programming energy devoted to mechanizing first-order logic. Motives vary over a spectrum of applications: exhibiting intelligent behaviour in mathematics [3,4], proving correctness of computer programs [17], information retrieval [5], natural language processing [24], robot problem-solving [7]. Several first-order proof procedures have been proposed and implemented with varying degrees of success. Among these are systems of natural deduction [25], Herbrand search procedures [6], and resolution [20].

It quickly became apparent that these proof procedures alone cannot succeed on any interesting mathematical theory, at least not within the lifetime of the universe. One approach towards alleviating these difficulties was to develop completeness preserving refinements of the rules of inference. Essentially, these are suitably restricted rules, often depending upon the syntactic structure of the current formulae, which generate a narrower (but usually deeper) search tree. Virtually all of the results obtained along these lines are for resolution systems. Examples are resolution with merging [2], linear resolution [16], A-ordering [22] etc. plus a whole host of combined strategies. Experimental evidence [27] indicates that this approach alone fails on even mildly serious theorems.

Virtually everyone is now agreed that knowledge about the problem domain must be used in the logic. The question is how. There seem to be two approaches.

1.1 Semantics as Domain Dependent Heuristics

In this approach, semantic information is embedded, in the logic, as suitable domain dependent heuristics which act like new rules of

inference. No representation of the problem domain itself is present. Semantics is conveyed through some fixed, finite set of heuristic procedures representing that knowledge of the problem domain which is believed to be significant in guiding the search for proofs. A typical paradigm is something like: "If the current formula(e) have such and such a syntactic structure and/or if the parameter α is less than 7.3, then apply the following domain dependent rule of inference." (Of course, most systems based upon this approach are considerably more sophisticated than that - we want only to suggest their flavour.) The point here is that this semantic information is incorporated into the system by augmenting its purely syntactic rules. The theorem-prover continues to be syntax-driven.

Examples of this approach are [4] for analysis, [3] for set theory, [18] for equality and [23] for partial orderings. It is reasonably clear that such domain dependent heuristics will be essential components of any theorem-prover capable of doing real mathematics. For example, a number theorist will require procedures for solving various kinds of equations and for formula manipulation. Bledsoe's system for limit theorems in analysis [4] embodies just such procedures and is an excellent example of their integration into a logical system. We shall argue that much more than this is required.

If, as we remarked, the theorem-prover is still being syntax-driven, what fundamental difference is there between this approach and the refinement techniques for resolution? In principle, we believe there is none. These heuristics are merely domain dependent refinements.

Because these heuristics are embedded at the syntactic level of proof generation, every node of the search tree is treated uniformly. This in itself is no criticism. However, each node represents a first

order formula which, presumably, means something on the problem domain. There are arbitrarily many possible formulae - hence arbitrarily many possible meanings. But there are only finitely many semantic heuristics coded at the inference level. In effect therefore, this approach assumes that the set of all possible formulae can be collapsed to a finite set of equivalence classes and that each such class characterizes the meaning of its member formulae. Moreover, this finite set of "meanings" is sufficient semantic information to efficiently guide the search for proofs. We consider neither of these consequences to be plausible.

However, there seems to be a way out. One could argue that these finitely many heuristics need not be independent. If we allow them to interact in highly complex ways, perhaps we could distinguish semantically among all possible formulae. Fine. But how do we determine these interactions? For that matter, how do we discover the initial base set of semantic heuristics? We must anticipate, at the coding stage, all of that knowledge the domain has to offer which could be of assistance in discovering proofs, together with their interactions. If later we discover a new heuristic, this can lead to a major overhaul of the program. There is a very real danger of an overproliferation of special, mutually interacting heuristic procedures. In the limit, a program relying solely on this approach might suffocate in its own code. Briefly, then, this approach lacks transparency, flexibility and generality.

Worse still is the lack of any kind of reasonable control over dead-end searches. If the application of an heuristic or rule of inference leads the proof astray, there is no provision for using knowledge about the problem domain to detect this. Those techniques which are currently used such as setting parameters for maximal clause length or depth of

function-nesting are clearly ad hoc, and independent of the domain. It might be argued that special purpose heuristics can be developed to detect blind alleys. Perhaps - but this would only aggravate the first problem.

This difficulty with blind alleys is compounded in the presence of a large number of axioms and theorems which might be irrelevant to the proof being sought. Such formulae are guaranteed to lead to dead-end searches. There is no way that a serious theorem-proving system can avoid having to deal with this situation. To our knowledge, no current implementation of a theorem-prover, all of which are based on refinements and/or domain dependent heuristics, is capable of coping with this problem.

1.2 Semantics as the Representation of Models

The main thrust of this paper is the following: Instead of relying entirely upon heuristic procedures which represent fixed, a priori knowledge about the problem domain, we represent the problem domain itself i.e. we present to the theorem-prover a model of the axiomatic system involved. Of course, the representation of a model is, by itself, useless. In addition what is needed is a set of procedures for extracting information about the model when required by the theorem-prover together with a flexible, general interface between such a semantic subsystem and the purely syntactic logical system. The distinction, therefore, between this approach and that based upon domain dependent heuristics is that the latter explicitly represents that semantic information which is believed to be a priori relevant, while the former implicitly represents all of the information available in the model which is capable of being extracted by the available procedures.

The idea of using models for theorem-provers is by no means new. Gelernter and his co-workers [8,9,10,11,12] developed a system for plane geometry whose success was due primarily to its use of geometric diagrams. Because our own ideas evolve very naturally from theirs, we devote Section 2 to its description. The only other developed theory of the use of models in theorem-proving is due to Slagle [22]. If we ignore such niceties as what we mean by a model, and truth in a model, we can describe his system as follows: The only rule of inference is clash-resolution [21]. If M is a model, C, $E_1,\ldots,E_q$ are clauses, and σ a most general substitution such that

$C\sigma = KU\{L_1,\ldots,L_q\}$ is true in M

$E_i\sigma = F_iU\{\overline{L}_i\}$ is false in M

and $R = K \cup F_1 \cup \ldots \cup F_q$ is false in M

then R is a <u>latent maximal semantic resolvent</u>. These are the only resolvents which need to be inferred. Slagle proves the completeness of this inference rule, and gives two examples of semantic proofs. Unfortunately this approach has never caught on, possibly because these maximal resolvents are difficult and costly to compute, and possible because Slagle's paper is very abstractly written. In any event we believe that his system is worthy of closer study, and that many of the ideas of the present paper can be profitably incorporated into semantic resolution.

The rest of this paper is devoted to an extended argument in favour of the use of models in theorem-proving. Section 2 is a study of the Geometry Machine with the principal goal of abstracting its underlying semantic and syntactic formal system, and isolating those deficiencies which prevent the general use of this formal system. Section 3 provides formal definitions of "model" and "truth in a model" and attempts to come to grips with the problem of finitely representing infinite models. In

the process, some unresolved problems are uncovered associated with specifying semantics for Skolemized formulae. Section 4 generalizes the formal system underlying the Geometry Machine, and discusses some of its features. Section 5 indicates relationships between the semantic approach to theorem-proving and other areas of artificial intelligence. Section 6 concludes the paper with some suggestions for further research.

2. The Geometry Theorem Machine Revisited

In a series of articles [8,9,10,11,12] Gelernter and his co-workers describe a theorem-proving system for elementary plane geometry. With the exception of Gilmore's careful analysis of the logic underlying their system [13] this work has been largely ignored by the theorem-proving community, despite the obvious success of their use of semantics in dealing with many of the problems which plague current theorem-proving programs. In this section we outline the basic semantic and logical features of Gelernter's work, with a view towards generalizing these ideas in Section 4. Much of this outline relies on Gilmore's study [13].

2.1 The Axioms

Gilmore distinguishes two classes of axioms underlying the Geometry Machine:

Class 1 - Universally quantified axioms. With the exception of the identity axiom $(x)(x=x)$ these are all implication sentences i.e. sentences of the form

$$(x_1)\ (x_2)\ \ldots\ (x_n)\ (L_1 \wedge L_2 \wedge \ldots \wedge L_m \supset L_{m+1})$$

where $L_1,\ldots,L_{m+1}$ are literals and $x_1,\ldots,x_n$ are all of the free variables occurring in the L's. The x's are interpreted as points. In addition, every axiom which is an implication sentence has the property that each variable occurring in the conclusion L_{m+1} occurs also in at least one of the premises $L_1,\ldots,L_m$. Let us call such sentences p-implication sentences.

Class 2 - Existentially quantified axioms. In fact, there is just one, namely

$$(xyzw)[\sim xy \,||\, zw \supset \exists\, u(\text{coll } xyu \wedge \text{coll } zwu)]$$

which asserts that if line segment xy is not parallel to zw, then there exists a point u such that x,y and u are collinear, and z,w and u are collinear. However, this axiom is used in a special way which we describe later. The important point is that it never explicitly interacts with the

purely logical rules of inference, which we now describe.

2.2 The Rules of Inference

Gelernter's system is designed to succeed on theorems in geometry which are p-implication sentences derivable from the Class 1 axioms. Of course, previously proved theorems may be used in the proof, but since these are also p-implication sentences, their form is the same as that of the axioms. Let $(y_1)\ldots(y_r)\mathcal{A}(y_1,\ldots,y_r)$ be the prenex normal form of the conjunction of all of the Class 1 axioms and previously proved theorems. Let $(x_1)\ldots(x_n)[H_1\wedge\ldots\wedge H_m\supset S]$, briefly $(x_1)\ldots(x_n)H(x_1,\ldots,x_n)$, be the current theorem to be proved. Then our current goal is

$$\vdash(y_1)\ldots(y_r)\mathcal{A}(y_1,\ldots,y_r)\supset(x_1)\ldots(x_n)H(x_1,\ldots,x_n)$$

In Skolemized form (see Section 3.2) this goal is

$$\vdash\mathcal{A}(y_1,\ldots,y_r)\supset H(A_1,\ldots,A_n)$$

where $A_1,\ldots,A_n$ are 0-place Skolem functions. This is equivalent to

$$\vdash\mathcal{A}(y_1,\ldots,y_r)\wedge H_1(A_1,\ldots,A_n)\wedge\ldots\wedge H_m(A_1,,..,A_n)\supset S(A_1,\ldots,A_n) \qquad (1)$$

which is Gelernter's top level goal. The basic rules of inference involve substitution for variables, and "backward chaining". Thus to establish $S(A_1,\ldots,A_n)$, find an H identical to it, or an axiom α of $\mathcal{A}(y_1,\ldots,y_r)$ whose conclusion is of the form $S(y_1,\ldots,y_n)$. If such an α can be found, make the substitution $A_1|y_1,\ldots,A_n|y_n$ in the premises of α. If the premises still contain free variables, substitute for these uniformly choosing arbitrary elements from the set $\{A_1,\ldots,A_n\}$. Equivalently, find a substitution instance σ over the Herbrand Universe $\{A_1,\ldots,A_n\}$ such that $\alpha\sigma$ has no free variables, and such that $\alpha\sigma$ has $S(A_1,\ldots,A_n)$ as its conclusion. Having succeeded in discovering such an α, we take the premises of $\alpha\sigma$ to be the current goals, all of which must succeed, and proceed recursively. The rules of inference can clearly be described as "reasoning backwards from conclusion to premises". We can present these rules more

precisely via the following formal system whose resemblance to that used by Bledsoe, Boyer and Henneman [4] is intentional:

	Current Subgoal	Next Subgoal
1.	$R \vdash B \supset C$	
	If B and C are identical	T (true)
	If there exists a substitution σ such that $B\sigma = C$	T
2.	$R \vdash (K \supset B) \supset C$	
	If there exists σ such that $B\sigma = C$ and $K\sigma$ has no free variables	$\vdash R \supset K\sigma$
3.	$R \vdash W \supset U \wedge V$	$R \vdash W \supset U$ and $R \vdash W \supset V$
4.	$R \vdash U \wedge V \supset W$	$R \wedge V \vdash U \supset W$ or $R \wedge U \vdash V \supset W$

Here B and C are literals and K a conjunction of one or more literals. U, V and W are arbitrary wffs. R is what Bledsoe et al. call the reserve. Initially, R is the null conjunct ϕ and the top level subgoal is given by (1). σ , of course, is a substitution over $\{A_1,\ldots,A_n\}$. The rules are applied, in order, to any current subgoal. If any rule fails, the next rule is attempted. If all fail on the current subgoal, NIL (false) is returned.

Example 1. The Pons Asinorum. In an isosceles triangle, the base angles are equal. Only one axiom is required - "If two sides and the contained angle of one triangle are respectively equal to the two sides and contained angle of another triangle, then corresponding angles are equal".

α: Δ xyz $\wedge$ Δ uvw $\wedge$ xy=uv $\wedge$ xz=uw $\wedge$ $\angle$ yxz= $\angle$ vuw $\supset$ $\angle$ xyz= $\angle$ uvw

The top level subgoal is

$\phi \vdash \alpha \wedge \Delta$ ABC $\wedge$ AB=AC $\supset$ $\angle$ ABC= $\angle$ ACB

Two applications of Rule 4 yield the following three subgoals, one of which must succeed.

1. $\alpha \wedge \Delta$ ABC $\vdash$ AB=AC $\supset$ $\angle$ ABC= $\angle$ ACB

This fails.

2. $\alpha \wedge$ AB=AC $\vdash \Delta$ABC $\supset \angle$ ABC= $\angle$ ACB

This fails.

3. AB=AC $\wedge \Delta$ ABC $\vdash \alpha \supset \angle$ ABC = $\angle$ ACB

Rule 2 applies and yields σ = (A|x, B|y, C|z, A|u, C|v, B|w). The current subgoal is

4. $\phi \vdash$ AB=AC $\wedge \Delta$ ABC $\supset \Delta$ ABC $\wedge \Delta$ ACB $\wedge$ AB=AC $\wedge$ AC=AB $\wedge \angle$ BAC= $\angle$ CAB

Four applications of Rule 3 yield

5. $\phi \vdash$ AB=AC $\wedge \Delta$ ABC $\supset \Delta$ ABC

6. $\phi \vdash$ AB=AC $\wedge \Delta$ ABC $\supset \Delta$ ACB

⋮

9. $\phi \vdash$ AB=AC $\wedge \Delta$ ABC $\supset \angle$ BAC= $\angle$ CAB

all of which must succeed. These all succeed by an application of Rule 4 followed by Rule1provided, in the case of 6. 8. and 9. the system knows a little bit about triangles and equality of line segments and angles.

2.3 Semantics - The Use of Diagrams

Clearly, if the above formal system were unleased on any full-blown set of axioms and theorems of plane geometry there would be an uncomfortably long wait for most proofs. One of the ways Gelernter et al. prune the search tree generated by the rules of inference is by means of a geometric diagram. Here the Skolem constants $A_1,\ldots,A_n$ are interpreted as fixed points in the plane and the occurring predicates like Δ , coll etc. and functions

like midpt, intersectionpt, etc. are given their usual geometric interpretation. The actual co-ordinates assigned to $A_1,\ldots,A_n$ must be so chosen as to make the hypotheses $H_1(A_1,\ldots,A_n),\ldots,H_m(A_1,\ldots,A_n)$ true when interpreted in the diagram. By a model M we mean such a co-ordinate assignment to the A's, together with the standard geometric interpretation of the predicate and function symbols. If W is a wff, $M \models W$ means that W is true in M and $M \not\models W$, that W is false in M. This, of course, is simply an informal description of the usual Tarskian definition of model and truth in a model. (We give a formal definition in Section 3.) It follows that

$M \models H_1(A_1,\ldots,A_n) \wedge \ldots \wedge H_n(A_1,\ldots,A_n)$.

Also $M \models (y_1)\ldots(y_r)\mathcal{a}(y_1,\ldots,y_r)$. This is not so obvious as it might seem at first. It is true because of the special form of the axioms as implication sentences. If, for example, the Class 2 intersection axiom had been included in $\mathcal{a}$, no model with finite domain $\{A_1,\ldots,A_n\}$ would in general simultaneously satisfy $H_1,\ldots,H_m$ and $\mathcal{a}$. This is so because all possible points of intersection of non parallel line segments would have to be explicitly indicated in the diagram. But these new points themselves define line segments which in turn define new points of intersection, etc. Strictly speaking, there is no finite model for full plane geometry (as opposed to the fragment of plane geometry considered by Gelernter et al.) However, there is a sense in which there is a useable model, and we shall return to this question in Section 3.

Now consider Rule 2 of the rules of inference and suppose that $M \not\models K_\sigma$. In that case $R \supset K_\sigma$ is not provable. This is so because each conjunct of R is either an axiom, or one of $H_1,\ldots,H_m$ (an easy induction) and hence R is true in M. Thus there is a model for R in which K^σ is false

so by the completeness theorem for first order logic, $R \supset K\sigma$ is not provable. It follows that, without diminishing the class of theorems provable within this formal system, we can replace Rule 2 by

2'. $R \vdash (K \supset B) \supset C$

If there exists σ such that

$B\sigma = C$ and $K\sigma$ has no free

variables and $M \models K\sigma$ $\qquad\qquad \vdash R \supset K\sigma$

This is the syntactic and semantic formal system underlying the Geometry Theorem Machine. We call this system L_G.

Example 2.

In the presence of the axiom

β: $xy || uv \wedge \text{coll } xwy \wedge \text{coll } uzv \supset \angle xwz = \angle vzw$

one of the subgoals generated for the Pons Asinorum would be

$\Delta\, ABC \wedge AB = AC \vdash \beta \supset \angle ABC = \angle ACB$

Rule 2' would require the substitutions $A|x, B|w, C|z, A|v$. But there is no substitution for u and y from {A,B,C} which makes

$Ay || uA \wedge \text{coll } ABy \wedge \text{coll } uCA$

true in a diagram of an isosceles triangle ABC. Hence, this subgoal fails.

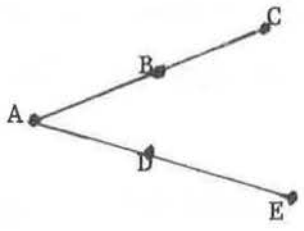
2.4 The Intersection Axiom

As we remarked earlier, this Class 2 axiom is treated by Gelernter in a special way. This is the only axiom which permits the introduction of new points. Equivalently, this axiom allows for augmenting the model M by explicitly adding new points and line segments to the diagram. However, in Gelernter's system this is not done dynamically, as the search for a proof unfolds, but only when the search for a proof fails with the given model M. In that case, a pair of non parallel lines in the diagram is chosen (it is not clear from [8,9,10,11,12] how the lines are chosen),

their point of intersection E is determined, the new diagram together with the new domain $\{E,A_1,\ldots,A_n\}$ serves as the new model and a proof is again attempted with L_G. This is obviously an ad hoc procedure for dealing with one instance of a general problem in plane geometry, namely - What constructions should be made? It is clear that the formal system L_G is well suited for proofs which require no constructions to be made in the initial diagram, and that L_G alone will fail on any theorem requiring a construction. Thus the Geometry Machine would succeed (and did) in finding the elegant proof of Example 1 for the Pons Asinorum. It could not, even with the intersection axiom, find the usual high school proof which requires extending the bisector of angle A to meet the base BC, since this requires an axiom stating that every angle has a bisector. Such an axiom is existential and hence not a Class 1 axiom.

2.5 Another Use of the Diagram

The Geometry Machine exhibits a certain degree of selectivity between what it proves formally, and what it decides is obvious from the diagram. For example, if a current subgoal is $\angle BAD=\angle CAE$ and if part of the diagram looks like



the Geometry Machine returns "True - by the diagram", rather than a tedious syntactic proof of the same subgoal. Aside from the obvious economy derived from avoiding certain syntactic subproofs, this methodology requires fewer axioms. Thus, instead of cluttering up the axiom set with axioms like

coll xyz $\wedge$ coll xuv $\wedge$ between xyz $\wedge$ between xuv $\supset \angle zxv = \angle yxu$

the Geometry Machine embeds this as a search procedure on the model.

This "confusion" of the motions "provable" and "true in the model" is one that mathematicians profitably make all the time. For example the statement 2+3=5 is true in the standard model for number theory, and so we accept it as provable, despite the fact that strictly speaking it should be given a syntactic proof within Peano arithmetic. It is clear that theorem-provers in general must be capable of making this "confusion" in order to avoid trivial, but possibly lengthy deductions.

2.6 Summary

The Geometry Machine uses the semantic and syntactic system L_G with a fixed model M whose diagram satisfies the initial hypotheses of the theorem to be proved. The model M may be altered only by appealing to the intersection axiom which allows for the addition of new points (as intersections of old lines) and new line segments. This is done only when L_G fails with the model M. In that case the new model is presented to L_G, and a fresh proof attempted.

All of the axioms which interact with the system L_G are p-implication sentences, as are the theorems to be proved. In particular, the semantic and syntactic rules are incapable of coping with existentially quantified axioms or theorems.

The formal system L_G defines a negationless logic since the negation sign does not appear explicitly in any of its rules of inference. Similarly, there is no rule for handling disjunctions.

The rest of this paper is devoted to generalizing the system L_G. In particular, the generalized logic and semantics will provide for:

1. dynamic modification of models as the proof unfolds
2. arbitrary wffs as axioms and theorems
3. negation and disjunction
4. infinite models (in a sense to be defined)
5. special treatment of the equality predicate
6. use of the model to make inferences

3. Interpretation of Wffs in a Model

In the case of the Geometry Machine, all theorems have a specialized form:

$H_1(A_1,\ldots,A_n) \wedge \ldots \wedge H_m(A_1,\ldots,A_n) \supset S(A_1,\ldots,A_n)$

and the associated models all have the same characteristics - a fixed diagram with finitely many points, each point of which is bound to one of the Skolem constants $A_1,\ldots,A_n$ and conversely. To test $M \models W(x_1,\ldots,x_r, A_1,\ldots,A_n)$ try to find points $x_1,\ldots,x_r$ in the diagram which make W true. For the remainder of this paper, we want to adopt a general view of syntax and semantics for first order logic, as they apply to theorem-proving in mathematical theories. The intuitive notions of model, and truth in a model, which we have been assuming in the description of the Geometry Machine will no longer be adequate.

3.1 Some Definitions

We are dealing with the usual alphabet of symbols for quantifier-free first order logic:

Variables x, y, z, ...,

Predicate symbols P, Q, R, ... , each of an arbitrary number of arguments.

Function symbols f, g, h, ... , each of an arbitrary number of arguments.

Logical connectives $\supset$, $\vee$, $\wedge$, $-$ (no quantifiers).

Punctuation symbols as needed.

Terms, well formed formulae (wffs), atoms, and literals have their usual definitions [6]. An interpretation I is specified by a non empty set D (the domain) together with

1. For each n-ary predicate symbol P, an associated function P_I: $D^n \rightarrow \{0,1\}$.
2. For each n-ary function symbol f, an associated function f_I: $D^n \rightarrow D$.

The interpretation I is finite provided D is finite. Define, for $a_1,\ldots,a_n \varepsilon D$

$[x_i]_I(a_1,\ldots,a_n) = a_i \quad 1 \leq i \leq n$

If $t^{(1)},\ldots,t^{(m)}$ are terms and f is an m-ary function symbol, and $x_1,\ldots,x_r$ are all of the variables occurring in these terms, then for $1 \leq r \leq n$, define

$[f\ (t^{(1)},\ldots,t^{(m)})]_I\ (a_1,\ldots,a_n) = f_I([t^{(1)}]_I\ (a_1,\ldots,a_n),\ldots,[t^{(m)}]_I(a_1,\ldots,a_n))$

If P is an m-ary predicate symbol, then

$[P\ t^{(1)},\ldots,t^{(m)}]_I(a_1,\ldots,a_n) = P_I[t^{(1)}]_I(a_1,\ldots,a_n),\ldots,[t^{(m)}]_I(a_1,\ldots,a_n)$

We extend these ideas to non-atomic wffs in the usual way. Define

$[\overline{W}]_I(a_1,\ldots,a_n) = 1$ if $[W]_I(a_1,\ldots,a_n) = 0$

$= 0$ if $[W]_I(a_1,\ldots,a_n) = 1$

$[W \wedge W']_I(a_1,\ldots,a_n) = 1$ if $[W]_I(a_1,\ldots,a_n) = 1$ and $[W']_I(a_1,\ldots a_n)=1$

$= 0$ otherwise

etc.

We want to define the notion "wff W is true in I". Before doing so, we must agree on whether the free variables of W are to be interpreted as universally quantified, or existentially quantified. Later, we shall need both notions. Therefore, define

$I \models_E W$ iff there exist $a_1,\ldots,a_n \varepsilon D$ such that $[W]_I(a_1,\ldots,a_n) = 1$

$I \models_U W$ iff for all $a_1,\ldots,a_n \varepsilon D$, $[W]_I(a_1,\ldots,a_n) = 1$

If $I \models_U W$, then I is a model for W. This is the usual Tarskian definition of model. There are a number of immediate consequences of these definitions.

$I \models_E \overline{W}$ iff not $I \models_U W$

$I \models_E WVW'$ iff $I \models_E W$ or $I \models_E W'$

If $I \models_E W \supset W'$ and $I \models_U W$ then $I \models_E W'$.

This last result provides the basis for our use of semantics in theorem-proving. If W is taken to be the axioms and W' the theorem to be proved, we determine an appropriate model M of W. Each subgoal T generated by the rules of inference is tested to determine whether $M \models_E T$. If not, this subgoal is rejected. Notice that in our current notation Rule 2' of Section 2 should refer to $M \models_E K\sigma$. However, since $K\sigma$ contains no free variables, the distinction is a pedantic one.

3.2 Wffs in Skolem Form

One does not usually begin with wffs in quantifier-free form, but with first order wffs with quantifiers. The usual procedure is to convert these to quantifier-free form by introducing so-called Skolem functions. We briefly describe a procedure for doing this. Assume that W is a first order wff with quantifiers and no free variables. Assume further that no two occurrences of quantifiers in W quantify the same variable. Transform W into an equivalent wff W' as follows:

1. Eliminate all occurrences of $\supset$ through the transformation $P \supset Q \rightarrow \overline{P} \vee Q$
2. Distribute all negations across quantifiers via $\sim(x)Px \rightarrow (\exists x)\overline{P}x$ and $\sim(\exists x)Px \rightarrow (x)\overline{P}x$.
3. Distribute all negations across disjunctions and conjunctions via $\sim(P \vee Q) \rightarrow \overline{P} \wedge \overline{Q}$ and $\sim(P \wedge Q) \rightarrow \overline{P} \vee \overline{Q}$.

Suppose, in W', that (y) is a universal quantifier. Let $(\exists x_1)$, $(\exists x_2), \ldots, (\exists x_n)$ be all of the existential quantifiers whose scopes contain an occurrence of (y). Delete the quantifier (y) from W and replace each occurrence of y in W by $f(x_1, \ldots, x_n)$ where f is a new function symbol. f is called a Skolem function. Do this for each universal quantifier (y). Delete all the existential quantifiers in W. The resulting wff is quantifier-free, and said to be in Skolem form.

Example 3

W: (x) {(u) [(Pu,x $\wedge$ $\exists$ y ((z) Qy,z,u $\supset$ Rx,y)] $\supset$ $\exists$ w Sx,w}

W': (x) {$\exists$ u[$\overline{\text{Pu}}$,x V (y) ((z) Q y,z,u $\wedge$ $\overline{\text{R}}$ x, y)] V S x, w}

Skolem form of W

Pu, a $\wedge$ (Q f (u), g (u), u $\supset$ Ra, f (u)) $\supset$ S a, w

3.3 On Generality of Models

For our purposes, there are a number of deficiencies in the Tarskian definition of model just given. As we shall see, some of these can be resolved in a natural way, while others seem to require a different notion of model. All stem from the fact that our wffs are quantifier-free and the simple observation that not every quantifier-free wff is the Skolem form of some first order wff with quantifiers.

Consider first the theorem "In a group, if x^2=e for all group elements x, then the group is commutative". Formally,

$\vdash$ Group Axioms $\wedge$ x^2 = e $\supset$ ab = ba

An appropriate Tarskian model M would be a finite group with domain D for which d^2 = identity element of D, for each d ε D. In addition, a and b would each be bound to fixed elements d_a and d_b of D. In testing $M \models_E W(a,b)$ we would determine the truth value of $W(d_a,d_b)$. Evidently, there is a loss of generality here, since we really want a and b to independently range over D i.e. we would prefer $M \models_E W(a,b)$ iff for all $d_1,d_2 \varepsilon$ D $W(d_1,d_2)$ is true. Notice that we do not always want Skolem constants to range over the domain in establishing the truth value of a wff. Geometry is a case in point. However, for algebraic systems, it represents a more general test for truth. But now what are we to make of the following formulation of the same theorem?

$\vdash$ Group Axioms $\wedge$ x^2 = e $\wedge$ ab = c $\supset$ ba = c

Here, if M is a model of the antecedent, then in testing $M \models_E W(a,b,c)$ we

cannot permit a, b and c to independently range over D. a, b and c are constrained, by M, to assume values d_a, d_b, and d_c in D such that $d_a\, d_b = d_c$.

A natural solution seems to be to define a number of "parallel" Tarskian models, each with the same domain and multiplication table, and each with a different interpretation of the Skolem functions. In a 4 element domain this makes for 16 models $M_1,\ldots,M_{16}$. Then to test, say W(a, b, c) in the "general model", test each of $M_1 \models_E W(a, b, c),\ldots,$ $M_{16} \models_E W$ (a, b, c). Of course, these 16 Tarskian models collapse to a single non-Tarskian "general model" under appropriate book-keeping.

Unfortunately this approach - defining a number of parallel models, one for each set of interpretations of the function symbols - fails in general. To see why, consider a first order formula of the form $\exists x(y)W(x,y)$ with Skolem form W(x,f(x)), f being a new Skolem function. Let D be a finite domain and consider a number of parallel models each differing from the other only in the interpretation of f. Suppose there is one such model for each function f: D → D. Then testing the truth of W(x,f(x)) in this parallel model, is equivalent to evaluating the propositional form $\bigwedge_{f:D \to D} \bigvee_{x \in D} W(x,f(x))$. But this propositional form is equivalent to $\bigwedge_{y \in D} \bigvee_{x \in D} W(x,y)$. Evaluating this is equivalent to testing the truth of $(y)\exists xW(x,y)$ which is by no means the wff with which we began. Notice, however, that the original wff $\exists x(y)W(x,y)$ implies $(y)\exists xW(x,y)$, so that if the latter is false in a model, so also is the former. In general, one can show that if $Q_1Q_2\ldots Q_rW$ is a first order wff, where $Q_1,\ldots,Q_r$ are all of its quantifiers, then $Q_1Q_2\ldots Q_rW \supset Q_1'Q_2'\ldots Q_r'W$ is valid, (the reverse implication may not be) where $Q_1'Q_2'\ldots Q_r'$ is a rearrangement of the quantifier string $Q_1Q_2\ldots Q_r$ such that all of the universal quantifiers appear first.

One can also show that the use of parallel models in testing the truth of $Q_1Q_2...Q_rW$ is equivalent to testing the truth of $Q_1'Q_2'...Q_r'W$ as in the example just treated. In particular, if the Skolem form of $Q_1Q_2...Q_rW$ tests false in a parallel model, then $Q_1Q_2...Q_rW$ is false, but if it tests true, we know nothing about the truth value of $Q_1Q_2...Q_rW$. Fortunately, as we have seen for the Geometry Machine (Rule 2') and as we shall see for its generalization (Section 4), semantics affects the search-tree for a proof only when certain wffs are false in a model. It follows that we are justified in using parallel models as semantic aids in theorem-proving. What we thereby lose is generality, since $Q_1'Q_2'...Q_r'W$ is usually a much weaker assertion than $Q_1Q_2...Q_rW$.

A reasonable looking alternative follows from the observation that Skolem functions really mean "for all" so that, for example, W(x,f(x)) symbolizes $\exists x(y)W(x,y)$ and this compiles into the correct test $\bigvee_{x \in D} \bigwedge_{y \in D} W(x,y)$. Unfortunately, this approach fails in general because there are quantifier-free wffs which cannot be translated into first-order wffs with quantifiers. For example, if f is a Skolem function, then W(f(x), f(y)) is not the Skolem form of any quantified first-order wff. In many cases, such a translation is possible. [15] contains an appropriate algorithm.

There is a need for a semantic theory of quantifier-free wffs which provides the generality lacking in the Tarskian or "parallel model" approaches. The only other alternative is to reject the quantifier-free approach on which so many proof procedures (e.g. resolution [20], natural deduction [4]) depend - not a cheerful prospect. In any event, the remainder of this paper depends only on the availability of some well-defined notion of model. The notions which we have considered (Tarskian,

parallel) will certainly do, despite the fact that, for some wffs, they provide a weak semantics. The fact remains that a strong semantic theory for quantifier-free wffs would be welcome.

3.4 Wffs as Procedures

There are two ways in which a wff can be viewed: Either as a static statement which asserts something as true or false in a model, or as a dynamic procedure (program, algorithm) which, given a model M, returns the value true or false. In our discussions of semantics we shall take the latter point of view. Thus, in implementing the test $M \models_E W(x_1,\ldots,x_n)$, we imagine first compiling the expression $W(x_1,\ldots,x_n)$ into a procedure Π with free variables $x_1,\ldots,x_n$ and then evaluating Π on all n-tuples $(a_1,\ldots,a_n)$ of elements of D. This is particularly simple in a programming language like LISP, if we choose the syntax of wffs to be the same as LISP syntax, in which case Π is identical with W. One then simply EVAL's Π over the n-tuples of D. Although this point of view makes sense for both finite and infinite models, the procedural test for the truth of W is guaranteed to terminate only for finite models.

3.5 On Infinite Models

We have already remarked that plane geometry has no finite models. Certainly number theory has no finite model. In fact, most branches of mathematics suffer from (and are enriched by) this fact of life. Nevertheless mathematicians have a clear idea of the objects that they are proving theorems about. There is an underlying model which directs their proofs and suggests new theorems to be proved. How is such an infinite model internalized and how does it interact with the purely syntactic process of generating proofs?

We suggest that infinite models, as used by mathematicians, are not really models at all in the Tarskian sense. Rather, they consist,in part, of a finite set of procedures which map elements, or tuples of elements of the domain D (in the Tarskian sense) into D or {0,1}. The domain D cannot be explicitly specified (as can, say, the domain of a finite group) so we cannot hope to inspect it element by element. It is however implicitly specified as the domain of definition of the associated procedures, and as a data type. Hence, the natural numbers are implicitly specified as the domain of definition of the procedures ADD (.,.),LESS THAN (.,.),MULTIPLY (.,.), ADD1(.) etc. and by the type declaration INTEGER. The points of plane geometry can be specified implicitly as the domain of definition of the procedures BETWEEN (.,.,.), TRIANGLE (.,.,.), DISTANCE (.,.) etc. and by the type declaration (REAL,REAL).

One might be tempted to argue that there is something circular about the declaration of a data type as an approach to implicitly specifying an infinite domain, e.g. INTEGER for the natural numbers. However, the type INTEGER by no means tells you what a natural number is, but rather tells you how to represent natural numbers e.g. in decimal if you happen to be human, or binary if you are a computer. Specifying a representation is, of course, a finite matter. The procedures associated with the infinite model then act on the representation specified by the data type.

Let us be more precise. Suppose that D is the domain of an interpretation in the Tarskian sense. An intentional interpretation is specified by (with reference to the definitions at the beginning of this section)

1. a data type for the elements of D.
2. a finite (possibly empty) subset F of D.

3. for each n-ary predicate symbol P, a procedure P_I: $D^n \rightarrow \{0,1\}$.

4. for each n-ary function symbol f, a procedure f_I:$D^n \rightarrow D$.

Of course, the procedures in 3. and 4. compute on objects whose representation is specified by the data type of 1. F is intended to act as a finite "representative" of the domain. The elements of F are those elements of D which we choose to explicitly represent in the interpretation.

If $W(x_1,\ldots,x_n)$ is a wff, let $\Pi_I(x_1,\ldots,x_n)$ be its procedural representation under the intentional interpretation I. Then we shall write $I \models_E W(x_1,\ldots,x_n)$ iff there exists $a_1,\ldots,a_n \in D$ such that $\Pi_I(a_1,\ldots,a_n)$ returns 1. There is a similar definition for $I \models_U W(x_1,\ldots,x_n)$. The only difference between these definitions and those in the Tarskian sense is one of emphasis on the procedural approach.

If $I \models_U W$, then I is an <u>intensional model</u> for W. If D is finite and F = D, I is an <u>extensional model</u> for W. Obviously the question of the satisfiability of W in an extensional model is decidable by straightforward enumeration.

What have we gained by slightly rephrasing the Tarskian definitions at the beginning of this section? For one thing, we now have a precise way of finitely representing infinite models. This is so because mathematical theories have only finitely many predicate and function symbols, so only finitely many procedures are required. Another advantage is that these definitions provide a conceptual framework within which it is possible to interpret the informal notion of "model", the way in which these are manipulated in discovering a proof, and the sense in which we informally determine truth values of wffs. We illustrate with an example from geometry. In the Pons Asinorum, we are accustomed to saying that a diagram of an isosceles triangle ABC is a model for the theorem. But what does this statement really mean? Triangle ABC is certainly not a model in the Tarskian sense for the axioms of geometry. (The existence of mid-points of all line

segments precludes this.) We suggest that what is really meant is an intentional model like the following:

1. a data type (REAL,REAL) for the domain elements.
2. F = {(0,0), (0,1), (.5,2)} (or any other isosceles triangle in the Cartesian plane).
3. finitely many predicate procedures of the kind BETWEEN (.,.,.), PERPENDICULAR (.,.,.,.) etc.
4. finitely many functional procedures of the kind MIDPOINT (.,.), BISECTANGLE (.,.,.) etc. In addition the procedures A, B, C of no arguments which return (.5,2), (0,0), (0,1) respectively.

If, during the course of the proof, we bisect ∠BAC and extend this to meet BC in E, we shall have a new intensional model with F = {(0,0) , (0,1), (.5,2), (.5,0)} and the new procedure E of no arguments which returns (.5,0). E will be determined as INTERSECTION ((0,0), (1,0), (.5,2), BISECTANGLE ((0,0), (.5,2), (1,0))). The moral is that intensional models admit constructions. The role of F should by now be clear. F represents those objects of the domain D which we choose to represent explicitly. Constructions correspond to augmenting F by newly determined explicit elements of D. These new elements are computationally determined by means of the available procedures. Just what new elements we determine, and how the available procedures combine to compute them is determined <u>from the syntax of the proof of the theorem, as the proof is being discovered</u>. We shall return to this point in Section 4. For the moment it is sufficient to remark that a model should be used to guide the proof, and conversely, the current state of the incomplete proof should suggest a model. The nature of this suggested model is determined by the initial model, and the syntactic form of the current subgoal of the proof. During the course of

discovering a proof, there should be continuous interplay between the syntax and semantics, each suggesting to the other how next to proceed.

In determining the truth value of a wff $W(x_1,\ldots,x_n)$ in a model we are free to use whatever "real world" knowledge that we have about the model. In geometry we tend to "eye" the diagram, mentally measuring distances and angles and moving points to test whether there exist points $x_1,\ldots,x_n$ so as to make $W(x_1,\ldots,x_n)$ true. A more careful student might try out various configurations using a ruler and compass for greater precision. The computer implemented analogue to this visual process would be Cartesian geometry. Since points are thereby represented as co-ordinates in the plane, line segments as linear equations, and length as the usual Euclidean norm, the wff $W(x_1,\ldots,x_n)$ represents a system Q of linear and quadratic equalities and inequalities, and it is straightforward to extract Q from W. Then $M \models_E W(x_1,\ldots,x_n)$ iff there is a solution to the system Q. It is tempting to argue that this use of Cartesian geometry is a cheat since the Cartesian approach is merely an analytic formulation of the Euclidean approach. Indeed, is there not something circular about all of this? The answer is no, since that knowledge which we bring to bear in the semantics is never appealed to in the syntactic proof which we are constructing. As we have seen for the Geometry Machine, semantics is used only to prune the search tree for the syntactic proof; it never intrudes as part of the proof. In fact, from these observations we can posit an even more "paradoxical" use of semantics. Suppose that the current theorem being proved is Theorem 47, and that, while browsing through the back of the book we discover Theorem 93 which, moreover, tells us something about the model we are using in attempting a proof of Theorem 47. Even if Theorem 93 depends for its proof on Theorem 47, we are justified in using its statement as semantic information to help us prove Theorem 47. There are clearly some ethical and pedagogical

problems here, but the logic is sound.

Most of our discussion of intensional models has centered on geometry. There is good reason for this - geometry seems to be the best example of a mathematical theory for which we have a vast store of "real world" knowledge about its principal models. As human theorem-provers, we have very good visual and analytic procedures for testing the satisfiability of wffs in these models. Moreover, this facility is more or less independent of the number, or nature, of the theorems we have proved. Neither of these virtues is enjoyed by, say, number theory, even if we ignore the induction axiom. The difficulty is that we have no uniform procedure for testing satisfiability over the natural numbers, as we do for geometry or for extensional models. In testing W(x), the best we can do in general is to try W(0), W(1),..., etc. in the hope of finding a natural number n such that W(n) is true, in which case we know that W(x) is satisfiable. There is no way that this process can determine that W(x) is false on the natural numbers. Unfortunately, proof trees are pruned only by encountering wffs which are false in the models. (See Rule 2' of L_G, and the system L of the next section.) Nevertheless, we believe that a reasonable theory of counter-examples is possible for number theory. Of course, not every false formula admits a counter-example e.g. formulae of the form $\exists x W(x)$, but many do e.g. $(x)W(x)$. It is intuitively clear that mathematicians combine the search for counterexamples to current sub-goals with the search for a proof. Counterexamples are semantic notions. What is needed for a semantic approach to number theory is a collection of (possibly ad hoc) techniques for discovering counterexamples to be used as a proof-tree pruning device, just as analytic techniques are used in geometry. To our knowledge, no work has been done on this problem, but we believe its solution to be a necessary

prerequisite for automating number theory.

What we hope has emerged from this discussion is the fundamental role played by knowledge about the models used in mathematics. The more one knows about the semantics, the easier it is to find a proof. (Compare the success of L_G with the performance of its purely syntactic component). We believe that geometry is easier than number theory precisely because we know much more about diagrams than we do about natural numbers. Assuming the legitimacy of this point of view, it becomes necessary to look much more closely at models for other branches of mathematics with the objective of finding procedures which can detect whether a given wff is false in that model.

4. A Generalized Semantic and Deductive System

This section is devoted to generalizing the system L_G of Gelernter et al. and illustrating a number of its features. The resulting system L will be seen to possess the properties promised at the end of Section 2.6.

4.1 The Deductive and Semantic System L

	Input	Output
1.	H; R $\vdash A \supset B$	
	If A and B are identical	T
	If $A\sigma = B\sigma$	σ
2.	H; R $\vdash A \wedge B$	
	If H ; R $\vdash A$ yields output σ_1 and a model M of H can be found such that $M \models_E A\sigma_1$ and if σ is a substitution such that $M \models_E B\sigma_1\sigma$ and if H ; R $\vdash B\sigma_1\sigma$ yields output σ_2	$\sigma_1\sigma\sigma_2$
3.	H ; R $\vdash A \vee B$	
	If a model M of H can be found such that $M \not\models_E A$	H ; R $\vdash B$
	If a model M of H can be found such that $M \not\models_E B$	H ; R $\vdash A$
	Otherwise	H ; R $\vdash A$ or H ; R $\vdash B$

4. $H ; R \vdash (A \supset B) \supset C$

   If $H ; R \vdash B \supset C$ yields output

   σ_1 and a model M of H can be found

   such that $M \models_E (B \supset C)\sigma_1$ and if

   σ is a substitution such that

   $M \models_E A\sigma_1\sigma$ and if $H ; \phi \vdash R \supset A\sigma_1\sigma$

   yields output σ_2 $\qquad\qquad$ $\sigma_1\sigma\sigma_2$

5. $H ; R \vdash A \supset (B \supset C)$ $\qquad\qquad$ $H ; R \vdash A \wedge B \supset C$

6. $H ; R \vdash A \vee B \supset C$

   If $H ; R \vdash A \supset C$ yields output σ_1

   and a model M of H can be found

   such that $M \models_E A\sigma_1 \supset C$ and if σ is

   a substitution such that

   $M \models_E B\sigma_1\sigma \supset C$ and if $H ; R \vdash B\sigma_1\sigma \supset C$

   yields output σ_2 $\qquad\qquad$ $\sigma_1\sigma\sigma_2$

7. $H ; R \vdash A \supset B \vee C$

   If a model M of H can be found

   such that $M \not\models_E A \supset B$ $\qquad\qquad$ $H ; R \vdash A \supset C$

   If a model M of H can be found

   such that $M \not\models_E A \supset C$ $\qquad\qquad$ $H ; R \vdash A \supset B$

   Otherwise $\qquad\qquad$ $H ; R \vdash A \supset B$ or

   $H ; R \vdash A \supset C$

8. $H ; R \vdash A \supset B \wedge C$

   If $H ; R \vdash A \supset B$ yields output

   σ_1 and a model M of H can be found

   such that $M \models_E A \supset B\sigma_1$ and if σ is

   a substitution such that $M \models_E A \supset C\sigma_1\sigma$

   and if $H ; R \vdash A \supset C\sigma_1\sigma$ yields output σ_2 $\qquad$ $\sigma_1\sigma\sigma_2$

9.	$H ; R \vdash A \wedge B \supset C$		$H \wedge A ; R \wedge A \vdash B \supset C$
		or	$H \wedge B ; R \wedge B \vdash A \supset C$
10.	$H ; R \vdash C \supset \overline{A} \vee B$		$H ; R \vdash C \wedge A \supset B$
11.	$H ; R \vdash \overline{A} \wedge B \supset C$		$H ; R \vdash B \supset A \vee C$
12.	$H ; R \vdash \overline{B} \supset C$		$H ; R \vdash B \vee C$
13.	$H ; R \vdash \dot{B} \supset \overline{C}$		$H ; R \vdash B \wedge C \supset NIL$
14.	$H ; R \vdash s = t \supset A$ If a model M of $H \wedge s = t$ can be found such that $M \models_E A'$ and if σ is a substitution such that $M \models_E A'\sigma$		$H \wedge s = t ; \phi \vdash R' \supset A'\sigma$ where R', A' are obtained from R,A by substituting t for s in R,A respectively.
15.	$H ; R \vdash A \supset s = t$ If there exists a substitution σ such that $s\sigma$ and $t\sigma$ are identical		σ

4.2 Remarks

1. The purely deductive aspect of L is due to Bledsoe, Boyer, and Henneman [4]. As they themselves remark, it is an incomplete system of natural deduction. As such, it stands in contrast to resolution theorem-proving techniques. Aside from its incompleteness, it is much closer to the deductive systems used by Wang [25]. Its only point of contact with resolution is through the use of the unification algorithm of Robinson [20] in returning most general unifiers σ.

2. A, B and C are Skolemized wffs, and s and t are terms. R is called the reserve by Bledsoe et al. The substitution σ of Rules 1 and 15 are obtained by the Unification Algorithm and hence are most general unifiers. The substitution σ of Rules 2, 4, 6, 8 and 14 is arbitrary and may be null. We explain its role in 5. below.

3. The top-level subgoal is a Skolemized wff of the form H ; $\phi \vdash H_1 \wedge \ldots \wedge H_n \supset W$ where $H_1,\ldots,H_n$ are the axioms, previously proved theorems, definitions, and special hypotheses of the current theorem, R is the null conjunct ϕ, and $H = H_1 \wedge \ldots \wedge H_n$. The rules are applied, in order, to any current subgoal. If any rule fails, the next rule is attempted. If all fail on the current subgoal, NIL is returned.

4. The models M can assume any of the forms discussed in Section 3. Thus M may denote a single model (one assignment to the Skolem functions) or several "parallel" models (several assignments to the Skolem functions). M may be extensional or intensional.

5. The substitution σ of Rules 2, 4, 6, 8 and 14 allows for "good guesses" to be made by the theorem-prover, based upon observations made in the model, as to what the occurring free variables actually represent. Thus if $M \models_E W(x_1,\ldots,x_n)$ and, moreover, if there exist unique objects $a_1,\ldots,a_n$ in the domain of M such that $W(a_1,\ldots,a_n)$ is true in M and if these $a_1,\ldots,a_n$ can be interpreted in the syntax as terms $t_1,\ldots,t_n$ involving only Skolem functions, then it would be an excellent guess to attempt, as the current syntactic goal, $\vdash W(t_1,\ldots,t_n)$ rather than $\vdash W(x_1,\ldots,x_n)$. Clearly the first goal will, in general, be much easier to prove than the second. Even when the a_i are not unique, there may be additional semantic information available with which to make a plausible guess, or several guesses which are pursued in parallel. We believe that this notion

captures a significant way in which humans use models for discovering proofs. A good σ-guesser would represent a powerful use of the model.

6. Suppose Rule 8 applies, and succeeds on its first AND-subgoal by returning σ_1. If an appropriate model M of H can be found such that $M \not\models_E A \supset C\sigma_1$, or if $H\ ;\ R \vdash A \supset C\sigma_1\sigma$ returns NIL, then we back up to the first AND-subgoal, attempt to have it succeed by returning a different substitution σ_1', and try to establish the second AND-subgoal with σ_1'. Without the use of M and σ, this corresponds to the back-up procedure used by Bledsoe et al. A similar technique is used with Rules 2, 4 and 6. This need for back-up is a serious computational limitation of Bledsoe's rules of inference; it also occurs in disguised form in resolution based systems of deduction.

In effect therefore, we are providing for the use of counterexamples to prune the search tree. If the test $M \not\models_E A \supset C\sigma_1$ succeeds, M is a counterexample to the subgoal $H\ ;\ R \vdash A \supset C\sigma_1$ so there is no sense in pursuing the second AND-subgoal. We feel that this use of models as "semantic seives" in trapping out incorrect substitutions σ_1 (for example, in assuring that there is an appropriate model M such that $M \models_E A \supset C\sigma_1\sigma$ and, moreover, that no counterexample has been found, before embarking upon the goal $H\ ;\ R \vdash A \supset C\sigma_1\sigma$) will tend to minimize back-up in the search for proofs. As we shall see in Section 5, there is an interesting analogy between this use of models in theorem-proving and the use of "real world knowledge" in parsing sentences in natural language.

7. The remaining rules which make use of semantics are 3 and 7. Without the intervening model, these would each generate two OR-subgoals. With the model, we can hope to prune away one of these subgoals.

8. The semantic rules all state "If a model M ... can be found such that ...". We have chosen this mode of expression to emphasize the dynamic nature of models as used by humans in discovering proofs. Thus, the model with which one begins a proof need not remain the same throughout the course of the proof. For example constructions may be made as in geometry. Or a case analysis may require a different model for each case. Because this dynamic facility appears to be central to human proof discovery, we have explicitly allowed for it in the system L.

The remainder of this section is devoted to examples which illustrate the features just described. For economy, we shall often suppress explicit references to R and H.

4.3 Examples

Example 4

We first give an example of a proof in propositional logic. The top level goal is

$H\ ;\ \phi \vdash (A \supset B) \wedge ((A \supset B) \supset (C \supset B)) \wedge (D \supset B) \supset (C \supset D \vee B)$

with $H = (A \supset B) \wedge ((A \supset B) \supset (C \supset B)) \wedge (D \supset B)$

Rule 5 applies.

1. $H\ ;\ \phi \vdash (A \supset B) \wedge ((A \supset B) \supset (C \supset B)) \wedge (D \supset B) \wedge C \supset D \vee B$

Rule 7 applies. With $M = \{A, B, C, \overline{D}\}$ we semantically eliminate one of the OR-subgoals, leaving

11. $H\ ;\ \phi \vdash (A \supset B) \wedge ((A \supset B) \supset (C \supset B)) \wedge (D \supset B) \wedge C \supset B$

Rule 9 applies.

111. $H \wedge (D \supset B)\ ;\ (A \supset B) \wedge ((A \supset B) \supset (C \supset B)) \wedge (D \supset B) \vdash C \supset B$

or 112. $H \wedge C\ ;\ C \vdash (A \supset B) \wedge ((A \supset B) \supset (C \supset B)) \wedge (D \supset B) \supset B$

111. fails syntactically. Pursue 112. Rule 9 applies.

1121. $H \wedge C\ ;\ C \wedge (A \supset B) \wedge ((A \supset B) \supset (C \supset B)) \vdash (D \supset B) \supset B$

or 1122. $H \wedge C \wedge (D \supset B)$; $C \wedge (D \supset B) \vdash (A \supset B) \wedge ((A \supset B) \supset (C \supset B)) \supset B$

Pursue 1121. Rule 4 applies, but with $M = \{A, B, C, \overline{D}\}$ its second AND-subgoal fails. Hence, pursue 1122. Rule 9 applies.

11221. $H \wedge C \wedge (D \supset B)$; $C \wedge (D \supset B) \wedge ((A \supset B) \supset (C \supset B)) \vdash (A \supset B) \supset B$

or 11222. $H \wedge C \wedge (D \supset B)$; $C \wedge (D \supset B) \wedge (A \supset B) \vdash ((A \supset B) \supset (C \supset B)) \supset B$

Pursue 11221. Rule 4 applies, but with $M = \{\overline{A}, B, C, \overline{D}\}$ its second AND-subgoal fails. Hence pursue 11222, which is easily established syntactically.

Notice that the semantic proof uses two models, and that these are dynamically determined, as the proof unfolds.

Example 5

This example illustrates how a diagram in geometry rejects an application of Rule 4. The theorem is

"If ΔABC has equal base angles, then AB=AC".

Assume that one of the axioms present is

α : square x y z w $\supset$ xy = yz

The theorem is

$\mathcal{A} \wedge \Delta ABC \wedge \angle ABC = \angle ACB \supset AB = AC$

where $\mathcal{A}$ is a conjunct of all the axioms. Several applications of Rule 9 will yield as one of its OR-subgoals.

1. $\vdash \alpha \supset AB = AC$

Rule 4 applies yielding the second AND-subgoal

$\vdash$ square ABC w

which is clearly false in any right angle free diagram of an isosceles triangle.

Notice that the Geometry Machine would also have rejected this application of Rule 4, but for the "wrong" reason. It would have tried the three possible substitutions of the given points A, B, and C for w, and found that none succeeded, whence the goal 1. would be rejected. On the other hand, we reject 1. because nowhere in the plane is there a suitable point w.

Example 6

This is a geometry example which illustrates the use of semantics in rejecting a subgoal generated by Rule 8. The theorem states that for each triangle there is a point equidistant from the three vertices. We assume that, among the axioms present is

α : f(u,v)u = f(u,v)v (Each line segment uv has a midpoint f(u,v))

The theorem is

$\mathcal{A} \wedge \Delta$ ABC $\supset$ xB=xC $\wedge$ xA = xC

where $\mathcal{A}$ is a conjunct of all the axioms. Rule 8 yields the first AND-subgoal

$\vdash \mathcal{A} \wedge \Delta$ ABC $\supset$ xB = xC

Several applications of Rule 9 will yield as one of its OR-subgoals

1. $\vdash \alpha \supset$ xB = xC

which succeeds with f(B,C) | x. This backs up to the application of Rule 8, yielding the second AND-subgoal

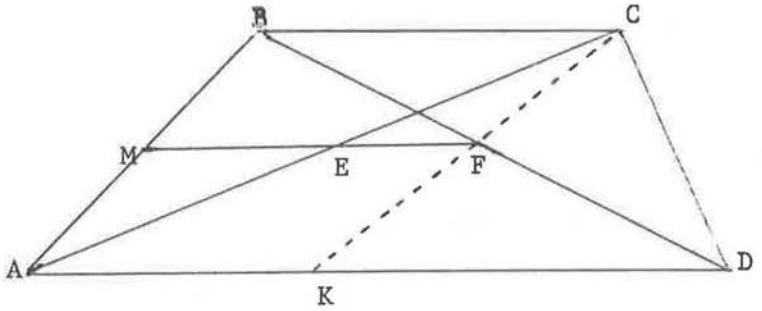
2. $\vdash \mathcal{A} \wedge \Delta$ ABC $\supset$ f(B,C)A = f(B,C)C

which is false in any "random" diagram of a triangle ABC. Hence, the OR-subgoal 1. is pruned from the search tree and the proof continues using the remaining OR-subgoals generated by Rule 9. The advantage here is not so much that 1. is pruned, but the time saved in not attempting a proof of the invalid goal 2.

Example 7

This example is drawn from Gelernter et al. [11]. It is of interest because its proof requires a subtle construction. We sketch a portion of the proof, using the system L, which would lead to the necessary construction being made as the proof unfolds. The theorem states: If ABCD is a trapezoid with BC||AD, and the line joining the midpoint E of AC to the midpoint F of BD meets AB at M, then MA = MB.

The initial model (without the point K) is



The crux of the proof is to prove that EF||AD and then, since, in ΔBAD, FB=FD and MF||AD, then MB=MA. To prove EF||AD, the line segment CF must be extended to meet AD in K. (The Geometry Machine was unable to discover this construction, and had to be given this hint before it found a proof). Then it must be established that FC=FK whence, since EC=EA, EF||AK. We shall indicate how the subgoal EF||AD leads to the subgoal FC=FK by forcing the construction of the line segment CF and its extension to M. We assume the presence (among others) of the two axioms

α: xy || uv ∧ coll u v w ⊃ xy || uw

β: Δ x y z ∧ coll x u y ∧ coll x w z ∧ ux = uy ∧ wx = wz ⊃ uw || yz

Assuming that EF||AD is to be proved, one of the subgoals generated would be

⊢ α ⊃ EF||AD

Rule 4 applies

1. $\vdash$ xy||u w $\supset$ EF||AD

This yields σ_1 = {E|x, F|y, A|u, D|w} . Since M$\models_E$ EF||Av $\wedge$ coll A v D (any v on AD will do) proceed with

2. $\vdash$ R $\supset$ EF||Av $\wedge$ coll A v D

Rule 8. applies. The first AND-subgoal is

21. $\vdash$ R $\supset$ EF||Av

Embedded in R will be the axiom β so that after a number of applications of Rule 9, we will obtain as an OR-subgoal

211. $\vdash$ β $\supset$ EF||Av

Rule 4 applies.

2111. $\vdash$ uw||yz $\supset$ EF||Av

This yields σ_1 = {E|u, F|w, A|y, v|z}

Now M$\models_E$ Δ xAv $\wedge$ coll xEA $\wedge$ coll xFv $\wedge$ Ex = EA $\wedge$ Fx = Fv

In fact, there are <u>unique</u> satisfying points x and v, namely x = C and v = K. Hence, take as the σ of Rule 4, σ = {C|x, K|v}. This yields a new model containing the new line segment CFK. The second AND-subgoal by Rule 4 is thus

2112. $\vdash\Delta$ CAF $\wedge$ coll CEA $\wedge$ coll CFK $\wedge$ EC = EA $\wedge$ FC = FK

The first four literals in this conjunct are easily established, leaving the goal $\vdash$FC = FK which is as far as we wanted to carry the proof. There still remains to be established the second AND-subgoal of Rule 8 at 2. This yields, backing up the substitution K|v

22. $\vdash$ coll AKD

This goal is clearly true in the new model, and is easily established.

Notice the use, at 2111. of the substitution σ. This is a good example of the use of a model in making "premature instances" of variables, and illustrates why Rules 2, 4, 6, 8 and 14 make provision via σ for

such "good guesses". Again, the example illustrates how the system L permits a model to change during the course of a proof.

Example 8

This example illustrates the use of semantics in disambiguating an incorrect backed-up substitution. The theorem states

"If S is a non-empty subset of a group such that $xy^{-1} \varepsilon S$ whenever x and $y \varepsilon S$, then $x^{-1} \varepsilon S$ whenever $x \varepsilon S$."

$\vdash ex = x \wedge xe = x \wedge xI(x) = e \wedge I(x)x = e \wedge Sb \wedge (Sx \wedge Sy \wedge xI(y)=z \supset Sz) \supset SI(b)$

An OR-subgoal due to Rule 9 is

1. $\vdash (Sx \wedge Sy \wedge xI(y) = z \supset Sz) \supset SI(b)$

Rule 4 applies and yields as its second AND-subgoal

11. $\vdash R \supset Sx \wedge Sy \wedge xI(y) = I(b)$

which is semantically valid. Rule 8 yields as its first AND-subgoal

111. $\vdash R \supset Sx$ which will succeed with $b|x$.

This backs up to the second AND-subgoal of Rule 8

112. $\vdash R \supset Sy \wedge bI(y) = I(b)$

This subgoal may or may not be semantically valid depending upon the model being used. Let us assume this failure is not detected and proceed. It is clear that another application of Rule 8 will yield $b|y$ from its first AND-subgoal, leaving as its second AND-subgoal

$\vdash R \supset bI(b) = I(b)$ which is clearly false in any model in which b is not assigned the group identity. This failure backs up to 11. This time, take as the first AND-subgoal

111. $\vdash R \supset xI(y) = I(b)$

which succeeds with $e|x$, $b|y$. The second AND-subgoal is therefore

112. $\vdash R \supset Se \wedge Sb$

which is semantically valid and easily proved.

Notice that a clever theorem-prover would have observed that, in M, {e|x, b|y} and {I(b)|x,e|y} are the only two obvious "solutions" to 11. Hence, it would have returned the two "guesses" σ = {e|x,b|y} and σ' = {I(b)|x,e|y} in its application of Rule 4 to 1. σ' leads to the subgoal

$\vdash R \supset SI(b) \wedge Se \wedge I(b)I(e) = I(b)$

which is rejected because it is subsumed by the top level goal. σ leads to the subgoal

11. $\vdash R \supset Se \wedge Sb \wedge eI(b) = I(b)$

which has a trivial, back-up-free proof.

5. Remarks

5.1 Some Linguistic Analogies

We wish to compare our approach to theorem-proving with Winograd's to natural language processing [26], in part because we have been greatly influenced by his conceptual framework, in part because a number of striking analogies emerge from this exercise. We shall assume that the reader is familiar with Winograd's basic results.

The universe of discourse for Winograd's system is the BLOCKS world, consisting of a set of basic relations like (#1S :B1 #BLOCK), (#COLOUR :B7 #GREEN) together with a set of PLANNER programs such as TC-ON which describe higher order semantic relations in terms of the basic relations, or such as TC-MAKESPACE which manipulate and alter the BLOCKS universe. The theorem-proving analogue is, of course, a model in which the basic relations are things like (CO-ORD A (.5, 2)), or the multiplication table for a group, while the higher order semantic relations are procedures like ISPRIME (.), QUADRILATERAL (.,.,.,.) etc. and the procedures for manipulating and altering the model correspond to constructions like BISECTANGLE (. , . , .), FINDNTHPRIME (.) etc.

Under Winograd's procedural approach, natural language sentences compile into PLANNER procedures which are then executed either for their effect on the BLOCKS world (when the input sentence is a command or represents new knowledge) or to return a value (when the input is a question). Under our approach to theorem-proving, wffs also compile into procedures which affect the model (in those cases when a construction is suggested) and which return a value (NIL or T together with the values in the model of the wff's free variables). Of course, there is no comparing Winograd's compilation process with ours - his requires an extremely involved

synthesis of parsing techniques and special semantic routines while ours, because of the simple unambiguous syntax and semantics of wffs, is essentially trivial - but the resulting procedures perform analogous tasks on their respective models.

Is there a theorem - proving analogue to the linguistic process of parsing a sentence? We think that, in many ways, the proof of a wff corresponds to the parse of a sentence. We shall try to clarify this idea with reference to the purely syntactic component of the system L. First, the rules of inference can be viewed as parsing rules; provided the input wff has constituents separated by appropriate logical connectives, a corresponding rule can be applied to yield output wff(s) which in turn must be "parsed". Secondly, there is a notion of syntactic ambiguity, since some of the parsing rules are themselves ambiguous (e.g. $(A \supset B) \supset (C \supset D)$ can be parsed by both Rules 4 and 5). Finally, provided the logical system is complete (L is not) and the wff is valid (grammatical?), a successful parse is assured. To be sure, there are several linguistic concepts on which this analogy crumbles. For one thing, if the wff is not valid (ungrammatical?) the parser cannot in general detect this. Also, such linguistic notions as deep structure, grammatical categories etc. have no obvious analogues in theorem-proving. Nevertheless, let us further pursue the analogy. We shall say that a wff is <u>ambiguous</u> if it has at least two proofs each of which returns a <u>different</u> substitution at the top level. Thus propositional tautologies are unambiguous, $Pa \supset Px$ is unambiguous, $Pa \wedge Pb \supset Px$ is ambiguous. Notice that this is a notion of syntactic ambiguity. There seems to be no reasonable notion of semantic ambiguity in theorem-proving.

Now consider, for example, Rule 2 of L and suppose that H; $R \vdash A$ has been successfully "parsed" returning substitution σ_1. We want, at this point, to determine whether or not this parse makes sense in the current context before proceeding with the parse of B, ie. we make the semantic test $M \models_E B\sigma_1$. If this test fails, a different parse of the ambiguous wff A is attempted. Similar remarks apply to Rules 4, 6 and 8. In effect, we are semantically disambiguating the parse as we parse. This process of continuous semantic disambiguation for guiding the parse is the same principle appealed to by Winograd for natural language parsing. Consider, as an example, the sentence "Carry the block in the box". During the attempted parse, the parser might consider treating the constituent "the block in the box" as a noun group. But in the BLOCKS world, if there is no block in the box this constituent should be rejected, just as the parse of A is rejected by the model M if $M \not\models_E B\sigma_1$. In both cases, "real world" information is used to reject contextually meaningless parses.

In 5.2 below we shall further pursue these analogies by arguing that adding new knowledge is a problem common to both natural language and theorem-proving systems.

5.2 On the Role of Theorem-Proving in Artificial Intelligence

In the past researchers in Artificial Intelligence have tended to view theorem-proving as a tool for the solution of problems arising in areas other than mathematics e.g. robot problem-solving [7], natural language processing [24] etc. The general point of view seems to be: "If you can axiomatize your problem domain, do so, then wait until the theorem-proving people have perfected their craft". Whatever the merits of this approach, we suggest that research into automatic theorem-proving has another, perhaps more fundamental role to play - that of dealing directly with problems common to many areas of Artificial Intelligence. Moreover,

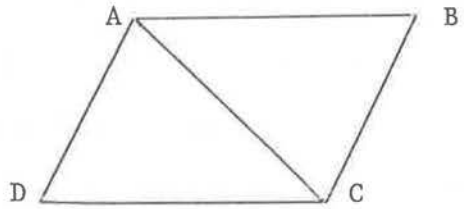
this role is a realistic one provided a model theoretic view of theorem-proving is adopted. Not only is the concept of a model almost universal in Artificial Intelligence, but also many of the problems associated with models in general arise in the context of theorem-proving, and must be solved if one hopes to do serious mathematics. Some examples:

Generalization and Hypothesis Formation

How do we generalize observations made in a model? For example, if a given geometric diagram is observed to possess some interesting property, how is this observation translated into an appropriate and "most general" provable wff? This problem, for mathematics, has as analogue the task of generalizing real world experience, which arises for example in robot research [19]. There is a fortunate difference, however. In mathematics the criterion for a correct generalization is well defined - the generalization must be provable. Criteria for interesting generalizations are much more problematic.

The Description Problem

A moment's reflection on the generalization problem for geometry leads to the conclusion that a prior concern is that of recognizing, and describing the relationships among, the various components of the diagram. Do we describe the following figure



as PARGM A,B,C,D $\wedge$ DIAG A,C

or Δ ABC $\wedge\Delta$ ACD $\wedge$ AB $||$CD $\wedge$AD$||$ BC

or ΔABC $\wedge\Delta$ ACD $\wedge$ Congruent ABC, ACD?

The Representation of New Knowledge

A theorem, once proved, is not superfluous information. On the contrary, it represents new knowledge about the problem domain. As such, it should not merely be added to the old stock of axioms and theorems to be treated as just another wff, but should compile into a semantic procedure which interacts with the available semantic procedures. Typically, this is desirable when the theorem says something about the combinatorics of semantic search procedures e.g. that an object with such and such properties can be found in a certain subset of the full domain. Such information can be very useful in testing the satisfiability of wffs in the model, and hence should be absorbed into the semantics. Similarly, if a theorem is an existential statement, its proof yields up a constructive definition of the variable(s) asserted to exist. This construction should become available in the semantics as a procedure for possible future use. Finally, new definitions given syntactically ought to compile into semantic procedures.

This is an old problem in a new guise. It occurs in information retrieval problems, as well as in natural language processing. The parallel is particularly striking if one compares the theorem-proving analogue with Winograd's system for natural language [26]. The semantics of the BLOCKS world consists of a set of mutually interacting procedures. Winograd remarks that new knowledge, input as declarative sentences (theorems?), must translate into new BLOCKS procedures which interact with the old. Clearly, theorem-provers will be faced with precisely the same problem.

Granted that theorem-proving research must confront many problems of general concern to the Artificial Intelligence community, what is the advantage of formulating and investigating these problems in a theorem-

proving context? We feel that the principal advantage derives from the non trivial, well defined nature of the problem domain. For mathematics we have a precise idea of the objects we are dealing with. There is a well defined syntax for the language (wffs), we know exactly what wffs talk about (a model), and we know what truth means. In other words, the "real world" is relatively simple and precisely specifiable, as is the language for talking about this world. Within this framework, we can investigate problems with wider import in Artificial Intelligence, like hypothesis formation and the representation of new knowledge. Moreover, this can be done without obscuring the real issues by seemingly open-ended considerations involving natural language, and the representation of a highly complex domain of discourse. Another advantage is that in theorem-proving the pragmatics of the situation is reasonably clear - a test of a theory of hypothesis formation for example, is how well it yields up theorems. Finally, since doing mathematics is by no means trivial, we necessarily avoid the "toy problem trap".

It seems reasonable to expect that solutions to these problems in mathematics will provide considerable insight into their real world analogues. This being the case, we can expect a change of emphasis in the future from theorem-proving as a tool for Artificial Intelligence, to theorem-proving as a problem environment about which interesting, and important general questions may be asked and investigated.

6. Conclusion

We have argued in favour of the use of models in theorem-proving. In particular, we have emphasized the importance of a smooth, natural interface between syntax and semantics, the way in which the current state of the proof suggests the form of the current model, and conversely how the model should be used in guiding the proof. We have indicated how inferences can be made in the model, and how observations in the model can be used to suggest inferences at the syntactic level. The semantic and syntactic system L has been used as the principal vehicle for conveying these ideas.

6.1 Some Suggestions for Further Research

A number of open research problems have been mentioned throughout this paper. We briefly summarize these now.

1. An adequate model theory for quantifier-free wffs.
2. The development of specialized domain dependent techniques for finding counterexamples.
3. The representation of new knowledge.
4. The problem of generalizing observations made in a model.
5. The description problem.

In addition, a number of other research areas immediately suggest themselves.

1. Interactive Theorem-Proving

The system L (or any other like it) is very "human-oriented" and hence represents an ideal vehicle for interactive computing. In particular, the user would not require specialized knowledge about available proof and editing strategies nor would he be required to learn an unnatural

deductive system. (cf. the interactive system of [1] which suffers from both of these deficiencies.) It is easy to imagine a system based upon the ideas of L in which the computer requests of the user new models, his guess of a σ, his opinion of an attempted application of Rule 4 for backward chaining, counterexamples, his opinion of the plausibility of a current subgoal, etc. Particularly appealing is the short training period that would be required of a student or mathematician before he could sit at a console to play with the system.

2. The Automatic Generation of Models

In some sense, presenting the system L with an initial model is a cheat. It would be preferable to have techniques which, when presented with the special hypotheses of the theorem to be proved, generate an appropriate model. This is by no means a trivial problem - indeed, the general problem is unsolvable. It can be argued that we are replacing one difficult problem (syntactic theorem-proving) by another (model discovery), in which case, what have we gained? That remains to be seen and depends largely upon a successful solution to the model discovery problem. Nevertheless, we believe that this is where many of the real problems of theorem-proving arise - in the semantics.

3. PLANNER - Type Recommendations

PLANNER [14] is a language specifically designed by Hewitt to do theorem-proving. Among its many features is the ability to program in recommendations as to what theorem(s) to use in order to establish a certain subgoal. For example, it is easy to express, as a PLANNER procedure, a recommendation like "In order to prove that line segment xy is parallel to uv, first try to prove that length xy = length uv in which case try

to prove parallelogram xyvu. If this fails, try to find a point w such that Δwxy and such that u = midpoint (w,x) and v = midpoint (w,y). If this fails, enter a blind search." Virtually all introductory geometry texts are full of friendly advice just like this, and every mathematician has a bag full of such tricks and special techniques. We would argue against exclusive reliance on such recommendations in theorem-proving, since these fall under the category of domain dependent heuristics which were discussed in Section 1. It would be equally foolish to ignore the important role they play. The integration into the system L of facilities for making recommendations in a transparent and flexible way is therefore an important problem.

4. Extending the System L

It should be emphasized that what we have presented in this paper is a paradigm for theorem-proving rather than a polished system. We are in favour of some kind of system of natural deduction - L happens to be one such - but L is incomplete. Unlike many other workers in this area, we feel that the underlying logic must be complete. After all, no mathematician would deny himself the full deductive power of first order logic. We have chosen to deal with the system L because of its simplicity and naturalness, and because its syntactic component has been implemented and else-where described [4]. Nevertheless, L should be extended to a complete semantic and syntactic system of natural deduction.

Acknowledgment

This research was supported by National Research Council of Canada grant A-7642.

References

[1] Allen, J. and Luckham, D. An interactive theorem-proving program. Machine Intelligence 5, (eds. Meltzer, B. and Michie, D.). New York: American Elsevier (1970), 321-336.

[2] Andrews, P.B. Resolution with merging. J.ACM, 15 (1968), 367-381.

[3] Bledsoe, W.W. Splitting and reduction heuristics in automatic theorem proving. Artificial Intelligence, 2 (1971), 55-77.

[4] Bledsoe, W.W., Boyer, R.S. and Henneman, W.H. Computer proofs of limit theorems. Artificial Intelligence, 3 (1972), 27-60.

[5] Darlington, J.L. Theorem proving and information retrieval. Machine Intelligence 4, (eds. Meltzer, B. and Michie, D.). Edinburgh: Edinburgh University Press (1969), 173-181.

[6] Davis, M. and Putnam, H. A computing procedure for quantification theory. J.ACM, 7 (July 1960), 201-215.

[7] Fikes, R.E. and Nilsson, N.J. STRIPS - A new approach to the application of theorem proving to problem solving. Artificial Intelligence, 2 (1971), 189-208.

[8] Gelernter, H. Realization of a geometry theorem-proving machine. Computers and Thought (eds. Feigenbaum E.A. and Feldman J.) New York: McGraw Hill (1963), 134-152.

[9] Gelernter, H. Machine generated problem solving graphs. Proc. Symp. Math. Theory of Automata, Brooklyn: Polytechnic Press (1963), 179-203.

[10] Gelernter, H. A note on syntactic symmetry and the manipulation of formal systems by machine, Information and Control 2, (1959), 80-89.

[11] Gelernter, H., Hansen, J.R., and Loveland, D.W. Empirical explorations of the geometry theorem machine. Computers and Thought (eds. Feigenbaum E.A. and Feldman, J.) New York: McGraw Hill (1963), 153-163.

[12] Gelernter, H., and Rochester, N. Intelligent behavior in problem-solving machines. IBM J. R & D 2 (1958), 336-345.

[13] Gilmore, P.C. An examination of the geometry theorem machine. Artificial Ingelligence, 1 (1970), 171-187.

[14] Hewitt, C. Description and theoretical analysis (using schemata) of PLANNER: a language for proving theorems and manipulating models in a robot. MIT Artificial Intelligence Laboratory, AI TR-258 (April 1972).

[15] Lee, R.C.T. A completeness theorem and a computer program for finding theorems derivable from given axioms. Ph.D. dissertation, Univ. of California, Berkeley, 1967.

[16] Loveland, D.W. A linear format for resolution. Symposium on Automatic Demonstration (ed. Laudet, M.), New York: Springer-Verlag (1970).

[17] Manna, Z. Properties of programs and the first-order predicate calculus. J.ACM, 16 (April 1969), 244-255.

[18] Norton, L.M. Experiments with a heuristic theorem-proving program for predicate calculus with equality. Artificial Intelligence, 2 (1971), 261-284.

[19] Plotkin, G.D. A further note on inductive generalization. Machine Intelligence 6 (eds. Meltzer, B. and Michie, D.). New York: American Elsevier (1971), 101-124.

[20] Robinson, J.A. A machine-oriented logic based on the resolution principle. J.ACM, 12 (January 1965), 23-41.

[21] Robinson, J.A. A review of automatic theorem proving. Proc. Symp. in Appl. Math., Providence, R.I.: American Math. Soc. (1967).

[22] Slagle, J.R. Automatic theorem proving with renamable and semantic resolution. J.ACM, 14 (October 1967), 687-697.

[23] Slagle, J.R. Automatic theorem proving with built-in theories including equality, partial ordering, and sets. J.ACM, 19 (January 1972), 120-135.

[24] Sandewall, E. Representing natural language information in predicate calculus. Machine Intelligence 6, (eds. Meltzer, B. and Michie, D.). New York: American Elsevier (1971), 255-277.

[25] Wang, H. Toward mechanical mathematics. IBM J. Res. Dev., 4 (1960), 2-22.

[26] Winograd, T. Procedures as a representation for data in a computer program for understanding natural language. MIT Project MAC TR-84 (February 1971).

[27] Wos, L., Robinson, G.A., and Carson, D.F. The concept of demodulation in theorem proving. J.ACM, 14 (October 1967), 698-709.