NON-RESOLUTION THEOREM PROVING

W. W. BLEDSOE

ATP 29 SEPTEMBER 1975

BASED ON A TUTORIAL TALK GIVEN AT THE FOURTH INTERNATIONAL JOINT CONFERENCE ON ARTIFICIAL INTELLIGENCE, IJCAI-75, TBILISI, USSR, SEPT. 8, 1975.

*THIS WORK WAS SUPPORTED BY NSF GRANT DCR74-12886.

NON-RESOLUTION THEOREM PROVING

by

W.W. Bledsoe

ABSTRACT

This talk reviews those efforts in automatic theorem proving, during the past few years, which have emphasized techniques other than resolution. These include: knowledge bases, natural deduction, reduction (rewrite rules), typing, procedures, advice, controlled forward chaining, algebraic simplification, built-in associativity and commutavity, models, analogy, and man-machine systems.

Examples are given and suggestions are made for future work.

Table of Contents

I. Introduction

Automatic theorem proving was born in the early 1930's with the work of Herbrand, but did not get much interest until high speed digital computers were developed. Earlier work by Newell, Simon, Shaw, and Gelernter in the middle and late 1950's, emphasized the heuristic approach, but the weight soon shifted to various syntactic methods culminating in a large effort on resolution type systems in the last half of the 1960's. It was about 1970 when considerable interest was revived in heuristic methods and the use of human supplied, domain dependent, knowledge.

It is not my intention here to slight the great names in automatic theorem proving, and their contributions to all we do, but rather to show another side of it. For recent books on automatic theorem proving see Chang and Lee [19], Loveland [44], and Hays [31]. Also see Nilsson's recent review article [61].

The word "resolution" has come to be associated with general purpose types of theorem provers which use very little domain dependent information and few if any special heuristics besides those of a syntactic nature. It has also connoted the use of clauses and refutation proofs.

There was much hope in the late 60's that such systems, especially with various exciting improvements, such as set of support, model eliminations, etc., would be powerful provers. But by the early 70's there was emerging a belief that resolution type systems could never really "hack" it, could not prove really hard mathematical theorems, without some extensive changes in philosophy.

This report is about this other non-resolution effort. But we do not just want to emphasize non-resolution, but rather to emphasize the efforts that are less syntactic in nature, that use heuristics and user supplied knowledge, which is often domain dependent. Our belief is that other purely syntactic methods such as Gentzen systems will fare only about as well as resolution systems, unless they employ some of the kinds of concepts we mention below. Also much improvement in resolution systems can be gained by using such concepts, and this has been done in many cases.

The author was one of the researchers working on resolution type systems who "made the switch". It was in trying to prove a rather simple theorem in set theory[1] by paramodulation and resolution, where the program was experiencing a great deal of difficulty, that we became convinced that we were on the wrong track. The addition of a few semantically oriented rewrite rules and subgoaling procedures [7] made the proof of this theorem, as well as similar theorems in elementary set theory, very easy for the computer. Put simply: the computer was not doing what the human would do in proving this theorem. When we instructed it to proceed in a "human-like" way, it easily succeeded. Other researchers were having similar experiences.

This is not really a general review in any fair sense. Rather it is a list of things I feel are important, with a real bias toward my work, and that of my students and friends. See Slides 1A, 1B.

A list of references is given at the end of the paper.

[1]The family of subsets of $(A \cap B)$ is the same as the intersection of the family of subsets of A and the family of subsets of B. This example is treated later.

II. Concepts

We will now list some of the concepts and techniques that we have in mind, that seem to hold promise in automatic theorem proving, and briefly discuss them. Of course no such list could be complete and we apologize for glaring omissions. Also these concepts are not mutually exclusive, some being special cases of others.

See Slide 2.

The word "knowledge" is a key to much of this modern theorem proving. Somehow we want to use the knowledge accumulated by humans over the last few thousand years, to help direct the search for proofs. Once a proof has been found it is a relatively simple matter to verify its validity by purely syntactic procedures. So in a sense all of our concepts have to do with the storage and manipulation of knowledge of one sort or another.

The use of knowledge and built-in procedures partially eliminates the need for long lists of axioms, which tend to slow up proofs and use excessive amounts of memory. Such knowledge must be organized in a way that is easy to use and change.

See Slide 3

This slide shows the 13 concepts we will discuss in the succeeding pages.

BASIC CONCEPTS

KNOWLEDGE

- BUILD-IN MAN'S KNOWLEDGE
- OFTEN DOMAIN-SPECIFIC
- CONTEXTUAL AND PERMANENT

AVOID LISTING OF AXIOMS

- CLOGS UP THE SYSTEM

EASY TO USE AND CHANGE

CONCEPTS

* 1. KNOWLEDGE BASE

* 2. REDUCTIONS (REWRITE RULES)

* 3. ALGEBRAIC SIMPLIFICATION

* 4. BUILT-IN INEQUALITIES (AND TOTAL ORDERING)

* 5. NATURAL SYSTEMS

* 6. FORWARD CHAINING

* 7. OVERDIRECTOR

* 8. TYPES

* 9. ADVICE

* 10. PROCEDURES (AND BUILT-IN CONCEPTS)

* 11. MODELS (AND COUNTEREXAMPLES)

* 12. ANALOGY

* 13. MAN-MACHINE

1. KNOWLEDGE BASE

We store information in a knowledge base (or data base), process that information to obtain other information (by procedural forward chaining, etc.), and interrogate the data base when necessary to answer questions. A central idea here is, that facts are stored about "objects" rather than "predicates". For example the hypothesis Open(A_0) would be stored with "Open" as a property of "A_0" rather than with "A_0" as a property of "Open". (Objects are the skolem constants arising in a proof.) Also knowledge is stored about concepts. This knowledge can be stored in procedures or in lists or other structures.

The planner - QA4 type systems are ideally suited for using these concepts. See for example Winograd's Thesis [84], especially Sections 3.1.3 - 3.3.1 for an excellent description of some of these concepts.

Some concepts associated with a knowledge base are shown in Slide 4. Demons are routines that watch the knowledge base and only act whenever certain properties become true of the data base. Languages like Microplanner, Conniver, and QA4 greatly facilitate the use of demons.

Some parts of the base remains static (as for example, properties of continuous functions) while other parts such as information about objects in the proof are dynamic and should be carried in a contextual data base.

The "graph" provers of Bundy [17], Ballantyne and Bennett [4] use such a data base, as do the provers of Winograd [84], Goldstein [28], and others. Minsky's frames [55] appear to offer good advice for organizing data for a knowledge base.

Shortly we will show an example from analysis [5] which utilizes a data base with many of the concepts we have discussed in this paper.

KNOWLEDGE BASE

CONTAINS FACTS ABOUT CONCEPTS AND OBJECTS

FACTS ARE MANIPULATED DURING PROOF TO OBTAIN NEW FACTS (CONTEXTUAL)

PROCEDURAL FORWARD CHAINING

STATIC INFORMATION

LOOK UP ANSWERS

OBJECT ORIENTED: $\underline{\text{OPEN } (A_0)}$

PROPERTY LIST $A_0 \rightsquigarrow_{\text{OPEN}}$

DEMONS

<u>EXAMPLES</u>

$$\left.\begin{array}{l}\text{PARTIAL SETS}\\ \text{MONADS}\\ \text{REDUCTION RULES}\end{array}\right\}\quad \text{SEE LATER}$$

"GRAPH" PROVERS

MODELS

FRAMES

★ 2. REDUCTION

See Slide 5.

A reduction is a rewrite rule,

$$A \dashrightarrow B .$$

For example, the rule

$$t \in (A \cap B) \dashrightarrow t \in A \wedge t \in B$$

requires that we change all subformulas of the form $t \in (A \cap B)$ into the form $(t \in A \wedge t \in B)$, (but never rewrite the later into the former). Such rules are <u>semantic</u>; their inclusion, and their use is not based upon their syntactic structure but on their meaning. The user supplies these rules. Slide 6 lists some such rules.

The use of REDUCTIONS is best illustrated by an example from [7].

See Slide 7.

Here the formula subsets(A) means the set of all subsets of A; we have shortened it to sb(A) for this proof.

The reduction rules and definitions given in Slide 6 are used in the proof. Notice how easy and "humanlike" the proof proceeds when reductions are used. A corresponding resolution proof (without built-in partial ordering) required 14 clauses and a lengthy deduction.

Reductions also offer a convenient way for storing unit facts that can be easily used during proofs.

See Slide 8.

CONTROLLED DEFINITION INSTANTIATION.

In this example we did <u>not</u> instantiate all definitions possible, but rather followed the rule: when all other strategies fail, instantiate the definition of the main connective of the conclusion.

REDUCTION

REDUCTION (REWRITE RULES)

CONDITIONAL REDUCTION

CONTROLLED DEFINITION INSTANTIATION

COMPLETE SETS OF REDUCTIONS

DEFINITIONS

$A = B$	$A \subseteq B \wedge B \subseteq A$	(SET EQUALITY)
$A \subseteq B$	$\forall x(x \in A \to x \in B)$	

SKOLEM FORM

$(x_0 \in A \to x_0 \in B)$	IN "CONCLUSION"
$(x \in A \to x \in B)$	IN "HYPOTHESIS"

SUBSETS (A)	$\{B : B \subseteq A\}$

SB(A)

REDUCE TABLE (REWRITE RULES)

IN	OUT
$T \in (A \cap B)$	$T \in A \wedge T \in B$
$T \in (A \cup B)$	$T \in A \vee T \in B$
$T \in \{x : P(x)\}$	$P(T)$
$T \in$ SUBSETS (A)	$T \subseteq A$
$T \subseteq A \cap B$	$T \subseteq A \wedge T \subseteq B$
$T \in \bigcup_{\alpha \in F} G(\alpha)$	$\exists \alpha(\alpha \in F \wedge T \in G(\alpha))$
$\vdots$	

SLIDE 6

THEOREM. $\forall A\ \forall B$ (SUBSETS $(A \cap B)$ = SUBSETS $(A) \cap$ SUBSETS (B))

(1) $SB(A \cap B) = SB(A) \cap SB(B)$ THE GOAL

$[SB(A \cap B) \subseteq SB(A) \cap SB(B)] \wedge [\ \ldots \supseteq \ldots\ ]$, DEFN OF =

SUBGOAL 1

(11) $[SB(A \cap B) \subseteq SB(A) \cap SB(B)]$

$[T_0 \in SB(A \cap B) \Rightarrow T_0 \in (SB(A) \cap SB(B))]$, DEFN OF $\subseteq$

$[T_0 \subseteq (A \cap B) \Rightarrow T_0 \in SB(A) \wedge T_0 \in SB(B)]$

REDUCE

$[T_0 \subseteq A \wedge T_0 \subseteq B \Rightarrow T_0 \subseteq A \wedge T_0 \subseteq B]$

REDUCE

"T"

SUBGOAL 2

(12) $[SB(A) \cap SB(B) \subseteq SB(A \cap B)]$

"T" SIMILARLY

QED

Storing Unit Facts By Reductions

IN	OUT
$\emptyset \subseteq A$	"T"
$A \subseteq A$	"T"
$\emptyset \in \omega$	"T"
$A \in \emptyset$	"FALSE"
$0 \leq 1$	"T"
OPEN $\emptyset$	"T"
$A \subseteq \bar{A}$	"T"
•	
•	
•	

WE WOULD NOT INCLUDE

$P(Y) \wedge (P(X) \longrightarrow Q(X)) \longrightarrow Q(Y)$	"T"

BECAUSE IT IS TOO COMPLEX.

Instantiating all definitions can badly clutter up the proof and is often not needed. For example, we would not define "Regular" in the theorem

$$(\text{Separable } \mathcal{T} \wedge \text{Regular } \mathcal{T} \longrightarrow \text{Regular } \mathcal{T}) .$$

In general, definition instantiation should be carefully controlled. See Slide 9.

CONDITIONAL REDUCTION

This is a slight generalization of the reduction concept, whereby the program will perform the reduction only if a given stated condition is true in the <u>data base</u>. We do <u>not</u> want a large effort expended to determine whether the condition is true, because Reductions are supposed to be performed quickly, so we verify the condition in the data base (rather than call the prover itself for this purpose). See Slide 10 for a simple example.

COMPLETE SETS OF REDUCTIONS

Instead of using a reduction

$$A \dashrightarrow B$$

one could get the same effect by adding the formula

$$A = B$$

as another hypothesis, but this would usually greatly increase the run time and storage space. So it is clearly desirable to use reductions instead of equations whenever we can.

When can we use <u>only</u> reductions instead of equations? This is the object of much research right now. The principal workers in this area, which is called <u>complete sets of reductions</u>, are shown on Slide 11. The pioneering work of Knuth and Bendix [40] dealt principally with group theory, where the initial equality axioms can <u>all</u> be converted to reductions. See Slide 12. Slide 13 gives a proof in group theory using these reductions.

CONTROLLED DEFINITION INSTANTIATION

<u>EXAMPLES</u>:

DO NOT EXPAND DEFINITIONS IN:

$(A \subseteq B \wedge x \in A \wedge \text{OPEN } A \longrightarrow \text{OPEN } A)$

DO EXPAND THE DEFINITION OF "OCCLFR" IN

$(\text{REGULAR } (\mathcal{T}) \wedge \text{OCLFR}(\mathcal{T}) \longrightarrow \text{OCCLFR}(\mathcal{T}))$

IF OTHER ATTEMPTS FAIL.

CONDITIONAL REDUCTIONS

IN	CONDITION	OUT
INTERIOR (A)	OPEN (A)	A
CLOSURE (A)	CLOSED (A)	A
$\|A\|$	$A \geq 0$	A
$\|A\|$	$A < 0$	-A
REGULAR ($\mathcal{J}$)	$T_1(\mathcal{J})$	$T_3(\mathcal{J})$
NORMAL ($\mathcal{J}$)	$T_1(\mathcal{J})$	$T_4(\mathcal{J})$

EXAMPLE THEOREM. $2 \leq I \wedge 4 \leq J \leq 9$

$$\wedge \ (0 \leq K \wedge L \leq 10 \longrightarrow P(K,L)) \longrightarrow P(I,|J|).$$

PROOF. THE HYPOTHESIS $2 \leq I$ AND $4 \leq J \leq 9$ ARE STORED IN THE DATA BASE ON THE PROPERTY LISTS OF I AND J. THE TERM $|J|$ IS REDUCED (REWRITTEN) TO J AFTER A CHECK IN THE DATA BASE VERIFIES THE CONDITION $J \geq 0$.

BACKCHAINING ON THE THIRD HYPOTHESIS NOW GIVES THE SUBGOAL $(0 \leq I$ AND $J \leq 10)$ WHICH IS EASILY VERIFIED BY THE DATA BASE. QED

COMPLETE SETS OF REDUCTIONS

KNUTH AND BENDIX (40)	
SLAGEL (73)	NARROWINGS
PLOTKIN (63)	
LANKFORD (41,42)	
WINKER (83)	DEMODULATIONS
LIFSHITS (43)	

COMPLETE SET OF REDUCTIONS

FOR A GROUP

KB1	$x + 0 \to x$
KB2	$0 + x \to x$
KB3	$x + (-x) \to 0$
KB4	$(-x) + x \to 0$
KB5	$(x + y) + z \to x + (y + z)$
KB6	$-0 \to 0$
KB7	$-(-x) \to x$
KB8	$-(x + y) \to (-y) + (-x)$
KB9	$x + ((-x) + y) \to y$
KB10	$(-x) + (x + y) \to y$

THEOREM.

- $[(D + (C + (-C))) + (-(0 + (-A)))] \in H$

$$\longrightarrow (D + A) \in H$$

PROOF. (USING THE COMPLETE SET OF REDUCTIONS FOR A GROUP ON SLIDE 12).

- $[(D + (C + (-C))) + (-(-A))] \in H$

  $\longrightarrow (D + A) \in H$ KB2

- $[(D + 0) + (-(-A))] \in H$

  $\longrightarrow (D + A) \in H$ KB3

- $[D + (-(-A))] \in H \longrightarrow (D + A) \in H]$ KB1

- $[D + A] \in H \longrightarrow (D + A) \in H]$ KB6

- TRUE

Unfortunately there is no complete set of reductions for commutative group theory, ring theory, and many other theories, so efforts have been made and are being made to extend the concepts of [40], to these theories. [73,63,41,42,83].

For example, in proving a theorem of the form

$$(x + (x + x) = 0 \longrightarrow C) ,$$

when the hypothesis, $(x + (x + x) = 0)$, is added to the axioms of group theory the whole set no longer can be converted to a complete set of reductions. The extensions mentioned above are designed to cope with such situations. For example, Lankford's method [41], handles the theorem

$$(x + (x + x) = 0 \longrightarrow$$
$$A + B + (-A) + B + A + (-B) + (-A) + (-B) = 0)$$

(Associate to the right)

by adding the hypothesis as another reduction

11. $$(x + (x + x) \longrightarrow 0) ,$$

and proving the theorem in two rounds. See Slides 14, 15, 16.

It is hoped that these techniques might help provide a more satisfactory answer to the equality problem. However, it is our belief that heuristics methods and built-in procedures such as those given by Slagle and Norton [76] and Nevins [56] will be needed for any truly effective handling of equality.

A challenging problem in this area is the Ring Theory theorem given on Slide 17.

COMPLETE SET OF REDUCTIONS

FOR A GROUP

KB1	$x + 0 \to x$
KB2	$0 + x \to x$
KB3	$x + (-x) \to 0$
KB4	$(-x) + x \to 0$
KB5	$(x + y) + z \to x + (y + z)$
KB6	$-0 \longrightarrow 0$
KB7	$-(-x) \to x$
KB8	$-(x + y) \to (-y) + (-x)$
KB9	$x + ((-x) + y) \to y$
KB10	$(-x) + (x + y) \to y$

THEOREM. $x + (x + x) = 0 \quad \to$

$$A + B + (-A) + B + A + (-B) + (-A) + (-B) = 0.$$

(ASSOCIATE TO THE RIGHT)

THE HYPOTHESIS $(x + (x + x) \to 0)$ IS ADDED AS ANOTHER RECUCTION.

LANKFORD'S METHOD

11.	$x + (x+x) \longrightarrow 0$	ADDED

IN THE FIRST ROUND, THREE NEW REDUCTIONS ARE GENERATED, AND ONE EQUALITY:

N1	$x + x + x + y \longrightarrow y$	11,KB5
N2	$x+y + x+y + x+y \longrightarrow 0$	11,KB5
N3	$-x \longrightarrow x + x$	11,KB10

E1	$x+y + x+y = y+y + x+x$ (THIS CANNOT BE MADE A REDUCTION)	N3,KB8

N1-3 ACT UPON KB1-10,11 AND ELIMINATE ALL OF KB1-10,11 EXCEPT

KB1	$x + 0 \longrightarrow 0$
KB2	$0 + x \longrightarrow$
KB5	$(x+y) + z \longrightarrow x+(y+z)$
11	$x + x + x \longrightarrow 0$

In the second round, three new reductions are generated, and no new equations

N4 $x + y + \quad x + y \quad + \quad x + y \quad + z \longrightarrow 0$

N5 $x + y + z \quad + \quad x + y + z \quad + x + y + z \longrightarrow 0$

N6 $x + y \quad + \quad x + y \quad + x \longrightarrow y + y$

The Goal:

$$A + B + (-A) + B + A + (-B) + (-A) + (-B) = 0$$

is proved in this round by applying

N3, E1, N1, N6 & 11

Steps: 2

Formulas saved: 7

Time: 30 seconds

CHALLENGING PROBLEM

<u>Th.</u> In a Ring

$$x^3 = 1 \longrightarrow AB = BA$$

(Recall that a Ring is + Commutative)

3. ALGEBRAIC SIMPLIFICATION

There is a strong need to avoid adding the field axioms for the real numbers as hypotheses to a theorem being proved, because this greatly slows proofs. The associativity and commutativity axioms for $+$ and $\cdot$ are especially troublesome, so several efforts have been made to "build these in".

Some references to this work are:

Slagle and Norton [74,75]
QA4, QLISP [65,68]
Plotkin [63]
Frönig [25]
Stickel [77]
Bledsoe, et al [9,12]

Of course much is learned from the researchers working in the field of symbol manipulation and algebraic simplification, where these methods have been applied to other problems in physics and mathematics. However, automatic theorem proving presents difficulties not covered by that work.

For example, the theorem

$$P(a + b + c) \longrightarrow P(b + a + c)$$

is easily handled by using a canonical form, but the theorem

$$P(k + 2) \longrightarrow P(b + 5)$$

where k is a variable and b is a constant, presents more difficulty. An ordinary unification algorithm

$$\text{UNIFY}(k + 2, b + 5)$$

would put b for k, b/k, and fail. An "Algebraic Unifier", could write the equation

$$k + 2 = b + 5 ,$$

and "solve for" k, getting $k = b + 3$, and return $(b + 3)$ for k, to successfully complete the proof.

A similar approach works on the example

$$\begin{aligned} \mathrm{UNIFY}(B[k+1] &= \mathrm{Amax}(B, j, k+1) , \\ A_0(i_0] &= \mathrm{Amax}(A_0, 1, i_0)) \end{aligned}$$

where B, j, k are variables, and A_0, i_0 are constants. This example is from the field of program verification (See [13], pp. 27-28).

Data types such as sets, bags, and triples [68,65] handle some of these problems.

★ 4. BUILT-IN INEQUALITIES (and total orderings)

Again we must avoid the explicit use of such axioms as the transitivity axiom

$$(x \leq y \wedge y \leq z \longrightarrow x \leq z) \ .$$

See Slide 18.

Slagle and Norton [75] have built-in axioms for handling total and partial ordering (including inequalities). See Slide 19.

Slide 20 gives a theorem from program verification which was proved by Slagle and Norton's program. This theorem and others like it have been proved by the "interval type" methods of Bledsoe and Tyson [13,8], and by others.

Also Slagle and Norton have built-in partial ordering for handling some problems in set theory.

✓

INEQUALITIES

(AVOID AXIOMS LIKE

$X \leq Y \wedge Y \leq Z \longrightarrow X \leq Z$)

BUILT IN TOTAL ORDERING

- SLAGLE & NORTON

INTERVAL TYPES

- BLEDSOE ET AL

SLAGLE & NORTON

Built In

PARTIAL ORDERING

(INCLUDES TOTAL ORDERING)

$$\begin{tikzcd}
A \le B \arrow[dr, dash] & & B \le C \arrow[dl, dash] \\
& A \le C &
\end{tikzcd}
\qquad
\begin{tikzcd}
A < B \arrow[dr, dash] & & B \le C \arrow[dl, dash] \\
& A < C &
\end{tikzcd}$$

$$\begin{tikzcd}
A \le B \arrow[dr, dash] & & B \le A \arrow[dl, dash] \\
& A = B &
\end{tikzcd}
\qquad
\begin{tikzcd}
A \le B \arrow[dr, dash] & & A = C \arrow[dl, dash] \\
& C \le B &
\end{tikzcd}$$

$$\begin{tikzcd}
A < A \arrow[d] \\
\square
\end{tikzcd}
\qquad
\begin{tikzcd}
A \le B \arrow[d] \\
A < B \wedge A = B
\end{tikzcd}$$

Allow Unification in All of these Cases.

THEOREM.

$$
\begin{aligned}
& J < I \\
\wedge\ & M \le P \le Q \le N \\
\wedge\ & \forall X\, \forall Y (M \le X < I \wedge J < Y \le N \rightarrow A[X] \le A[Y]) \\
\wedge\ & \forall X\, \forall Y (M \le X \le Y \le J \rightarrow A[X] \le A[Y]) \\
\wedge\ & \forall X\, \forall Y (I \le X \le Y \le N \rightarrow A[X] \le A[Y]) \\
\rightarrow\ & A[P] \le A[Q]
\end{aligned}
$$

A verification condition from Hoare's FIND program

★ 5. NATURAL SYSTEMS

We have chosen to emphasize the so called "natural" systems in this report. I would not like to define the term, but will only give examples. In general we are not talking about refutation systems such as resolution, though we sometimes do proofs by contradiction [66]. They are sometimes called goal oriented systems, or Gentzen type systems. See Slide 21.

We are given a goal G and a hypothesis H and wish to show that G follows from H,

$$(H \longrightarrow G)$$

or more generally to find a substitution θ for which

$$(H\theta \longrightarrow G\theta)$$

is a valid propositional formula. A set of rules is given for manipulating H and G to obtain the desired θ. See Slide 22. The Rules on Slides 23 and 24 are from the IMPLY System described in [12,9]. They are given more precisely and completely in [12]. See Slide 25 for a simple example.

Newell, Simon, and Shaw's logic theorist [59,60], and Gelernter's geometry machine [26], were natural (or goal directed) systems, although we will see that they included various other features. See Slides 26 and 27. Reiter's MATH-HACK system [66] is much like that of [12] but has the important addition of models which we will mention later, and other features.

Other natural systems include the Planner-Conniver-QA4 group, and those of Maslov, Goldstein, Nevins, Bibel, Boyer-Moore, Ernst, [48,49,28,34,78,52,68,65, 6,14,23,46,15,30] and others. See Slide 28.

What are the advantages (if any) of the natural systems? There may be none--

NATURAL SYSTEMS

GOAL ORIENTED

GENTZEN SYSTEMS

(NOT REFUTATION)

NATURAL SYSTEM

$(H \Rightarrow G)$

H IS A SET OF HYPOTHESES.

G IS A GOAL.

TO FIND A SUBSTITUTION θ FOR WHICH

$$(H\theta \rightarrow G\theta)$$

IS A VALID PROPOSITIONAL FORMULA.

<u>EXAMPLE</u>.

$$P(A) \wedge (P(x) \rightarrow Q(x)) \Rightarrow Q(A)$$

ANSWER: $\theta \equiv A|X$

SOMETIMES θ MUST BE MORE COMPLICATED (SEE APPENDIX 3 OF [12]) AS IN THE FOLLOWING EXAMPLE, WHERE A DISJUNCTION IS NEEDED

$$(P(x) \longrightarrow P(A) \wedge P(B))$$

ANSWER: $\theta = A/X \vee B/X$

IMPLY RULES

A PARTIAL SET FROM [12]

I4. $(H \Rightarrow A \wedge B)$ "SPLIT"

IF $(H \Rightarrow A)$ RETURNS θ

AND $(H \Rightarrow B\theta)$ RETURNS λ

THEN RETURN $\theta \circ \lambda$

I3. $(H_1 \vee H_2 \Rightarrow C)$ "CASES"

IF $(H_1 \Rightarrow C)$ RETURNS θ

AND $(H_2\theta \Rightarrow C)$ RETURNS λ

THEN RETURN $\theta \circ \lambda$

I5. $(H \Rightarrow C)$

PUT $C' :=$ REDUCE(C); $H' :=$ REDUCE(H)

CALL $(H' \Rightarrow C')$

I7. $(H \Rightarrow (A \rightarrow B))$ "PROMOTE"

CALL $(H \wedge A \Rightarrow B)$.

I13. $(H \Rightarrow C)$

PUT $C' :=$ DEFINE(C)

CALL $(H \Rightarrow C')$

```
H2.   $(H \Rightarrow C)$                                   "MATCH"
        IF  $H\theta \equiv C\theta$,  RETURN  $\theta$.

H6.   $(A \wedge B \Rightarrow C)$                          "OR-FORK"
        IF  $(A \Rightarrow C)$ RETURNS $\theta$ (NOT NIL),  RETURN $\theta$.
        ELSE CALL  $(B \Rightarrow C)$

H7.   $H \wedge (A \rightarrow D) \Rightarrow C$            "BACK-CHAIN"
        IF  $(D \Rightarrow C)$ RETURNS  $\theta$,
        AND  $(H \Rightarrow A\theta)$ RETURNS  $\lambda$,
           THEN RETURN $\theta \circ \lambda$

H9.   $H \wedge (A = B) \Rightarrow C$                      "SUB = "
        PUT  $A' :=$ CHOOSE $(A,B)$, $B' :=$ OTHER$(A,B)$
        CALL $(H(A'/B') \Rightarrow C(A'/B'))$.
```

Example Using IMPLY

<u>Theorem</u>. $(P(A) \wedge \forall x(P(x) \rightarrow Q(x)) \rightarrow Q(A))$.

$P(A) \wedge (P(x) \rightarrow Q(x)) \Rightarrow Q(A))$

$(P(A) \Rightarrow (Q(A))$ H6

FAILS

$(P(x) \rightarrow Q(x)) \Rightarrow Q(A))$

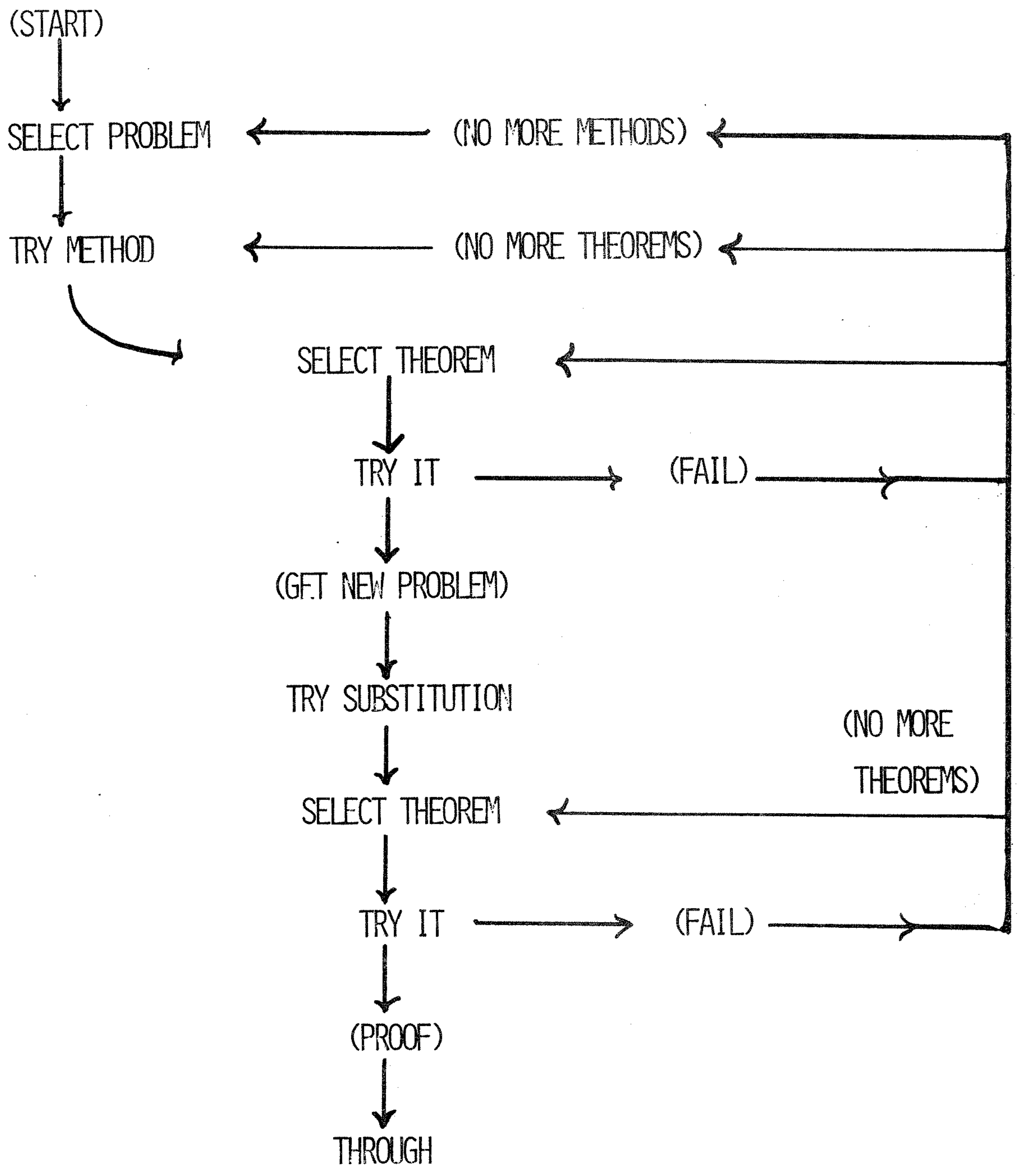
$(Q(x) \Rightarrow Q(A))$ H7

Return $\theta \equiv A|x$ H2

$(P(A) \Rightarrow P(x)(A|x))$

Return TRUE.

Returns $A|x$.



GENERAL FLOW DIAGRAM OF THE
LOGIC THEORIST [59]

NEWELL - SIMON _ SHAW
1957

SLIDE 26

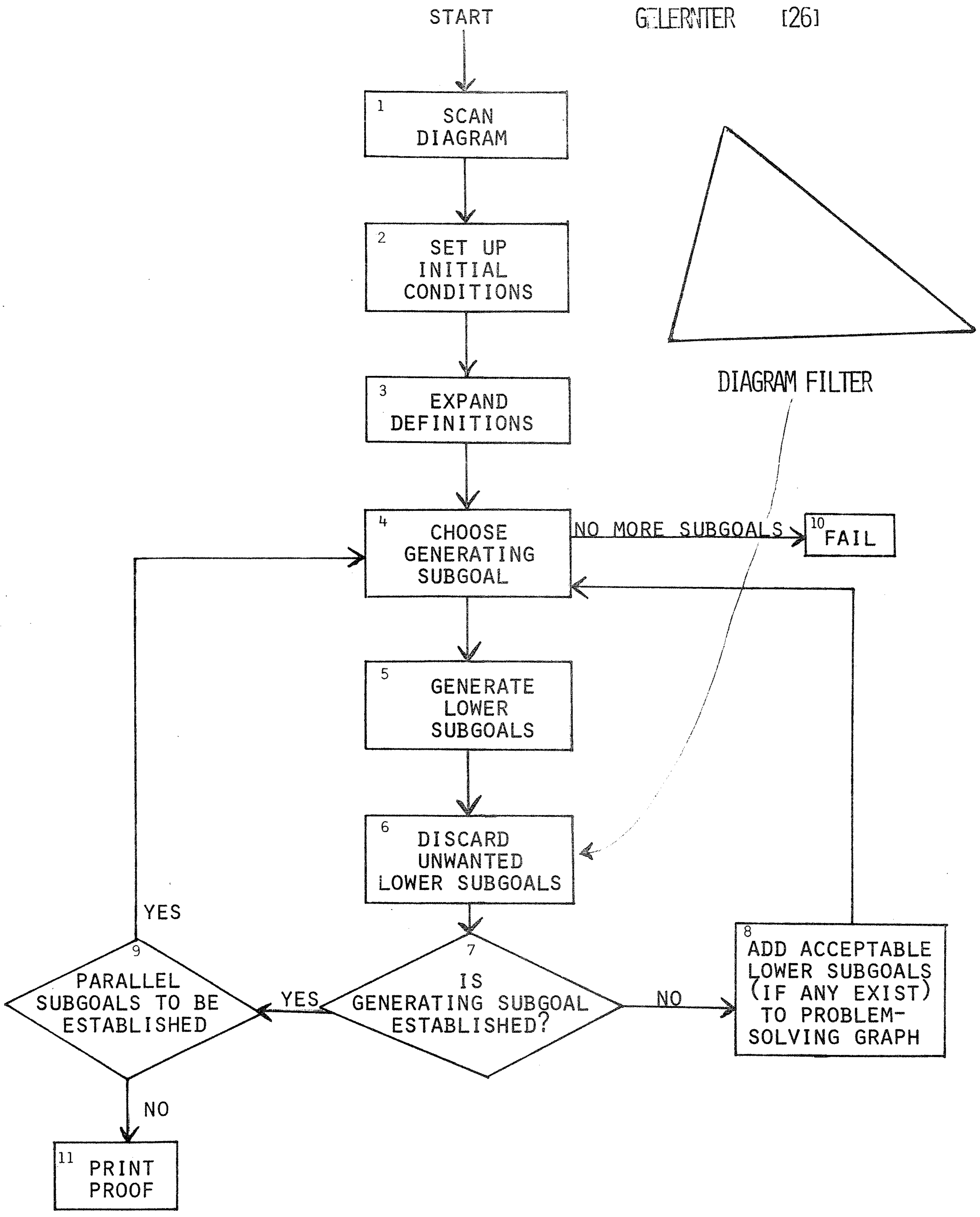

START
1 SCAN DIAGRAM
2 SET UP INITIAL CONDITIONS
3 EXPAND DEFINITIONS
4 CHOOSE GENERATING SUBGOAL
NO MORE SUBGOALS
10 FAIL
5 GENERATE LOWER SUBGOALS
DIAGRAM FILTER
6 DISCARD UNWANTED LOWER SUBGOALS
7 IS GENERATING SUBGOAL ESTABLISHED?
NO
8 ADD ACCEPTABLE LOWER SUBGOALS (IF ANY EXIST) TO PROBLEM-SOLVING GRAPH
YES
9 PARALLEL SUBGOALS TO BE ESTABLISHED
YES
NO
11 PRINT PROOF

FIGURE 3. SIMPLIFIED FLOW CHART FOR THE GEOMETRY-THEOREM PROVING MACHINE.

OTHER NATURAL SYSTEMS

NSS - Logic Theorist (Later)

Gelernter's Geometry machine (Later)

Reiter's math-hack - Much like this but more

Maslov

Bibel Nevins

Ernst

Boyer-Moore

$$\left.\begin{array}{l}\text{Planner}\\ \text{Conniver}\\ \text{QA4, QLISP}\end{array}\right\}\ \text{Examples Later}$$

Goldstein

especially in the long run--and especially if the techniques we emphasize here are built into the resolution systems. But we feel that this is not easy to do. Specifically we feel that the natural systems are:

- Easier for human use
- Easier for machine use of knowledge

See Slides 29, 30, 31, 32.

On the question of completeness I refer to a statement by Winograd. Slide 33.

NATURAL SYSTEMS

- Easier for human use
- Easier for machine use of knowledge

NATURAL SYSTEMS

HUMAN USE

- BRING TO BEAR KNOWLEDGE FROM PURE MATHEMATICS IN THE SAME FORM USED THERE
- RECOGNIZE SITUATIONS WHERE SUCH KNOWLEDGE CAN BE USED
- PROFESSIONAL MATHEMATICIAN WILL WANT TO PARTICIPATE
- EASIER TO DESIGN, PROGRAM, WORK UPON
- ESSENTIAL FOR MAN-MACHINE INTERACTION (WHERE THE MAN IS A TRAINED MATHEMATICIAN)

NATURAL SYSTEMS

MACHINE USE

AUTOMATICALLY LIMITS THE SEARCH. DOES NOT START ALL PROOFS OF THE THEOREM. (SYNTACTIC SEARCH STRATEGY).

A NATURAL VEHICLE TO HANG ON HEURISTICS, KNOWLEDGE, SEMANTICS. (SEMANTIC SEARCH STRATEGIES).

EASIER TO COMBINE PROCEDURES WITH DEDUCTION.

CONTEXTUAL DATA BASE PROBLEM. ONE FOR EACH CLAUSE.

INCOMPLETE

NEW LANGUAGES (PLANNER, QA4). EASE THE IMPLEMENTATION.

NATURAL SYSTEMS

"THERE IS A NATURALNESS WITH WHICH SYSTEMS OF NATURAL DEDUCTION ADMIT A SEMANTIC COMPONENT WITH THE RESULT THAT A GREAT DEAL OF CONTROL IS GAINED OVER THE SEARCH FOR A PROOF. IT IS PRECISELY FOR THIS REASON THAT WE ARGUE IN FAVOUR OF THEIR USE IN AUTOMATIC THEOREM-PROVING, IN OPPOSITION TO THE USUAL RESOLUTION-BASED SYSTEMS, WHICH APPEAR TO LACK ANY KIND OF REASONABLE CONTROL OVER DEAD-END SEARCHES."

RAYMOND REITER [66]

"A POINT WORTHY OF STRESS IS THAT A DEDUCTIVE SYSTEM IS NOT "SIMPLER" MERELY BECAUSE IT EMPLOYS FEWER RULES OF INFERENCE. A MORE MEANINGFUL MEASURE OF SIMPLICITY IS THE EASE WITH WHICH HEURISTIC CONSIDERATIONS CAN BE ABSORBED INTO THE SYSTEM."

ARTHUR NEVINS [56]

INCOMPLETENESS

"IT IS WORTH POINTING OUT HERE THAT COMPLETENESS MAY IN FACT BE A BAD PROPERTY. IT MEANS (WE BELIEVE, NECESSARILY) THAT IF THE THEOREM-PROVER IS GIVEN SOMETHING TO PROVE WHICH IN FACT IS FALSE, IT WILL EXHAUST EVERY POSSIBLE WAY OF TRYING TO PROVE IT. BY FORSAKING COMPLETENESS, WE ALLOW OURSELVES TO USE GOOD SENSE IN DECIDING WHEN TO GIVE UP."

TERRY WINOGRAD [84 , P. 236]

✯ 6. FORWARD CHAINING

Forward chaining is accomplished when one hypothesis is applied to another to obtain an additional hypothesis. See the example in Slide 34.

Since such a process can, in some cases, result in an infinite repetition it is important that it be controlled by a cut-off mechanism. Also we have found other controls desirable, such as allowing only those new hypotheses which are ground formulas.

Procedural forward chaining is where a procedure is invoked which manipulates items in the data base (or in the hypothesis) to produce new items. This is exhibited in the non-standard analysis example given below and in [5].

Several programs have found a good deal of success through extended use of forward chaining. See Slide 35. Nevins remarks [58, pp.2,3] are particularly interesting on this point.