FORWARD CHAINING

CONTROLLED FORWARD CHAINING

- GROUND RESULTS ONLY
- CUT-OFF LEVELS

PROCEDURAL FORWARD CHAINING

- EXAMPLES LATER
- MANIPULATION OF DATA BASE

EXAMPLE

$$\begin{tikzcd}
 & \text{Hypothesis} & & & & \text{Conclusion} \\
P(A) \arrow[r, phantom, "\wedge"] \arrow[rr, dashed, bend right] & {} & (P(x) \arrow[r] & Q(x)) \arrow[d, dashed] \arrow[rr, Rightarrow] & & C \\
 & & \wedge & Q(A) & &
\end{tikzcd}$$

FORWARD CHAINING

NEWELL - SIMON - SHAW

MANY OTHERS

EXTENSIVE USE

BUNDY [17] - DOING ARITHMETIC WITH DIAGRAMS

SIKLOSSY & MARINOV [71] - BRITISH MUSEUM

BALLANTYNE & BENNETT [4] - TOPOLOGY

NEVINS [58] - PLANE GEOMETRY

NON-STANDARD ANALYSIS [5] (LATER)

★ 7. OVERDIRECTOR

Every prover has a control routine which directs the search tree. See Slide 36. Newell, Simon, and Shaw's control structure is shown in Slide 37.

This overdirector can bring to bear strategies or experts (see [28]), heuristics, and advice tables, as it sees fit.

It is important that such an overdirector have the flexibility to switch from one line of attach to another, and back again, as the proof proceeds, thus providing a parallel search capability. This of course, requires a (controlled) back-up mechanism such as that possessed by Micro-planner. Unrestricted back-up is intolerable. A contexual data base, which can be consulted by the overdirector to help it decide whether and how much to back-up, or what other line of attach to take, is an indispensable part of the prover we have in mind. The concepts of Conniver [52] and QA4 [68,65] apply here.

EXAMPLE from Non-Standard Analysis

The following example is given here to exhibit the use of some of the concepts we have described above. These techniques have been used by Mike Ballantyne to prove by computer (not interactively) several difficult theorems in Intermediate Analysis. See [5] for a complete description of this work.

The reader need not be conversant with non-standard analysis (or even intermediate analysis) to follow the example given on Slides 38 and 39.

Notice that the proof follows the general procedure described by the rules of Slides 23-24. First, the fact that f is continuous, is noted in the data base and the hypothesis $\mathrm{Cont}(f,S_0)$ is dropped. Next the term "Compact" is defined (in non-standard terms), and the formula $x_0 \in f(S_0)$ is "promoted" to the hypothesis by Rule 17 of Slide 23, and then reduced to produce the new hypothesis.

$$V_0 \in S_0 \wedge x_0 = f(V_0) .$$

OVERDIRECTOR

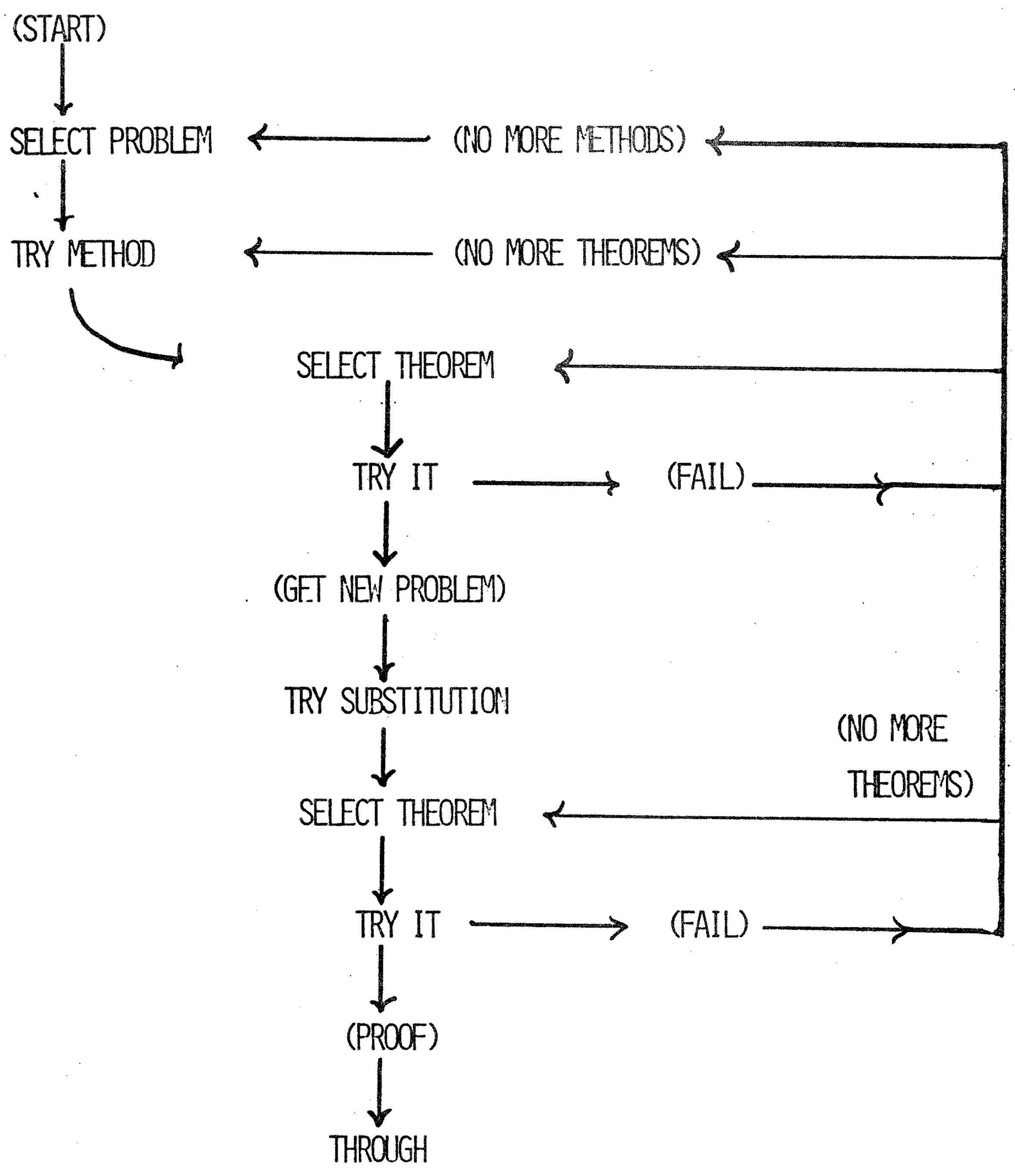
Directs the search tree

Strategies-Experts

Heuristics

Advice tables

Controlled backup



GENERAL FLOW DIAGRAM OF THE
LOGIC THEORIST [59]

NEWELL - SIMON _ SHAW
1957

MIKE BALLANTYNE

EXAMPLE

5.2 <u>THEOREM</u>. IF F IS CONTINUOUS ON A COMPACT SET S, THEN F(S) IS COMPACT.

- $\text{CONT}(F, S_0) \wedge \text{COMPACT}(S_0) \Longrightarrow \text{COMPACT}(F(S_0))$

  NOTE: F IS CONTINUOUS ON S_0 $\left(\begin{array}{c}\text{IN}\\ \text{DATA}\\ \text{BASE}\end{array}\right)$

- $\text{COMPACT}(S_0) \Longrightarrow \text{COMPACT}(F(S_0))$

- $(x \in S_0 \longrightarrow \text{ST}(x) \in S_0) \Longrightarrow (x_0 \in F(S_0) \longrightarrow \text{ST}(x_0) \in F(S_0))$

  DEFINITION

- $(\qquad \text{''} \qquad) \wedge x_0 \in F(S_0) \Longrightarrow \text{ST}(x_0) \in F(S_0)$

- $(x \in S_0 \longrightarrow \text{ST}(x) \in S_0) \wedge (V_0 \in S_0 \wedge x_0 = F(V_0) \Longrightarrow$

  REDUCE

- $(x \in S_0 \longrightarrow \text{ST}(x) \in S_0) \wedge (V_0 \in S_0 \wedge x_0 = F(V_0)$

  $\wedge \; \underline{\text{ST}(V_0) \in S_0} \Longrightarrow \text{ST}(x_0) \in F(S_0)$

  FORWARD CHAIN

DATA BASE	AGENT
• S_0' : $(\mathrm{ST}(V_0)\ ,\ V_0)$	EL
TYPE: $\mathrm{ST}(V_0)$, STANDARD	STANDARD
TYPE: $\mathrm{ST}(V_0)$, FINITE V_0 , FINITE	"
M_1 : $(\mathrm{ST}(V_0)\ ,\ V_0)$	FINITE
R_1 : $x_0 \dashrightarrow \mathrm{F}(V_0)$	EQUALS
∘ M_2 : $(\mathrm{F}(\mathrm{ST}(V_0))\ ,\ \mathrm{F}(V_0))$	CONTINUOUS
• R_2 : $\mathrm{ST}(\mathrm{F}(V_0)) \dashrightarrow \mathrm{F}(\mathrm{ST}(V_0))$	"
$\mathrm{F}(S_0)'$: $(\mathrm{F}(\mathrm{ST}(V_0))\ ,\ \mathrm{F}(x_0))$	"
• $(x \in S_0 \longrightarrow \mathrm{ST}(x) \in S_0) \Longrightarrow \mathrm{ST}(x_0) \in \mathrm{F}(S_0)$	
• (") $\Longrightarrow \mathrm{ST}(\mathrm{F}(V_0)) \in \mathrm{F}(S_0)$	R_1
• ( ) $\Longrightarrow \mathrm{F}(\mathrm{ST}(V_0)) \in \mathrm{F}(S_0)$	R_2
• TRUE	$\mathrm{F}(S_0)'$

Forward chaining then gives the additional hypothesis: $st(V_0) \in S_0$.

At this point we leave the rules of Slides 23-24, and work with the data base. See [5] for details.

Various routines such as EL, STANDARD, FINITE, CONTINUOUS, are used to put things into the data base and to manipulate them to obtain others. For example, the program detects $(V_0 \in S_0)$ and $(st(V_0) \in S_0)$ in the hypothesis and calls EL which builds a set S_0' in the data base with the elements V_0, and $st(V_0)$, and drops $(V_0 \in S_0)$ and $(st(V_0) \in S_0)$ from the hypothesis. This set S_0' with only two elements represents the set S_0 which may be infinite.

Similarly the momad M_1: $(st(V_0), V_0)$ is built in the data base. The reader needs only to understand that continuous functions map monads into monads, (and not what a monad is) and hence that the monad M_1 is mapped into the monad M_2: $(f(st(V_0)), f(V_0))$.

The hypothesis $(x_0 = f(V_0))$ is used to generate the reduce rule R_1 and another routine generates R_2. Thus the goal $st(x_0) \in f(S_0)$ is easily converted to the new goal $f(st(V_0)) \in f(S_0)$, which is readily verified by inspection; i.e., the program notes that the set $f(S_0)'$ in the data base, contains the item $f(st(V_0))$.

In summary, one sees the manipulation of a data base and the execution of a few logical operations, to produce the proof of this theorem.

★ 8. TYPES

The concept of typing plays a fundamental role in mathematics and computer science. Using a letter e for the identity element of a group, lower case letters x,y,z, for members of the group, and capital letters G,H, for groups and subgroups, is immensely helpful to humans in proving theorems.

Similar typing is helpful in automatic provers. Other data type such as integer, real, negative, complex, bags, sets, types, interval types, infinitesimals, infinitely large, etc., can be advantageous in certain applications.

See Slide 40.

TYPES

e	IDENTITY IN A GROUP
X,Y,Z	MEMBERS OF A GROUP
G,H	GROUPS, SUBGROUPS
X,Y,Z	POINT
A,B,C	SETS
F,G,H	FAMILIES
$\mathcal{T}$	TOPOLOGY
P,Q	PREDICATE
X,Y	REALS
Z	COMPLEX
I,J,K	INTEGERS
ε, δ	INFINITESIMALS
R,S,T	STANDARD REALS
X,Y,Z	NON-STANDARD REALS
ω	INFINITELY LARGE INTEGERS

BAGS, SETS, TYPLES

INTERVAL TYPES

★ 9. ADVICE

One of the most powerful things a human can do to aid the prover is to provide "advice" for the use of a theorem or lemma. Carl Hewitt's PLANNER [34] exploits this idea.

For example in Slide 41 we see an example of Winograd's where, to determine that a thing x is a thesis we are "advised" to either verify that it is long, or that it contains a persuasive argument. This is given in Micro-planner language in Slide 42.

Another such advice lemma is given in Slide 43, and this is used in Slide 44 to prove a theorem. This proof also clearly emphasizes the need for simplification routines in proofs in analysis.

The concept depicted in Slide 43 might be generalized in a manner shown in Slide 45. Then perhaps an instantiation of it (like Slide 43) could be saved by the program for future use.

ADVICE

GOAL

(THESIS X)

VERIFY

(LONG X) (USE: CONTENTS - CHECK,
COUNT PAGES)

OR

(X CONTAINS Y)
AND
(ARGUMENT Y)
AND
(PURSUASIVE Y)

WINOGRAD'S EXAMPLE

```
(DEFINE THEOREM EVALUATE
                              ;EVALUATE is the name we are
                              ;giving to the theorem

  (THCONSE(X Y)
                              ;this indicates the type of
                              ;theorem and names its
                              ;variables

    (THGOAL(#THESIS $?X))
                              ;show that X is a thesis
                              ;the "$?" indicates a variable

    (THOR
                              ;THOR is like "or", trying things
                              ;in the order given until one works

      (THGOAL(#LONG $?X)(THUSE CONTENTS-CHECK COUNTPAGES))
                              ;THUSE says to try the theorem
                              ;named CONTENTS-CHECK first,
                              ;then if that doesn't work, try
                              ;the one named COUNTPAGES

      (THAND
                              ;THAND is like "and"

        (THGOAL(#CONTAINS $?X $?Y))
                              ;find something Y which is
                              ;contained in X

        (THGOAL(#ARGUMENT $?Y))
                              ;show that it is an argument

        (THGOAL(#PERSUASIVE $?Y)(THTBF THTRUE))))))
                              ;prove that it is persuasive, using
                              ;any theorems which are applicable
```

Figure 53 -- PLANNER Representation

ADVICE LEMMA

GOAL	VERIFY	
$(\lvert A\rvert \leq \varepsilon)$	$(A = B + C)$	1
	AND	
	$(\lvert B\rvert \leq \varepsilon_1)$	2
	AND	
	$(\lvert C\rvert \leq \varepsilon_2)$	3
	AND	
	$(\varepsilon_1 + \varepsilon_2 \leq \varepsilon)$	4
	OR	
	(OTHER ADVICE)	
	$(A = B \cdot C)$	1
	AND	
	$(\lvert B\rvert \leq \varepsilon_1)$	2
	AND	
	$\lvert C\rvert \leq \varepsilon_2)$	3
	AND	
	$(\varepsilon_1 \cdot \varepsilon_2 \leq \varepsilon)$	4
	OR	
	(OTHER ADVICE)	

EXAMPLE (Overbeek)

Th. $|a| \leq E^{\alpha}-1 \wedge |b| \leq E^{\beta}-1 \wedge 1+c=(a+1)(b+1)$
$\longrightarrow |c| \leq E^{\alpha+\beta}-1$

PROOF

Goals

$$c = \underbrace{ab}_{A} + \underbrace{a+b}_{B}$$

(1)	$c = A + B$	$a.b/A, (a+b)/B$
(2)	$\|a.b\| \leq \varepsilon_1$	
(21)	$a.b = A.B$	$a/A, b/B$
(22)	$\|a\| \leq \varepsilon_{11}$	$E^{\alpha}-1/\varepsilon_{11}$
(23)	$\|b\| \leq \varepsilon_{12}$	$E^{\beta}-1/\varepsilon_{12}$
(24)	$(E^{\alpha}-1)\cdot(E^{\beta}-1) \leq \varepsilon_1$	$(\quad)\cdot(\quad)/\varepsilon_1$
(3)	$\|a+b\| \leq \varepsilon_2$	
(32)	$\|a\| \leq \varepsilon_{21}$	$E^{\alpha}-1/\varepsilon_{21}$
(33)	$\|b\| \leq \varepsilon_{22}$	$E^{\beta}-1/\varepsilon_{22}$
(34)	$(E^{\alpha}-1)+(E^{\beta}-1) \leq \varepsilon_2$	$(E^{\alpha}-1)+(E^{\beta}-1)/\varepsilon_2$

(4) $(E^{\alpha}-1)\cdot(E^{\beta}-1) + ((E^{\alpha}-1)+(E^{\beta}-1)) \leq E^{\alpha+\beta}-1$

$E^{\alpha+\beta}-1 \leq E^{\alpha+\beta}-1$ TRUE

QED

Used: Simplification, Advice Lemma

ADVICE

GOAL

$P(C)$

VERIFY

FIND $P(A)$ IN HYPOTHESIS

AND

EXPRESS C IN TERMS OF A,

$C = F(A,B)$

AND

FIND $P(\alpha) \wedge P(\beta) \longrightarrow P(F(\alpha,\beta))$

AND

GOAL $P(B)$

OR

(OTHER ADVICE)

★ 10. PROCEDURES (and Built-in Concepts)

These have been discussed already, especially in the Non-standard Analysis example given in 7.

Slide 46 lists some of these concepts and examples. An "expert" is a set of procedures for solving one type of problem. See Goldstein [28].

In Slide 47 we see an INDUCTION heuristic being applied [7]. In general when a heuristic is to be applied, the program detects a pattern and consults a list of recommendations (See Slide 48). In this example it detects the presence of ω in the theorem being proved and proceeds as shown.

Slide 49 shows some strategies from Goldstein's geometry prover [28], and Slide 50 shows an example where the PAIRS heuristic [10] is being applied. In this example a partial match was obtained between the two formulas

$$\mathrm{Cover}(G_0) \quad \text{and} \quad \mathrm{Cover}(\bar{\bar{G}}_0) ,$$

which triggered the program to consult the PAIRS table (See Slide 51) for advice. The first advice given from the PAIRS table, namely $(G_0 \subseteq \bar{\bar{G}}_0)$, failed, but the second one, $(G_0 \subseteq\subseteq \bar{\bar{G}}_0)$, succeeded.

PROCEDURES

STRATEGIES

HEURISTICS

- SYNTACTIC
- SEMANTIC (DOMAIN DEPENDENT)

EXPERTS

<u>EXAMPLES</u>

- INDUCTION
- BUILT-IN PARTIAL AND TOTAL ORDERING, INEQUALITY, ASSOCIATIVITY, ETC.
- "SOLVERS"

- GOLDSTEIN'S GEOMETRY PROVER
- LIMIT HERUISTIC
- PAIRS HEURISTIC

<u>CONCEPTS</u>:

- FOLLOW A PLAN RATHER THAN SEARCH.
- CALCULATE AN ANSWER RATHER THAN PROVE A FORMULA.

INDUCTION HEURISTIC

THEOREM. $\omega = \bigcup_{\alpha \in \omega} \alpha$

(1) $(\omega \subseteq \bigcup_{\alpha\in\omega} \alpha) \wedge (\bigcup_{\alpha\in\omega} \alpha \subseteq \omega)$, DEFN OF =

SUBGOAL 1

(11) $(\omega \subseteq \bigcup_{\alpha\in\omega} \alpha)$ EASY

SUBGOAL 2

(12) $(\bigcup_{\alpha\in\omega} \alpha \subseteq \omega)$

$(T_0 \in \bigcup_{\alpha\in\omega} \alpha \rightarrow T_0 \in \omega)$ DEFN OF $\subseteq$

$(\alpha_0 \in \omega \wedge T_0 \in \alpha_0 \rightarrow T_0 \in \omega)$ REDUCE

PROOF FAILS BY THE NORMAL PROCEDURES. IT DETECTS THE PRESENCE OF ω: TRY INDUCTION. PRE-INDUCTION PUTS IT IN THE FORM:

$$(\alpha_0 \in \omega \rightarrow \underbrace{(T_0 \in \alpha_0 \rightarrow T_0 \in \omega)}_{P(\alpha_0)})$$

IT NOW TRYS: $\underset{.1}{P(0)}$ AND $\underset{.2}{(P(\alpha_0) \rightarrow P(\alpha_0 + 1))}$.

12.1 $(T_0 \in 0 \rightarrow T_0 \in \omega)$ SUCCEEDS

12.1 $(\alpha_0 \in \omega \wedge (T_0 \in \alpha_0 \rightarrow T_0 \in \omega) \rightarrow (\underbrace{T_0 \in (\alpha_0+1) \rightarrow}_{T_0 = \alpha_0 \vee T_0 \in \alpha_0} T_0 \in \omega))$.
SUCCEEDS

HEURISTIC RULE

1. DETECT A PATTERN

2. CONSULT A TABLE OF RECOMMENDATIONS

   (THINGS TO TRY)

GOLDSTEIN (1973)

STRATEGY EQTR13

TO-PROVE: Triangle XYZ = Triangle UVW

ESTABLISH: 10 seq XZ = seq VW

20 angle XYZ = angle UVW

30 angle YZX = angle VWU

REASON: congruance by ASA

(This is like backchaining)

CONVERSION ANGLE-BISECTOR

GIVEN: seq DB besects angle ABC

ASSERT: angle ABD = angle CDB

FORGET: given

(This is like reduce)

COROLLARY EQTRI-2

GIVEN: Triangle XYZ = Triangle UVW

ASSERT: Angle XYZ = angle UVW

angle YZX = angle VWU

angle ZXY = angle WUV

[illegible]

PAIRS Heuristic

Example

THEOREM. $\forall G\ (\text{Cover}(G) \rightarrow \text{Cover}(\bar{\bar{G}}))$

(1)
$$\begin{tikzcd} \text{Cover}(G_0) \arrow[dr, dash, dashed] & & \text{Cover}(\bar{\bar{G}}_0) \arrow[dl, dash, dashed] \\ & \text{No Match} & \end{tikzcd}$$

Partial Match: Use PAIRS Heuristic

Consult PAIRS Table under "Cover"

(1.1) Try $(G_0 \subseteq \bar{\bar{G}}_0)$ Fails

(1.2) Try $(G_0 \subseteq\subseteq \bar{\bar{G}}_0)$ "T" by REDUCE

or

$(A_0 \in G_0 \Rightarrow \underbrace{C \in \bar{\bar{G}}_0}\ \wedge\ A_0 \subseteq C)$ defn of $\subseteq\subsetneq$

$(A_0 \in G_0 \Rightarrow B \in G_0 \wedge C = \bar{B} \wedge A_0 \subseteq C)$

$(A_0 \in G_0 \Rightarrow B \in G_0 \wedge A_0 \subseteq \bar{B})$ sub =

SUBGOAL 1

(1.21) $(A_0 \in G_0 \Rightarrow B \in G_0)$ A_0/B

SUBGOAL 2

(1.22) $(A_0 \in G_0 \Rightarrow A_0 \subseteq \bar{A}_0)$ "T" by REDUCE

QED

DEFINITIONS

$\bar{A} \equiv$ The Closure of A. (note: $A \subseteq \bar{A}$).

$\bar{\bar{G}} \equiv \{\bar{A} : A \in G\}$

cover $(G, \bar{X}) \equiv \bar{X} \subseteq \bigcup_{\alpha \in G} \alpha$

$G \subseteq\subseteq F$ (G is a refinement of F)

$\equiv \forall A \in G \, \exists C \in F \, (A \subseteq C)$

REDUCE TABLE (Single Entry)

IN	OUT
$A \subseteq \bar{A}$	"T"
$G \subseteq\subseteq \bar{\bar{G}}$	"T"
⋮	

PAIRS Table

IN	Pattern	Recommendations
Cover	(Cover (G) $\rightarrow$ Cover (F))	$[\underset{.1}{(G \subseteq F)}\underset{.2}{(G \subseteq\subseteq F)} \cdots]$
Countable	(Countable A $\rightarrow$ Countable B)	$[\underset{.1}{(B \subseteq A)}$

$\underset{.2}{(\exists F\ (F \text{ is a function} \wedge \text{domain } F \subseteq A \wedge B \subseteq \text{range } F))} \cdots]$

⋮

11. MODELS

In Slide 27 we see the flow chart of Gelernter's geometry prover, with its famous "diagram filter", being used to discard unwanted subgoals. This is an excellent example of a MODEL or counterexample being used to help with a proof. Since models and counterexamples play such crucial roles in mathematics it is not surprising that they have been found useful in automatic provers. We expect their role to be expanded.

Slide 52 shows Reiter's Rule 4 (See [66]) and an explanation of how the model M is used in the execution of this rule; and Slide 53 gives an example of a theorem being proved by his system. Slide 54 gives an example of a group that might be used for his model.

The names of some others who have used Models and Counterexamples in automatic proofs are shown in Slide 55.

Reiter's Use of MODELS in MATH-HACK

4. $\psi \vdash A \wedge B$

If $\psi \vdash A$ returns σ_1, $M \models_E B\sigma_1$,

and $\psi \vdash B\sigma_1$ returns σ_2

$\sigma_1 \sigma_2$

Suppose that, during an attempted proof of A, x is instantiated by the term t. At this point, make the semantic test $M \models_E B(t)$. If successful, proceed with the proof of A. Otherwise, A's proof has obviously gone astray and must be redirected. Thus, rather than patiently waiting for A to deliver a (possibly wrong) σ_1, the wff B should be continuously semantically monitoring the proof of A, thereby minimizing the risk of receiving an incorrect σ_1. We believe that this kind of parallel processing of dependent sub-goals will consideraly alleviate the problem of back-up encountered by purely syntactic theorem-provers.

REITER EXAMPLE

THEOREM. IF S IS A SUBSET OF A GROUP SUCH THAT $xy^{-1} \in S$ WHENEVER x AND $y \in S$, THEN $x^{-1} \in S$ WHENEVER $x \in S$.

$$ex = x \wedge xe = x \wedge \overset{\alpha}{xx^{-1} = e} \wedge x^{-1}x = e$$
$$\wedge\, b \in S \wedge (x \in S \wedge y \in S \wedge xy^{-1} = z \rightarrow z \in S) \vdash b^{-1} \in S .$$

BACKCHAIN ON α TO GET THE SUBGOAL

(1) $\vdash$ $x \in S \wedge y \in S \wedge xy^{-1} = b^{-1}$

(11) $\vdash$ $x \in S$ $\quad$ b/x

(12) $\vdash$ $y \in S \wedge by^{-1} = b^{-1}$

(121) $\vdash$ $y \in S$ $\quad$ b/y

(122) $\vdash$ $bb^{-1} = b$

FAILS IN THE MODEL, SO BACK UP TO (1). REORDER SUBGOALS.

(11) $\vdash$ $xy^{-1} = b^{-1}$ $\quad$ $e/x,\ b/y$

(12) $\vdash$ $e \in S \wedge b \in S$

EASILY PROVED.

FOR REITER EXAMPLE

HERE THE MODEL M MIGHT BE THE KLEIN FOUR GROUP

$$\begin{array}{c|cccc} & e & A & B & C \\ \hline e & e & A & B & C \\ A & A & e & C & B \\ B & B & C & e & A \\ C & C & B & A & e \end{array}$$

IN WHICH GOAL (122), $BB^{-1} = B^{-1}$, CLEARLY FAILS. MORE COMPLICATED MODELS ARE NEEDED FOR OTHER PROOFS, ESPECIALLY WHERE COMMUTATIVITY IS NOT ASSUMED.

MODELS-COUNTEREXAMPLES

GELERNTER [26]	-	GEOMETRY
SLAGLE [72]	-	RESOLUTION
REITER [66]	-	GROUPS
NEVINS [58]	-	GEOMETRY
SIKLOSSY [70]	-	ROBOTS (DISPROVER)
WINOGRAD [84]	-	BLOCK'S WORLD
BALLANTYNE [3]	-	TOPOLOGY
HENSCHEN [33]	-	GROUPS

•
•
•

★ 12. ANALOGY

Perhaps the biggest error made by researchers in automatic theorem proving has been in essentially ignoring of the concept of analogy in proof discovery. It is the very heart of most mathematical activity and yet only Kling [39] has used it in an automatic prover. His paper showed how, with the use of knowledge, a proof in group theory could be used to help obtain a similar proof in ring theory.

We strongly urge that other workers in this field famaliarize themselves with Kling's work and extend and apply them more effectively.

★ 13. MAN-MACHINE

One of the most irksome things about current automatic theorem provers is the apparent need for the human user to prove the theorem himself before he gives it to the computer to do so. This is necessary because he must determine (for the computer) what axioms, or supporting theorems, are needed in the proof, and if he puts in too many, the proof will bog down. See [12, p.45].

See Slide 56.

This problem is partially eliminated by the use of the various concepts mentioned above, such as procedures and REDUCTION tables, which effectively carry the information needed from some of these reference theorems, and are able to give this information when needed without slowing the system down. The remainder of the difficulty can be eliminated by having the human user insert reference theorems only when they are needed.

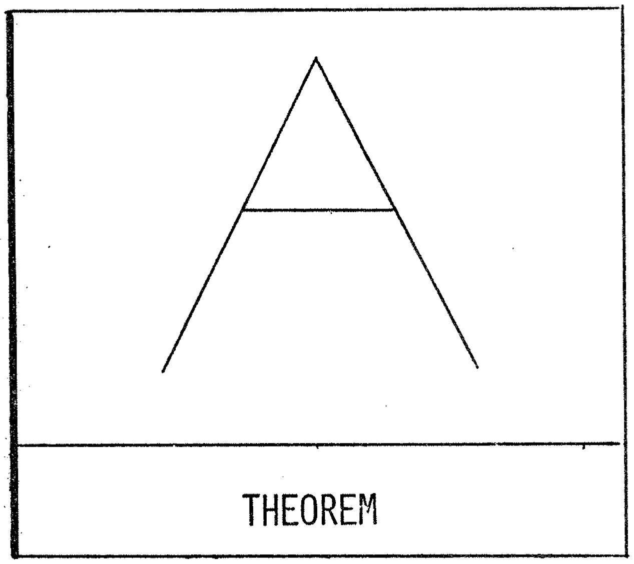
Also present systems cannot prove very hard theorems, so they don't get involved in interesting mathematics. We take as a maxim:

> Automatic provers will not compete successfully with humans for the next 100 years. Therefore the most effective systems will be those in which the computer acts as an <u>assistant</u> to the human user.

Thus it is imperative that this work attracts researchers from pure mathematics, and therefore, that interactive programs be made convenient for the <u>user</u>, not the programmer.

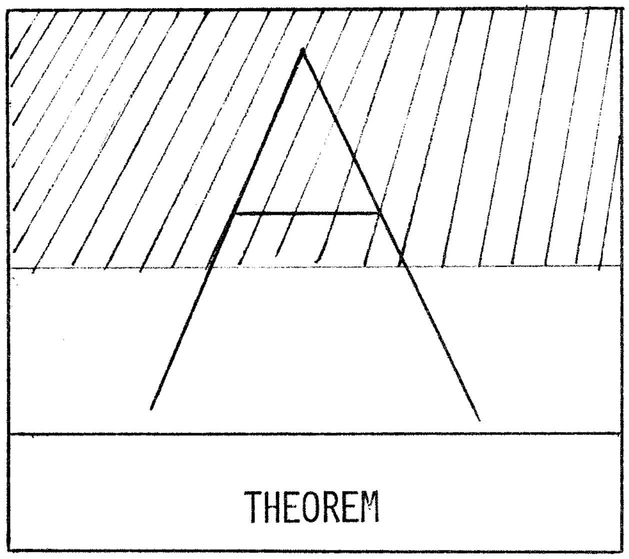
Some of the needs of the user mathematician are listed on Slide 57. Point 3 on Slide 57 is important because a mathematician will not long use a system which repeatedly requires him to give trivial information to the system.

We feel that a well-built system can be exercised on a <u>large</u> number of examples, thereby obtaining much valuable information on the utility of concepts in the program.



AXIOMS AND SUPPORTING THEOREMS NEEDED IN THE PROOF.

THEOREM BEING PROVED



BUILT IN PROCEDURES AND REDUCTION TABLES

GIVEN ONLY WHEN NEEDED

SLIDE 56

MATHEMATICIAN'S NEEDS

1. READ AND EASILY COMPREHEND THE SCOPE
2. FOLLOW THE PROOF
3. HELP COMPUTER ONLY WHEN NEEDED
4. AXIOMS AND REFERENCE THEOREMS
   - (I) BUILT-IN (SOME)
   - (II) OTHERS ADDED ONLY WHEN NEEDED
5. CONVENIENT COMMANDS

By running a large number of examples, the user can learn by experience, those places where he needs to improve the automatic part of the system, places where a little extra programming can greatly reduce the load on the human user.

This objective has been partly attained in an interactive program verification system [29,13] which has been running for the last year in Ralph London's laboratory of the Information Sciences Institute, Los Angeles, and is now also running at the University of Texas. Peter Bruell, Mabry Tyson, and Larry Fagan were instrumental in developing this system. Much more needs to be done on it to make it truly effective.

Others who have (earlier) worked on interactive systems include

Guard, et al [30]
Allen and Luckham [2]
Huet [37]

and others.

III. Programming Languages

The new programming languages, such as PLANNER [34], MICRO-PLANNER [78], QA-4 [68], Q-LISP [65], and PLASMA [35], which have been proposed and/or implemented during the last few years have much to offer automatic theorem proving. Especially are they rich in concepts such as: Knowledge, data base, procedures, goal oriented, automatic backup, pattern directed invocation, demons, data types.

Also these languages have built-in structures and controls to handle the kinds of things we propose.

However, we do not believe that the lack of use of these programming languages in current automatic prover has hurt their performance. No proof of a hard theorem has been omitted because the user did not use one of these. This may not remain to be the case as automatic provers get more sophisticated, and as these languages get more powerful and efficient. Many of their features are ready-made for provers, and we should move toward adopting them, with needed modifications, for our use.

IV. Comments

The reader should not get the idea that we have found _the_ secret to automatic theorem proving. We believe in these concepts but are certain that others will evolve.

We have _talked_ at lot and proved very _few_ hard theorems (by computer) during the last several years. It is time _do_, to show that our concepts are good. It is time to get a lot more _experience_ with our provers. This will allow us to eliminate some of our "good" ideas.

It is _not_ the time to give up on automatic theorem proving. How can that be rational at a time when so little has yet been done to try the ideas we already have? For example, why doesn't someone else use analogy in automatic proofs?

One thing that would help push this field ahead, would be for authors to follow the practice of publishing the proof of at least one _hard theorem_ in each new methods paper. We do not believe this field will remain vital unless we develop truly powerful provers, and not just theories.

Completeness in itself is not a bad concept, if handled correctly. For example, a complete unification system with built-in associativity and commutativity such as [77], needs to be reworked in a way that will make it a useful part of a practical prover. It is believed that a properly constructed _overdirector_ (See II.7) can so direct the search that one can have both efficiency and (essential) completeness. At least we can try for this.

"Trapping" remains a serious problem, whereby a substitution a/x that satisfies a goal $P(x)$ may fail on $Q(x)$, and hence on

$$(P(x) \wedge Q(x)) .$$

Backing-up theoretically solves this but can be very time consuming. Huet's "delaying" as used for matching in higher order logic [36] might be a good idea here.

Another worry is the "learning" problem. During the last decade most researchers in AI have avoided machine learning, because of such poor results from earlier experiments, and have favored the use of Man's "knowledge" in AI programs. However, eventually that barrier must be removed if the automatic prover is to be very effective. Slides 45 and 43 and accompanying comments provide an example of the kind of controlled learning that might be useful.

Other works such as studies on Induction (by Meltzer [53] and others) might be important to our efforts.

One should also not ignore proof checking as a potential use for automatic theorem proving [51,1,11], and also computer aided teaching of mathematics [47].

V. Challenges

Let me close by suggesting a few theorems, from various fields of mathematics, whose proofs by automatic means would be impressive at this time or in the near future. See Slides 58, 59.

In our efforts to mold our experience into an effective theorem prover, we are reminded of a 1918 statement by Albert Einstein [62]:

> Man tries to make for himself in the fashion that suits him best a simplified and intelligible picture of the world. He then tries to some extent to substitute this cosmos of his for the world of experience, and thus to overcome it. .....
> The supreme task...is to arrive at those universal elementary laws from which the cosmos can be built up by pure deduction. There is no logical path to these laws; only intuition, resting on sympathetic understanding of experience, can reach them...

FIELD	THEOREMS PROVED	CHALLENGE
SET THEORY	Elementary Set Theory $\omega = \omega \cap \text{Subsets}(\omega)$ $\omega = \bigcup_{\alpha \in \omega} \alpha$	• Schoeder-Bernstein Theorem
CALCULUS	Limit Theorems (with Limit Heuristic)	• Limit Theorems (ω/0 Limit Heuristic) • Rolle's Theorem • $\int f\,dx$ exists for f continuous
ANALYSIS	Bolzano Weierstrass Th. Cont. fcn on Compact set is Uniformly Cont. (Using Non-Standard Anal.)	• Bolzano Weierstrass Th. (ω/0 Non-Standard Anal.) • Cont. fcr on Compact Set is Uniformly Cont. (ω/0 Non-Standard Anal.) • Heine Borel Theorem • Hahn Banach Theorem

FIELD	THEOREMS PROVED	CHALLENGE
GEOMETRY	GELERNTER'S	• Pythagorean Theorem
TOPOLOGY	Open(A) $\wedge$ Open(B) $\longrightarrow$ Open(A $\cup$ B)	• A separable, normal space is metrizable • Tichenoff Theorem
ALGEBRA	GROUP Right Identity $x + (x + x) = 0 \longrightarrow$ $a + b + (-a) + b + a + (-b) + (-a) + (-b) = 0$	RING • $x^3 = 1$ for $x \neq 0$ $\longrightarrow a \cdot b = b \cdot a$

• $x^3 = 1$ for
$x \neq 0 \rightarrow a \cdot b = b \cdot a$

References

1. Paul W. Abrahams. Application of LISP to checking mathematical proofs. In The Programming Language LISP: its operation and applications, The MIT Press, Cambridge, Mass., 1966, pp. 137-160.

2. J. Allen and D. Luckham. An interactive theorem-proving program. Machine Intelligence, 5(1970), 321-336.

3. Michael Ballantyne. Computer generation of counterexamples in topology. The Univ. of Texas at Austin Math. Dept. Memo ATP-24, 1975.

4. Michael Ballantyne and William Bennett. Graphing methods for topological proof. The Univ. of Texas at Austin Math. Dept. Memo ATP-7, 1973.

5. A.M. Ballantyne and W.W. Bledsoe. Automatic proofs of theorems in analysis using non-standard techniques. The Univ. of Texas at Austin Math. Dept. Memo ATP-23, July 1975.

6. W. Bibel and J. Schreiber. Proof search in a Gentzen-like system of first order logic. Bericht Nr. 7412, Technische Universitat, 1974.

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

8. W.W. Bledsoe. The sup-inf method in Presburger arithmetic. Dept. of Math., The Univ. of Texas at Austin, Memo ATP-18. Dec. 1974. Essentially the same as: A new method for proving certain Presburger formulas. Fourth IJCAI, Tblisi, USSR, Sept. 3-8, 1975.

9. W.W. Bledsoe, R.S. Boyer, and W.H. Henneman. Computer proofs of limit theorems. Artif. Intell., Vol. 3, No. 1, pp. 27-60, Spring 1972.

10. W.W. Bledsoe and P. Bruell. A man-machine theorem-proving system. In Adv. Papers 3rd Int. Joint Conf. Artif. Intell., 1973, pp. 55-65; also Artif. Intell., Vol. 5, No. 1, pp. 51-72, Spring 1974.

11. W.W. Bledsoe and E.J. Gilbert. Automatic theorem proof-checking in set theory. Saudia Corp. Research Report, SC-RR-67-525, July 1967.

12. W.W. Bledsoe and Mabry Tyson. The UT interactive theorem prover. The Univ. of Texas at Austin Math. Dept. Memo ATP-17, May 1975.

13. W.W. Bledsoe and Mabry Tyson. Typing and proof by cases in program verification. The Univ. of Texas at Austin Math. Dept. Memo ATP-15, May 1975.

14. R.S. Boyer and J.S. Moore. Proving theorems about Lisp functions. J. Assoc. Comput. Mach., Vol. 22, pp. 129-144, Jan. 1975.

15. Frank Brown. (unfinished Ph.D. thesis on automatic theorem proving), Univ. of Edinburgh, 1975.

16. Peter Bruell. A description of the functions of the man-machine topology theorem prover. The Univ. of Texas at Austin Math. Dept. Memo ATP-8, 1973.

17. A. Bundy. Doing arithmetic with diagrams. In Adv. Papers 3rd Int. Joint Conf. Artif. Intell., 1973, 130-138.

18. R.L. de Carvalho. Some results in automatic theorem-proving with applications in elementary set theory and topology. Ph.D. Thesis, Dept. of C.S., Univ. of Toronto, Canada. Tech. Report No. 71, Nov. 1974.

19. C. Chang and R.C. Lee. Symbolic logic and mechanical theorem proving. Academic Press, 1973.

20. D.C. Cooper. Theorem proving in computers. Advances in Programming and Non-numeric Computation. (L. Fox, ed.), 155-182.

21. J.L. Darlington. Automatic theorem proving with equality substitution and mathematical induction. Machine Intelligence, 3(1968), 113-127.

22. L.P. Deutsch. An interactive program verifier. Ph.D. Thesis, University of California, Berkeley, 1973. Also Xerox Palo Alto Research Center Report CSL-73-1, May 1973.

23. George W. Ernst. The utility of independent subgoals in theorem proving. Information and Control, April 1971. A definition-driven theorem prover. Int'l. Joint Conf. on Artificial Intelligence, Standord, Ca., August 1973, 51-55.

24. D.H. Fishman. Experiments with a resolution-based deductive question-answering system and a proposed clause representation for parallel search. Ph.D. Thesis, Dept. of Comp. Sci., Univ. of Maryland, (1973).

25. Fronig. Private Communication Institut fur informatik, Universitat Bonn.

26. H. Gelernter. Realization of a geometry theorem-proving machine. Proc. Int'l. Conf. Information Processing, 1959, Paris UNESCO House, 273-282.

27. G. Gentzen. Untersuchungen uber das logische Schliessen I. Mathemat. Zeitschrift 39, 1935, 176-210.

28. Ira Goldstein. Elementary geometry theorem proving. MIT-AI Lab Memo 280, April 1973.

29. D.I. Good, R.L. London and W.W. Bledsoe. An interactive verification system. Proceedings of the 1975 International Conf. on Reliable Software, Los Angeles, April 1975, 482-492, and IEEE Trans. on Software Engineering 1(1975), 59-67.

30. J.R. Guard, F.C. Oglesby, J.H. Bennett and L.G. Settle. Semi-automated mathematics. J. ACM 16(1969), 49-62.

31. Patrick Hays, Forthcoming book on automatic theorem proving. University of Essex.

32. A.C. Hearn. Reduce 2: A system and language for algebraic manipulation. In Proc. Assoc. Comput. Mach., 2nd Symp. Symbolic and Algebraic Manipulation, 1971, 128-133; also Reduce 2 User's Manual, 2nd ed., Univ. of Utah, Salt Lake City, UCP-19, 1974.

33. Lawrence J. Henschen. Semantic resolution of horn sets. Advanced papers for IJCAI-75, Tbilisi, USSR, Sept. 1975.

34. Carl Hewitt. Description and theoretical analysis (using schemata) of PLANNER: a language for proving theorems and manipulating models in a robot. Ph.D. Thesis (June 1971). AI-TR-258 MIT-AI-Lab. April 1972.

35. Carl Hewitt. How to use what you know. MIT-AI Lab. working paper 93, May 1975.

36. G.P. Huet. Constrained resolution: a complete method for higher order logic. Ph.D. thesis, Case Western Reserve Univ. Jennings Computing Center Report 1117.

37. G.P. Huet. Experiments with an interactive prover for logic with equality. Report 1106, Jennings Computing Center, Case Western Reserve University.

38. J.C. King. A program verifier. Ph.D. dissertation, Carnegie-Mellon Univ. Pittsburgh, Pa., 1969.

39. R.E. Kling. A paradigm for reasoning by analogy. AI Jour. 2, (1971), 147-178.

40. D.E. Knuth and P.B. Bendix. Simple word problems in universal algebras. Computational Problems in Abstract Algebra. J. Leech, Ed., Pergamon Press, 1970, 263-297.

41. Dallas S. Lankford. Complete sets of reductions for computational logic. The Univ. of Texas at Austin Math. Dept. Memo ATP-21, Jan. 1975.

42. Dallas S. Lankford. Canonical algebraic simplification in computational logic. The Univ. of Texas at Austin Math. Dept. Memo ATP-25, 1975.

43. V.A. Lifshits. Specialization of the form of deduction in the predicate calculus with equality and function symbols. Proc. of the STEKLOV Inst. of Mathematics. No. 98(1968), 1-23.

44. Donald Loveland. Forthcoming book on mechanical theorem proving in first order logic. (Duke University).

45. Donald W. Loveland and M.E. Stickel. A hole in goal trees: some guidance from resolution theory. Proc. third Int'l. Joint Conf. on Art. Intel., Stanford, 1973, 153-161.

46. Bernard Luya. Un systeme complet de deduction maturelle. Thesis, University of Paris VII, Jan. 1975.

47. Vesko Marinov. (An interactive system for teaching set theory by computer at IMSSS, Ventura Hall, Stanford). Private Communication.

48. S. Ju Maslov. Proof-search strategies for methods of the resolution type. Machine Intelligence 6(1971), 77-90.

49. S. Ju Maslov. (1964) An inverse method of establishing deducibility in classical predicate calculus. Dokl. Nauk SSSR, 159, 17-20.

50. John McCarthy. Programs with common sense. (The advice taker). In Semantic Information Processing, Marvin Minsky (Ed.), 403-418.

51. John McCarthy. Computer programs for checking mathematical proofs. Proc. Amer. Math. Soc. on Recursive Function Theory, held in New York, April, 1961.

52. D.V. McDermott and Gerald J. Sussman. The CONNIVER reference manual. AI Memo 259. MIT-AI-Lab. (May 1972), (Revised July 1973).

53. Bernard Meltzer. The programming of deduction and induction. Univ. of Edinburgh, Dept. of Art. Int., DCL Memo 45, 1971. Also See AI Jour., 1(1970), 189-192.

54. Jack Minker, D.H. Fishman and J.R. McSkimin. The Q^* algorithm -- a search strategy for a deductive question-answering system. A.I. Jour., 4(1973), 225-243.

55. Marvin Minsky. A framework for representing knowledge. In P. Winston (Ed.), The Psychology of Computer Vision. New York: McGraw-Hill, in press.

56. Arthur J. Nevins. A human oriented logic for automatic theorem proving. MIT-AI-Lab Memo 268, Oct. 1972. JACM 21(1974), 606-621.

57. Arthur J. Nevins. A relaxation approach to splitting in an automatic theorem prover. MIT-AI-Lab. Memo 302, Jan. 1974. To appear in the AI Jour.

58. Arthur J. Nevins. Plane geometry theorem proving using forward chaining. MIT-AI-Lab. Memo 303, Jan. 1974.

59. A. Newell, J.C. Shaw and H.A. Simon. Empirical explorations of the logic theory machine: a case study in heuristics. RAND Corp. Memo P-951, Feb. 28, 1957. Proc. Western Joint Computer Conf. 1956, 218-239. Computers and Thought, Feigenbaum and Feldman (Eds.), 134-152.

60. A. Newell, J.C. Shaw and H.A. Simon. Report on a general problem-solving program. RAND Corp. Memo P-1584, Dec. 30, 1958.

61. Nils Nilsson. Artificial Intelligence. (Including a review of automatic theorem proving.) IFlP, Stockholm, Sweden, 1974.

62. Robert M. Pirsig. Zen and the art of motorcycle maintenance, pp. 106-7.

63. G.D. Plotkin. Building equational theories. Machine Intelligence 7, 1972, 73-89.

64. D. Prawitz. An improved proof procedure. Theoria 25, 102-139, (1960).

65. Rene Reboh and Earl Sacerdoti. A preliminary QLISP manual. Stanford Research Inst., A.I. Center Tech. Note 81, August 1973.

66. Raymond Reiter. A semantically guided deductive system for automatic theorem proving. Proc. Third Int'l. Joint Conf. on Art. Intel., 1973, 41-46.

67. Raymond Reiter. A paradigm for automated formal inference. To be presented at the IEEE theorem proving workshop, Argonne Nat'l. Lab., Ill., June 3-5, 1975.

68. J.R. Rulifson, J.A. Derksen and R.J. Waldinger. "QA4: a procedural calculus for intuitive reasoning". Standord Res. Inst. Artif. Intell. Center, Standord, Calif., Tech. Note 13, Nov. 1972.

69. Robert S. Shostak. On the completeness of the sup-inf method. Stanford Research Institute. Report 1975.

70. L. Siklossy and J. Roach. Proving the impossible is impossible is possible: disproofs based on hereditary partitions. IJCAI-73, 383-387.

71. L. Siklossy, A. Rich and V. Marinov. Breadth-first search: some surprising results. A.I. Jour., 4(1973), 1-28.

72. J.R. Slagle. Automatic theorem proving with renamable and semantic resolution. JACM, 14(1967), 687-697.

73. J.R. Slagle. Automated theorem-proving for theories with simplifiers, commutativity and associativity. JACM, 21(1974), 622-642.

74. J.R. Slagle. Automatic theorem proving with built-in theories of equality, Partial Order and Sets. JACM, 19(1972), 120-135.

75. J.R. Slagle and L. Norton. Experiments with an automatic theorem prover having partial ordering rules. CACM, 16(1973), 682-688.

76. L.M. Norton. Experiments with a heuristic theorem-proving program for the predicate calculus with equality. A.I. Jour., 2(1971), 261-284.

77. Mark Stickel. A complete unification algorithm for associative-commutative functions. Advanced papers for IJCAI-75, Tbilisi, USSR, Sept. 1975, 71-76.

78. G.J. Sussman, T. Winograd and E. Charniak. Micro-planner manual. MIT-AI Lab. Memo 203A, Dec. 1971.

79. N. Suzuki. Verifying programs by algebraic and logical reduction. Proc. Int'l. Conf. on Reliable Software, 1975, 473-481.

80. Mabry Tyson. An algebraic simplifier. The Univ. of Texas at Austin Math. Dept. Memo ATP-26, (to appear).

81. R.J. Waldinger and K.N. Levitt. Reasoning about programs. Artif. Intel., 5(1974), 235-316.

82. Hao Wang. Toward mechanical mathematics. IBM J. Res. Dev. 4(1960), 224-268.

83. Steven K. Winker. Complete demodulations in automatic theorem proving. University of Northern Illinois, Computer Science Department, July 1975.

84. Terry Winograd. Procedures as a representation for data in a computer program for understanding natural language. Ph.D. Thesis, MIT. MAC-TR-84, Feb. 1971.