MEMORANDUM
RM-3731-CC
JUNE 1963

THE LOGIC THEORY MACHINE:
A MODEL HEURISTIC PROGRAM

Einar Stefferud

PREPARED FOR:
CARNEGIE CORPORATION
THE LOGIC THEORY MACHINE:
A MODEL HEURISTIC PROGRAM

Einar Stefferrud

This research is sponsored by the Carnegie Corporation. Views or conclusions contained in this Memorandum should not be interpreted as representing the official opinion or policy of the Carnegie Corporation.
PREFACE

This Memorandum has been prepared to fill the need for a model program for use in teaching Information Processing Language-V (IPL-V). Experience in teaching IPL programming has shown that class discussion of a good program, developed as a pedagogical tool, is an essential ingredient for comprehending IPL applications and potentialities.

The IPL computer programming language was originally developed at The RAND Corporation, under U. S. Air Force Project RAND, and at Carnegie Institute of Technology, for expressing complex computer programs. IPL is being used to great advantage in dealing with problems requiring flexible memory structures and hierarchies of subroutines and data. IPL is also being applied in the field of artificial intelligence in studies of complex information processing, and by psychologists who are using computer simulation of human cognitive processes.

A number of colleges and universities now offer courses in IPL-V coding, and more such courses appear imminent. IPL coding was taught at the 1962 Summer Heuristic Programming Institute, held at The RAND Corporation and sponsored by the Carnegie Corporation. At that time, several interesting IPL programs were available, but none could serve as a pedagogical model. The Logic Theory Machine (LT) was determined to be the best candidate for such a model, and a new version, documented in this Memorandum, was developed. Bert F. Green and Fred M. Tonge consulted on the development of the program and preparation of the Memorandum.
Use of LT presumes familiarity with the Information Processing Language-V Manual (1) and particularly with Part Two, "Programmers' Reference Manual." In most cases, LT will be used in conjunction with a study of Part One of the Manual, "The Elements of IPL Programming."†

This Memorandum was made possible by funds granted by Carnegie Corporation of New York. The statements made and views expressed are solely the responsibility of the author.

†LT may be used by advanced students who have completed Part One of the Manual, or may be used simultaneously with Part One to illustrate the implementation in a complete program of the concepts being developed.
SUMMARY

This Memorandum contains a highly detailed program listing for the Logic Theory Machine (LT), a computer program written in Information Processing Language-V (IPL-V), and developed especially for use as a pedagogical model. The text portions of the Memorandum expand upon the documentation in the listing, tracing program flow, analyzing routines utilized, and providing insight into the structure and the development of the program.

LT was originally programmed in an early version of IPL by Newell, Shaw, and Simon\(^{(2-4)}\) to derive proofs of logic expressions in the sentential calculus of Whitehead and Russell.\(^{(5)}\) In rewriting the program for use as a teaching aid, a new method of replacement on subexpressions has been included, and many minor changes effecting improvements in clarity have been incorporated. Features of the code that were unjustifyably hard to explain have been simplified.

The Memorandum defines LT's activity in terms of problem solving, and then a representation of the defined problem is given in terms of IPL-V. Finally, what LT does is discussed in terms of process hierarchies which operate on the list structure representations of logic expressions.

LT can be implemented on any computer for which an IPL-V processor is available.\(^{1}\)

\(^1\)These include the IBM 650, 704, 709, 7090, 7094, Philco 2000, Bendix G-20, CDC 1604, UNIVAC 1105, and the AN/FSQ-32. A system for the Burroughs 220 is under development. LT, or any IPL-V program written in accordance with the IPL-V Manual, can be executed on any of these machines. In order to facilitate student modification of LT, information on obtaining the program deck can be obtained by writing The RAND Corporation.
# CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>PREFACE</strong></td>
<td>iii</td>
</tr>
<tr>
<td></td>
<td><strong>SUMMARY</strong></td>
<td>v</td>
</tr>
<tr>
<td>Section</td>
<td>I.  <strong>INTRODUCTION</strong></td>
<td>1</td>
</tr>
<tr>
<td>Section</td>
<td>II.  <strong>WHAT LT DOES</strong></td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>The Theorem-Proving Problem</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Proof Examples</td>
<td>4</td>
</tr>
<tr>
<td>Section</td>
<td>III. <strong>REPRESENTATION OF &quot;THE PROBLEM&quot;</strong></td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>in the IPL-V MACHINE</td>
<td></td>
</tr>
<tr>
<td>Section</td>
<td>IV.  <strong>HOW LT WORKS</strong></td>
<td>8</td>
</tr>
<tr>
<td>Section</td>
<td>V.   <strong>SUBSTITUTION AND MATCHING</strong></td>
<td>16</td>
</tr>
<tr>
<td>Section</td>
<td>VI.  <strong>THE OTHER METHODS</strong></td>
<td>24</td>
</tr>
<tr>
<td>Section</td>
<td>VII. <strong>UTILITY MEASURES</strong></td>
<td>33</td>
</tr>
<tr>
<td>Section</td>
<td>VIII. <strong>INFORMATION STORAGE AND RETRIEVAL</strong></td>
<td>38</td>
</tr>
<tr>
<td>Section</td>
<td>IX.  <strong>LOWER-LEVEL ROUTINES</strong></td>
<td>50</td>
</tr>
<tr>
<td>Section</td>
<td>X.   <strong>INPUT-OUTPUT ROUTINES</strong></td>
<td>62</td>
</tr>
<tr>
<td>Section</td>
<td>XI.  <strong>RUN EXECUTIVES AND DEBUGGING ROUTINES</strong></td>
<td>67</td>
</tr>
<tr>
<td>Section</td>
<td>XII. <strong>A SAMPLE RUN</strong></td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>Input Deck</td>
<td>70</td>
</tr>
<tr>
<td></td>
<td>Output</td>
<td>77</td>
</tr>
<tr>
<td>Section</td>
<td>XIII. <strong>LEVELS OF VOCABULARY AND USE OF SYMBOLS</strong></td>
<td>119</td>
</tr>
<tr>
<td>Section</td>
<td>XIV. <strong>COMPLETE VOCABULARY LISTING</strong></td>
<td>122</td>
</tr>
<tr>
<td>Section</td>
<td>XV.  <strong>COMPLETE PROGRAM LISTING</strong></td>
<td>129</td>
</tr>
<tr>
<td></td>
<td>REFERENCES</td>
<td>185</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

Figure
1. The Single-Problem Executive ......................... 12
2. Sample Subproblem Tree ............................... 13
3. The Substitution Method .................. 17
4. Test for Match with Substitution .......... 18
5. The Match Process ............................... 18
6. The Match Subprocess ...................... 19
7. Match with Substitution and Output
   Expanded Substitution List ................. 21
8. Expand Substitution List ......................... 21
9. Delineate Expression .......................... 22
10. Delineate Free Variable ......................... 23
11. Replace Free Variable by Delineated
    Substitutor ................................ 23
12. Detachment Method ............................... 25
13. Detachment Method Subprocess ............ 26
14. Sublevel Replacement Method I ............ 27
15. Try to Replace Located Segment .......... 28
16. Try to Replace Segment by Matching
    Left (Right) Sides ............................. 29
17. Finish Building New Subproblem ............ 31
18. Measure Utility ................................ 34
19. Add to Found List .............................. 35
20. Test Match of Total Expressions Without
    Substitution .................................. 36
21. Test Match of Segments Without
    Substitution .................................. 37
22. Axiom Map ....................................... 41
23. Axiom Map in Keypunched Form ............ 42–43
24. Add Expression to Map ......................... 44
25. Create List of Feasible Expressions ........ 46
26. Axiom Map with Overlaid Problem .......... 47
27. Find Number of Levels ....................... 52
28. Count Sublevels Recursively ................ 53
29. Find Number of Distinct Variables .......... 54
30. Generate Locations of Bound Variables ... 55
31. Generate Segment Locations from Given
    Level of Expression .......................... 56
32. Generate Subsegment Locations from
    Segment ........................................ 57
33. Create Main Segment from List Expression .. 59
34. Create Next Segment from List Expression .. 60
35. Read Next Logic Expression .................. 64–65
I. INTRODUCTION

Experience in teaching IPL (Information Processing Language)† programming has shown that class discussion of a good model program is an essential ingredient for comprehension of applications and potentialities of the IPL concepts. This documentation of the "Logic Theory Machine,"(2) otherwise known as LT, is specifically aimed at filling the need for such a program.

LT was the best candidate for development as a pedagogical model because 1) it is based in the readily understood context of theorem proving in sentential calculus; 2) literature on theorem proving is readily available;(3,4) and 3) the original LT, written in an early version of IPL by Newell, Shaw, and Simon, established the field of heuristic programming. Because it is a valuable example for students to examine and modify, LT has survived beyond its usefulness as a research tool.

The original version of LT was converted to IPL-V from IPL-II by Fred Tonge, and later converted into the present pedagogical model. The questions originally raised by LT are still valid and interesting, and although conversion has introduced some changes in LT's procedures, its structure remains essentially unchanged and its operating performance is nearly identical to that of the original.

LT does not represent an effort to obtain high machine efficiency. It is rather an effort to take ad-

† A discussion of the development of the IPL's can be found in the introduction to the IPL-V Manual. (1)

‡ The latter conversion was made by the author.
vantage of the powers of symbolization via heuristic symbol manipulation techniques. It is not very successful in comparison with people or even with some computer programs.\(^{(6,7)}\) (For an excellent treatment of this topic, see Minsky's discussion of problem solving.)\(^{(8)}\)

Since this Manual is designed for class use by students of list processing languages, the discussions and descriptions of LT assume familiarity with IPL-V. Several viewpoints are adopted for discussion purposes: first, what LT does is defined in terms of problem solving; then, representation of the defined problem is described in terms of IPL-V; and finally, what LT does is discussed in terms of process hierarchies which operate on the list structure representations of logic expressions.

The text describing LT is not intended to completely describe the program. Sections XIII through XV contain listings of the entire program along with supporting vocabulary listings. The text and flow diagrams are only intended to introduce the student to the program listing, which provides sufficient annotation to allow him to dissect LT's processes and learn how IPL-V can be used for complex information processing.

To facilitate detailed inspection, the entire program has been carefully and extensively documented in the comment fields. Outlines and flow diagrams have been supplied for some key routines, but others have been left for the student to construct, since learning to write down routine specifications is one of the most important facets of learning to use IPL-V.
II. WHAT LT DOES

THEOREM-PROVING PROBLEM

LT derives proofs of logic expressions in the sentential calculus of Whitehead and Russell.\(^{(5)}\) To do this, it uses the following entities:

1) Expressions, compounded from:
   a) free variables:
      \[ A, B, C, D, E, F, G \]
   b) bound variables:
      \[ P, Q, R, S, T \]
   c) connectives:
      \[
      \begin{align*}
      &- & \text{intuitive meaning:} & \text{NOT} \\
      &\lor & \text{intuitive meaning:} & \text{OR} \\
      &\rightarrow & \text{intuitive meaning:} & \text{IMPLIES} \\
      &\ast & \text{intuitive meaning:} & \text{AND} \\
      &\equiv & \text{intuitive meaning:} & \text{EQUIVALENT TO} \\
      \end{align*}
      \]

2) Axioms (expressions given as true):
   \[
   \begin{align*}
   *1.2 & \quad [A \lor A] & \equiv A \\
   *1.3 & \quad A \lor [B \lor A] & \equiv [B \lor A] \\
   *1.4 & \quad [A \lor B] \equiv [B \lor A] \\
   *1.5 & \quad [A \land [B \land C]] \equiv [B \land [A \land C]] \\
   *1.6 & \quad [A \land [C \land [B \land C]]] \equiv [C \land [B \land A]] \\
   \end{align*}
   \]

3) Definitions (expressions defining connectives):
   \[
   \begin{align*}
   *1.01 & \quad [A \equiv [\neg \neg A]] & \equiv [A \lor [A \land \neg A]] & \equiv [A \land [A \lor \neg A]] & \equiv [A \land [A \lor \neg A]] & \equiv [A \lor [A \land \neg A]] & \equiv [A \lor [A \land \neg A]] & \equiv [A \lor [A \land \neg A]] & \equiv [A \lor [A \land \neg A]] \\
   \end{align*}
   \]

\(^{\dagger}\) The distinction between bound and free variables is needed to distinguish variables of true expressions from variables of unproved expressions.
4) Problems (expressions proposed as provable theorems):
   *2.08  PIP
   *2.14  --PIP
   *2.45  -(PVQ)I-P
   *3.22  (P*Q)I(Q*P)
   *3.24  -(P*-P)
   *4.20  P=P
   etc.

5) Methods based on rules of inference:
   Substitution (for free variables)
   Replacement (through definitions)
   Detachment (A and AIB=B)
   Chaining (AIB and BIC=AIC)

PROOF EXAMPLES

TO PROVE:  *2.08  PIP
            *1.3  AI[BVA]                Given
            PIP[PVP]                Substitution
            *1.2  [AVA]IA               Given
            [PVP]IP                Substitution
            *2.08  PIP                Chaining
            Q.E.D.

Chaining uses the two expressions PIP[PVP] and [PVP]IP to yield PIP.

TO PROVE:  *2.14  --PIP
            *2.13  AV---A                Given
            PV---P                Substitution
            *1.4  [AVB]I[BVA]            Given
            [PV---P]I[---PVP]        Substitution
            ---PVP                Detachment
            *1.01  [AIB]..[-AVB]          Given
            [---PIP]..[---PVP]      Substitution
            *2.14  --PIP                Replacement
            Q.E.D.

This proof requires that *2.13 be previously proved or given as true.

† Chaining is not given as a rule in Whitehead and Russell, but is provable from the axioms. Other such methods might be developed, but chaining is the only one included in LT.
III. REPRESENTATION OF "THE PROBLEM" IN THE IPL-V MACHINE

Expressions, variables, and connectives are represented by IPL-V data list structures. An expression is recursively defined as a single variable or a list of subexpressions with a connective in its head. For example, Axiom *1.5 [AV[BVC]]I[BV[AVC]], looks like this:

Total Expression
 *15 9-0 Description list
  9-1 0 Main expression
  9-0 0

Description list
 Q15 Attr. "tree form"

Main Expression
  9-2 21*1.5 Value is data term containing text
  9-1 10 Main connective

Left Expression
  9-3 0 Left subexpression
  9-4 0 Right subexpression
  9-3 V0 Connective OR

Right Expression
  B0 Variable B
  9-6 0

Right of left
  B0
  C0 0

Right of Right
  A0
  C0 0 Variable C
Variables and connectives look like this:

<table>
<thead>
<tr>
<th>Connective</th>
<th>I0</th>
<th>9-0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMPLIES</td>
<td>9-0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Attr. &quot;type&quot;</td>
<td>Q14</td>
<td>J4</td>
<td>Q7</td>
</tr>
<tr>
<td>Value &quot;non-unary&quot;</td>
<td></td>
<td></td>
<td>9-1</td>
</tr>
<tr>
<td></td>
<td>9-1</td>
<td>21I</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Connective</th>
<th>V0</th>
<th>9-0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>OR</td>
<td>9-0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Q14</td>
<td>J4</td>
<td>Q7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9-1</td>
</tr>
<tr>
<td></td>
<td>9-1</td>
<td>21V</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Free Variable &quot;A&quot;</th>
<th>A0</th>
<th>9-0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attr. &quot;Variable&quot;</td>
<td></td>
<td>Q5</td>
<td>Q5</td>
</tr>
<tr>
<td>Attr. &quot;Free Var.&quot;</td>
<td>Q6</td>
<td>Q6</td>
<td>Q7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9-1</td>
</tr>
<tr>
<td></td>
<td>9-1</td>
<td>21A</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Free Variable &quot;B&quot;</th>
<th>B0</th>
<th>9-0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>9-0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Q5</td>
<td>Q5</td>
<td>Q6</td>
</tr>
<tr>
<td></td>
<td>Q6</td>
<td>Q7</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9-1</td>
</tr>
<tr>
<td></td>
<td>9-1</td>
<td>21B</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Free Variable &quot;C&quot;</th>
<th>C0</th>
<th>9-0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>9-0</td>
<td>Q5</td>
<td>Q5</td>
</tr>
<tr>
<td></td>
<td>Q6</td>
<td>Q6</td>
<td>Q7</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>9-1</td>
</tr>
<tr>
<td></td>
<td>9-1</td>
<td>21C</td>
<td></td>
</tr>
</tbody>
</table>
The rules of inference are embodied in routines called methods:

- M1 1 Detachment
- M2 Substitution
- M3 Replacement
- M4 Forward Chaining
- M5 Backward Chaining
- M6 Subexpression Replacement I
- M7 Subexpression Replacement II

Methods are applied to problems (unproved expressions) by the executive routines:

- M1 Single-Problem Executive
- M2 Multiple-Problem Executive
- M7 Apply Methods (1) to Problem (0)
- M8 Create a List of Methods for (0)

Application of methods to a problem results in symbol manipulation on the data list structures representing the problem and will, hopefully, result in finding a proof.
IV. HOW LT WORKS

In LT, the single problem executive uses the methods to find proofs for given problems. The methods are based on rules of inference set forth in *Principia Mathematica*. (5)

The substitution method tries to directly prove a given problem expression by matching it to an axiom expression or a previously proved theorem expression. The matching procedure tries to effect identity between the two expressions with an appropriate series of substitutions for free variables. If the match succeeds, a proof has been found.

The other methods do not try to find proofs directly. Instead, they try to construct subproblem expressions to serve as surrogates for the given problem. By construction, each new surrogate subproblem, when proved, will imply proof of the given problem from which it was developed. Substitution is immediately tried on each new subproblem in the hope that a proof is at hand.

The replacement methods try to develop subproblem expressions by replacing logical connectives as specified by the definitions. For example:

"Definition [AIB].=.[~AVB] and problem -PVP yield subproblem PIP."

PIP and -PVP are the same assertion in alternate forms as specified by definition *1.01.

The method of detachment is based on the rule of detachment (*1.11), and as in replacement, it tries to develop a surrogate for the problem expression. The rule of detachment:
"True expressions AIB and A yield new 
true expression B"
is used in a backward sense:

"True expression AIB and problem B yield 
subproblem A"
so that proof of the subproblem will imply proof of the 
problem from which it was derived. The substitution 
method is applied to each new subproblem immediately 
after it is developed.

Chaining is not set forth as a rule of inference, 
but its legitimacy as a method is provable from Axiom *1.5 
or Theorems *2.05 and *2.06 by detachment. To be 
strictly legal, chaining should not be used unless *2.05 
and *2.06 are in the set of true expressions. Appropriately, 
LT can prove all theorems through *2.06 without use of 
chaining.

The methods of chaining also produce subproblem 
expressions by working backward. For example:

"Problem AIC and theorem AIB yield sub-
problem BIC."

In this example, forward chaining works backward to obtain 
a new subproblem which, if proved, implies proof of the 
given problem. Backward chaining works backward in a 
similar manner:

"Problem AIC and theorem BIC yield sub-
problem AIB."
The main heuristic\(^\dagger\) in LT is this procedure of working backward. It is easy to see that the methods could work forward using only true expressions to develop more true expressions, testing each new one to see if it proves the given problem. This procedure would make LT's behavior independent of the given problem, up to the time of proof completion, and it would not perform any better than the British Museum Algorithm.\(^3\)

Working backward gives LT some vital sense of direction by taking advantage of the "heuristic connection"\(^\ddagger\) of its problem space. If LT is viewed as a trial solution generator with a solution tester, it is easy to see that the generator should have some sense of how to produce good trial solutions. LT's methods get this necessary sense of direction by working backward.

Substitution is the only method that finds proofs directly; thus, it serves as LT's trial solution tester. It is immediately applied to given problems and is applied as a subprocess to new subproblems as they are developed by the other methods.

The other methods serve as trial solution generators. New subproblems that do not lead directly to a proof are set aside in the untried subproblem list to be selected later for additional application of the other methods.

\(^\dagger\text{By heuristic, we mean, "Any principle or device that contributes to the reduction in the average search to solution." (9)}\)

\(^\ddagger\text{For an excellent discussion of "heuristic connection," see Marvin Minsky on, "The Problem of Search." (10)}\)
Selection of problems and application of methods is controlled by the single-problem executive routine, M1 (Fig. 1). This routine applies substitution first to avoid wasting effort on a directly provable problem. If substitution fails, the other methods are used to build trial proof sequences by developing subproblems. The method of derivation, the true expression used, and the problem from whence it came, are associated with each new subproblem.

The collection of subproblems develops into a tree of hypothesized proof sequences in which any proved subproblem constitutes proof of the given problem and of all intervening subproblems. Since all new subproblems have been through the substitution method, only subproblems at the outer reaches of the tree are candidates for further effort. These problems are kept on the untried problem list. M1 takes subproblems from the untried problems list, while the methods add new subproblems to it. Figure 2 shows an example of a subproblem tree.

It is interesting to note that while M1 works iteratively, the proof sequence tree grows recursively. This occurs because the context (derivation information) of each subproblem is directly associated with the subproblem itself, while the names of worthwhile candidates for additional effort are kept separately on the untried problems list. This arrangement allows the problem executive to work on whatever part of the tree it decides is most profitable looking.

If M1 worked recursively, LT would attack subproblems in the order of their development, which would involve a
Begin M1 [Executive for given problem (0)]

Set up for new problem (M3)

Test utility of given problem (M43)
  if O.K.             if N.G.          → Print rejected problem (M81)

Print "TO PROVE" given problem (M78)    Quit (J3)

Try substitution on given problem (M12)
  if no proof         if proof found    → Print proof (M71)

Select list of other methods (M8)

Apply other methods to problem (M7)
  (Each method adds any new
   found problems to the untried list)
  if no proof found    if proof found

Test if any limits exceeded (M90)
  if not              if yes

Find next subproblem (M60)
  if found            if failed        → Print failure (M72)

Print subproblem (M70)               Quit (J3)

Fig. 1--The Single-Problem Executive
TO PROVE:

Fig. 2--Sample Subproblem Tree
depth-first attack on the problem, and make it difficult, if not impossible, to temporarily shift attention to more interesting parts of the tree. The subproblem tree in some sense represents what is known about the problem and it seems reasonable that good problem-solving procedures will require some ability to take an overall look at "what can be done" in order to plan future activity.†

The procedures embodied in M1 raise a number of questions for which good answers are hard to find. For example, it is unreasonable for each method to try all available theorems, since most of them won't yield any progress. It is easy to answer that a good selection process is needed, but this is not very precise.

This problem is closely connected to the relevancy question in information retrieval research. A more precisely stated answer might be, "Design a theorem storage and retrieval system that will deliver the names of theorems, appropriate parts of which are feasible matches for a given expression." The word "appropriate" takes on a specific meaning for each method using the system.

There are many ways to implement such a system. An interesting one is implemented around routines M54 and M63 of LT. It is discussed in Sec. IX.

There are other interesting questions raised by M1, as enumerated in the following list:

1) What is meant by utility and how can it be measured?

2) How should substitution be implemented?

†See Refs. 11 and 12 for recent discussions of the implications of questions raised here.
3) How should the other methods be implemented?
4) How are methods to be selected?
5) What theorems should each method try?
6) How many theorems should each method try?
7) What kind of effort limits are meaningful? Useful?
8) How should the next subproblem be selected?

The remaining sections describe ways in which these questions have been resolved in the current version of LT. They are not the only or the best ways but they do enable LT to do a passable job of theorem proving.
V. SUBSTITUTION AND MATCHING

Matching is the heart of LT. The substitution method, M12 (Fig. 3), uses test for match routine M114 (Fig. 4) to try to effect identity between a problem and a theorem. M114 uses substitution for free variables as appropriate.

It may seem confusing, but substitution method M12 does not actually do any substitution. It only acts as an executive for matching a given problem with a sequence of appropriate theorems until a match is found or the theorem supply is exhausted. Determination of required substitutions is made by the match process.

Test for match routine, M114, is also an executive. It "puts off the work" to match process M11 (Fig. 5). M114's only reason for existence is to discard the output of M11 if a match is successful. Remember, TEST routines should leave no outputs in H0.

M11 does its matching with subprocess 9-100 (Fig. 6) after setting up an empty substitutions list. Since an expression is recursively defined in LT, the match subprocess is recursive. It tests for identity of variables and connectives between expressions, arranges effective substitution for free variables as required, and recursively matches corresponding subexpressions.

Required substitutions are effected through use of the substitution list. The match process adds new required substitutions to the list as they are discovered and looks up previously assigned substitutions when free variables are encountered in expressions. The substitution
Begin M12 [Substitution method for problem (0)]

Define Context (J43)

Find main expression (J81)

Tally substitution counter (J125)

Create list of feasible theorems (M63)

Apply 9-100 to each theorem of the feasible list (J100)

---[Subprocess 9-100 of M12]---

Make free variables of problem disjoint with theorem (M110)

Find main expression of theorem (J81)
  if found ——— if none ——— Quit + for generator

Test main expressions for match with substitution as required (M114, Fig. 4)
  if matched ——— if failed ——— Quit + for generator

Assign proof (J11)

Output problem and quit - for generator

Erase feasibles list (J71)

Clear context (J33)

Reverse H5 from generator and quit M12 (J5)

---

Fig. 3--The Substitution Method
Begin M114 [Test if segments (0), (1) match with substitution as required]

Match segments (M111, Fig. 5)
if matched if failed Quit -

Erase substitution list (J71), quit +

Fig. 4--Test for Match with Substitution

Begin M111 [Match segments (0), (1) with substitution as required. Output compact substitution list if successful]

Create empty substitution list (J90)

Match with substitution using created list (9-100, Fig. 6)
if matched if failed

Output created list, quit + Erase created list (J71), quit -

Fig. 5--The Match Process
Begin 9-100 of MILL [Match segments (0), (1). Use substitution list in 9-10, adding substitutions as required]

Define context (J31)

9-104: Test if LW0 is a variable (P9) no 9-101: Test if LW1 is a variable (P8) yes Test if LW1 is a free variable (P9) yes

9-110: Test if connectives 2W0, 2W1 are identical (J2) no

9-111: Find substitutor of LW0 (J10)

9-113: Locate next subsegment of LW0 (J60). Result to LW0 located

9-112: Locate next subsegment of LW1 (J60). Result to LW1 located

Match subsegments using 9-100 recursively

H5+ matched

H5- failed

Quit -

2W0 to (0) located

2W1 to (1) located

Locate next subsegment of LW1 (J60). Result to LW1 located

Quit +

Quit -

Quit +

Test if LW0 is a free variable (P9) no 9-102: Test if LW1 is a variable (P8) yes Test if LW1 is a free variable (P9) yes

no

no

no

no

yes

yes

yes

yes

no

 ASSIGN LW0 as substitutor for LW0 (J11)

Replace LW0 by its substitutor

no

yes

no

yes

 ASSIGN LW0 as substitutor for LW1 (J11)

Replace LW1 by its substitutor

clear context (J31), terminate with H5-

clear context (J31), terminate with H5+

Fig. 6--The Match Subprocess
list is a description list with free variables for attributes and assigned substitutions as values. Because the values are often subsegments of the actual expressions being matched and because they, in turn, may require substitutions within themselves, the substitution list output by M11 is called a compact substitution list.

Figure 6 shows how M11 uses the compact substitution list to effect identity between expressions. The IPL-V symbols in Fig. 6 are taken from the code listed in Sec. XV.

A compact substitution list is not suitable for actual substitution in newly constructed expressions as required by other users of the match process. An expanded form can be obtained by matching with M113 (Fig. 7) which applies M112 (Figs. 8-11) to expand the compact list from M111. The expanded list contains completely substituted locally named copies of the parts and pieces of original expressions that made up the original compact list.

M112 expands the list in one pass by replacing each assigned substitution by its properly substituted locally named copy, using the substitution list itself to look up required substitutions as it goes. The subprocess that constructs properly substituted copies for replacement works recursively and is called delineation (9-100 and 9-200 of M112, Figs. 9, 10). 9-300 of M112 (Fig. 11) is used to replace free variables in expressions.
Begin M113 [Match segments (0), (1) with substitution as required. Output expanded substitution list]

Match segments (M111)
  if matched
  if failed Quit

Expand substitution list (M112, Fig. 8)

Output list, quit +

Fig. 7--Match with Substitution and Output Expanded Substitution List

Begin M112 [Expand substitution list (0)]

Set context (J51)

Locate next expression (J60,J60)
  if located
  if none Quit, clear context (J31)

Delineate expression (9-100, Fig. 9)

Replace with delineated expression (2IW0)

Fig. 8--Expand Substitution List
Begin 9-100 of M112 [Delineate expression (0)]

Set context (J50)

Test if expression is a simple variable (P8)
  if not
    if yes
      Test if a free variable (P9)
        if yes
          if not → Output input, quit (J30)
        Delineate free variable (9-200, Fig. 10)
      Output result, quit (J30)
  9-101: Create local copy (J74, J136)

Generate free variable locations of expression (P28)

[Subprocess 9-300]

Replace the free variable by its delineated substitutor (Fig. 11)

Output delineated replacement expression

Quit, clear context (J30)

Fig. 9--Delineate Expression
Begin 9-200 of M112 [Delineate free variable (0)]

Set context (J50)

Find corresponding substitutor (J10)
  if found —— Output input, quit (J30)
  if none

Delineate substitutor (9-100, Fig. 9)

Quit, clear context (J30)

Fig. 10--Delineate Free Variable

Begin 9-300 of M112 [Replace free variable 2H0 by its delineated substitutor]

Set context (J50)

Delineate free variable (9-200, Fig. 10)

Replace with delineated expression (21W0)

Clear context, quit + for generator (J4)

Fig. 11--Replace Free Variable by Delineated Substitutor
VI. THE OTHER METHODS

The methods, other than M12, do not detect proofs directly. It is the purpose of the other methods to develop new subproblems, which, if they can be proved, will imply proof of the given problem.

In the detachment method (M11, Figs. 12, 13) this is done by matching the whole problem to the right sides of a sequence of theorems whose main connectives are IMPLIES. For each successful match, a new subproblem is constructed by copying the left side of the theorem and substituting into the copy from the expanded substitution list obtained from the successful match.

Problem $\neg PVP$ and axiom $\{AVB}\{BVA\}$ yield subproblem PV-P.

Replacement method M13 works the same way except that it can match expressions to either side of definitions (expressions with main connective $\Rightarrow$).

Problem $\neg PIP$ and definition $\{AIB\}_{\Rightarrow}\neg AVB$ yield subproblem $\neg PVP$.

Problem $\neg PVP$ and definition $\{AIB\}_{\Rightarrow}\neg AVB$ yield subproblem PIP.

The sublevel replacement method (M16, Figs. 14-16) matches problem subsegments to definitions, proceeding through a problem expression one level at a time. At each level, each subsegment is tried for replacement. A new subproblem is formed if one or more subsegments are replaced at a given level.
Begin M11 [Detachment method for problem (0)]

Define context (J45)

Find main expression of problem (J81)
  if found → if none → Quit -, clear context- (J35)

Find maps of both sides of main connective IMPLIES (J10)
  if found → if none → Quit -, clear context (J35)

Find map of right sides of IMPLIES (J82)
  if found → if none → Quit -, clear context (J35)

Create list of theorems with feasible right side matches for whole problem (M63)

Generate feasible theorems for process 9-100 (J100)

---[Subprocess 9-100]---

Match problem to theorem right.
Set H5 for generator (Fig. 13)

Erase feasibles list (J71)

Reverse H5 from generator (J5)
[H5+ means output (0) is a solution,
  H5- means no output]

Clear context (J35)

Fig. 12--Detachment Method
Begin 9-100 of M11 [Try detachment with theorem (0)]

Make free variables of problem disjoint with theorem (M110)

Find right side of theorem (P14)
  if found if none → Quit + for generator (J4)

Match problem to theorem right. (M113)
  if matched if not → Quit + for generator (J4)

Find theorem left side (P13)
  if found if none → Erase substitution list (J72)
  → Quit + for generator (J4)

Create new total expression from theorem left (P17,P24)

Apply substitution list to new expression (M115)

Erase substitution list (J72)

Finish building new subproblem and test utility (M19)
  if O.K. if N.G. → Quit + for generator

Try substitution. (M12)
  if proof if no proof → Quit + for generator

Output proving subproblem and quit - for generator

Fig. 13--Detachment Method Subprocess
Begin M16 [Sublevel replacement method for problem (0)]

Define context (J48)

Find definition maps list (J10)
  if found
  if none

Find map of left sides (J81)
  if found
  if none

Find map of right sides (J82)
  if found
  if none

Find beginning level for replacement (Q17)
  if found
  if none

Copy problem for replacement (P25)

Set "NEW SUBPROBLEM FLAG" to NO (60W5)

Generate segment locations from level 1W3 of problem copy (P26)

[Subprocess 9-100]

Try to replace the segment in location (0).
Set "NEW SUBPROBLEM FLAG" to J4 (ON) if successful.
Set H5+ for generator (Fig. 15)

Execute "NEW SUBPROBLEM FLAG" (01W5)
  if no
  if yes

  Finish up new subproblem 1W4 (M19)
    if good
    if no good

  Try substitution (M12)
    if proof
    if no proof

  Clear context (J38), quit +

Bump level by -1 and test
  if greater than 1 (9-20)
  if no

Assign new level to problem copy (J11)

Clear context (J38), quit -

Bump level by -1 and test
  if greater than 1 (9-20)
  if no

Erase copy (J72)

Fig. 14--Sublevel Replacement Method I
Begin subprocess 9-100 of M16
   [Try to replace the segment in location (0)]

Create a list of definitions with feasible left side matches to segment (M62)

Generate feasible definitions (J100)

[Subprocess 9-200]

Try to replace segment by matching left sides. Set "NEW SUBPROBLEM FLAG" to J4 (ON) and set H5- for generator if successful (Fig. 16)

if failed
   Erase feasibles (J71)
   Quit + for generator (J4)

Create a list of definitions with feasible right side matches to segment (M62)

Generate feasible definitions (J100)

[Subprocess 9-300]

Try to replace segment by matching right sides. Set "NEW SUBPROBLEM FLAG" to J4 (ON) and set H5- for generator if successful (Fig. 16)

Erase feasibles (J71), quit + for generator (J4)

Fig. 15--Try to Replace Located Segment
Fig. 16--Try to Replace Segment by Matching Left (Right) Sides
Problem \([\text{PI-P}]\) and definition \([\text{AIB}] = [-\text{AVB}]\) yield subproblem \([-\text{PV-P}]\).

Sublevel replacement method M17 is identical to M16 except that it tries to replace all subsegments starting with those at the lowest level of the problem expression.

The forward chaining method matches the left side of the problem with left sides of a sequence of appropriate theorems. In the event of a match, a new problem is constructed with copies of the right side of the problem and the right side of the theorem. The new subproblem is then substituted into according to the substitution list obtained from M13.

Problem \(\text{PV}---\) and theorem \(\text{AV-A}\) yield new subproblem \(\text{PI}---\).

Backward chaining is the same except that it matches right sides and uses the left sides to construct new subproblems.

Problem \(\text{PI}[\text{PVQ}]\) and theorem \([\text{AVB}]\) yield new subproblem \(\text{PI}[\text{QVP}]\).

Each new subproblem produced by a method needs further processing after appropriate substitution for free variables. Since this is the same for all methods, a separate routine, M19 (Fig. 17), was designed to finish off new subproblems.

M19 connects a new problem into the subproblem tree by assigning values to its derivation attributes. Then, after testing utility, it either quits with H5+ or erases the bad problem and quits with H5-. The utility measures used are discussed in the next section.
Begin M19 [Finish building new subproblem and measure utility]

Define context (J53)

Fill out description (J11's)

Test utility. Add to found list if can (M43)
  if no good    if O.K.
    Add to untried list (M51)
    Clear context (J33)
    Output subproblem and quit + (J4)

Test if printing rejects (J2)
  if no
    Print reject (M81)
  if yes

Erase no good subproblem (J72)

Clear context (J33), quit - (J3)

Fig. 17--Finish Building New Subproblem
The last thing each method does to each "good" subproblem is to try substitution in the hope that a proof is at hand.
VII. UTILITY MEASURES

LT considers a problem to have sufficient utility if it has not been previously found and is not clearly unprovable. The newness test adds a new problem to the found problems list if it doesn't match any problems already on the list. The non-provability test rejects single variables (P, -P), problems with matching sub-segments of main connective OR (PVP, [PIQ]V[PIQ]), and problems with faulty structures. Routine M43 (Fig. 18) handles utility measurements.

The test for match across OR uses M114 which uses substitution. The test for newness, M42 (Fig. 19), uses M40 (Fig. 20) which does not match with substitution but does consider any pair of free variables in corresponding positions to be matched.

In order to reduce search time in M42, the found problems list is kept in a structured form. The structure is based on the number of levels (Q2), number of distinct variables (Q3), and the number of variable places (Q4) in the problems stored. Only problems with identical values for Q2, Q3, and Q4 are put through a full match test (M40).

Other more interesting and profitable organization techniques might be developed as student exercises. For example, the found problems might be kept on a map similar to that used for true expressions. This might lead to new ways to organize the untried problems and facilitate new and interesting procedures for the single and multiple problem executives.
Begin M43 [Measure utility of problem (0)]

Define context (J43)

Find main expression (J81)
  if found
  if none

Go thru NOTS (P4)
  if O.K.
  if faulty expression

Test if simple variable (P8)
  if no
  if yes

Clear context (J33), quit -

Test if main connective is "OR" (J2)
  if no
  if yes

Test if right and left sides match (M114)
  if no
  if yes

Clear context (J33), quit -

Add to found list if can (M42)
  if cannot
  if can

Quit -

Quit +

No output, clear context (J33)

Fig. 18--Measure Utility
Begin M42 [Add to found list if can]

Define context (J42)

Find appropriate sub-sub-sub-list (9-100, 9-100, 9-100)
if found
if failed Quit - , clear context (J32)

Generate sub-sub-sub-list for matching subprocess (J100)

[Subprocess]

Compare total expressions (M40), reverse H5 (J5)

if no match
if any match
Quit - , clear context (J32)

Add at end of sub-sub-sub-list (J65)

Quit + , clear context (J32)

Begin 9-100 of M42 [Get sub-list]

Locate sub-list of appropriate level or locate place to insert a new sub-list (P55)
if to insert
if usable sub-list
Get sub-list and quit (J80)

Create and insert new sub-list (J91, J64)

Create and insert new data term marker (J120, J64)

Output created sublist and quit

Fig. 19--Add to Found List
Begin M40  [Test match of total expressions without substitution]

Define context (J51)

Find main segment of first total expression (J81)
if found  if none
  Find main segment of other total expression (J81)
if found
  Match main segments (M41, Fig. 21)

Discard LH0 (3OH0)

Clear context (J31), quit + or -

Fig. 20--Test Match of Total Expressions Without Substitution
Begin M41 [Test match of segments without substitutions]

Define context (J31)

Test if first is variable (P8)
  if no —— if yes —— Test if first and second are identical (J3)
  if no —— if yes —— Clear context (J31), quit +
  if yes —— Test if second is free variable (P9)
  if yes —— if no —— Clear context (J31), quit -
  if yes —— Test if first is free variable (P9)
  if yes —— Clear context (J31), quit + or -

Test if second is variable (P8)
  if no —— if yes —— Clear context (J31), quit - (J5)

Test if subsegment connectives are identical (J2)
  if yes —— if no —— Clear context (J31), quit -

Find left subsegment of first (J81)
  if none —— Find left subsegment of second (J81)
  if none —— Discard 1H0 (3OH0)

Discard 1H0 (3OH0)

Reverse H5 (J5), Clear context (J31), quit + or -

Find left subsegment of second (J81)
  if none —— Discard 1H0 (3OH0), clear context (J31), quit -

Match subsegments recursively (M41)
  if matched —— if no match —— Clear context (J31), quit -

Find right subsegment of first (J82)
  if none —— Find right subsegment of second (J82)
  if none —— Discard 1H0 (3OH0)

Discard 1H0 (3OH0)

Reverse H5 (J5), clear context (J31), quit + or - (J5)

Find right subsegment of second (J82)
  if none —— Discard 1H0 (3OH0), clear context (J31), quit -

Match subsegments recursively (M41)

Clear context (J31), quit + or -

Fig. 21--Test Match of Segments Without Substitution
VIII. INFORMATION STORAGE AND RETRIEVAL

The original version of LT similarity-tested all available axioms and theorems for matching with each problem expression. The similarity test used the number of levels, number of distinct variables, and number of variable places in expressions to measure match feasibility. The purpose of the test was to obtain efficiency by screening out unlikely match candidates.

These measures of match feasibility were not very effective because of the extensive processing required and because of the global nature of the measurements.

Selection of true expressions for matching should take advantage of the fact that successful matching depends on similarity of local structural characteristics. Of course, any cheap elimination of useless true expressions is good, but difficulties arise when relatively scarce useful theorems are eliminated. Use of the original similarity test sometimes made problems unprovable because crucial matches were prevented, thus preventing development of crucial subproblems.

These and other considerations have led to development of better measures of match feasibility. Work involved with extending LT† to handle problems involving AND and EQUIVALENCE connectives led to the ideas that an expression is better characterized by its form than by simple counts of its various elements, and that the essence of a form is in its connective structure.

†Development of the true expressions map and extension of LT to handle AND and EQUIVALENCE connectives was done by the author as a term project under Fred Tonge, using the Western Data Processing Center's IBM 7090.
New requirements for match feasibility were developed as follows:

A true expression is considered to be a feasible match if it has the same connective structure as the given problem, viewing the problem as contracted at subsegment places as required.

By "viewing as contracted," we mean viewing a segment as though it were a simple bound variable.† This corresponds to the M111 matching procedure of assigning segments as substitutors for free variables as required. For example, under the new feasibility requirement:

\[[PVQ]I[PV[PVQ]]\]

is a feasible match for Axioms

*1.2 \[[AVA]I\]
*1.3 \[[A][BVA]\]
*1.4 \[[AVB]I[BVA]\];

but not for Axioms

*1.5 \[[AV[BVC]]I[BV[AVC]]\]
*1.6 \[[A][B][CVA]I[CVB]\].

*1.5 fails because its left side has a segment place corresponding to a bound variable place. *1.6 fails because its subsegments have the wrong connectives.

Implementation of the new requirement was accomplished by mapping all true expressions onto one structure, called "The Map of All True Expressions" (L4). This map serves

†In LT, bound variables are treated as though they are segments of a particular but unknown form; thus, they can be substituted for free variables but nothing can be substituted for them.
as an index to the true expressions and facilitates selection of feasible matches without individual testing of each true expression.

The structure of the map is similar to that of expressions in that it forms a tree and has nodes corresponding to variable and segment places. Figure 22 shows the map structure when it contains Axioms *1.2 through *1.6. Main map L4 is an IPL-V description list with connectives as attributes and sublists of submaps as values. Each submap has the same form as the main map and each submap place corresponds to a variable or segment place in at least one true expression. The head of each submap holds a list of theorems, each of which has a variable place corresponding to the submap place. Figure 23 shows how the axiom map looks when keypunched.

By examining the map (Figs. 22, 23) it can be seen that Axiom *1.4 has a variable at the left of OR, which is at the left of main connective IMPLIES, etc. Main map L4 in Fig. 23 has only one attribute because all axioms have the same main connective.

Addition of definitions and newly proved theorems will cause additional attributes to appear in the main map. The head of L4 will always remain empty because no true expression can be without at least one connective. The map is constructed and searched by the routines discussed below.

M54 (Fig. 24) adds expressions to the map structure by mapping each new expression over those previously mapped, extending the structure as needed. Since the structure is a tree, M54 does its work with a recursive subprocess (9-100) which adds subsegments to submaps.
Fig. 22--Axiom Map
TRUE EXPRESSIONS MAP HOLDING AXIOMS

MAPS LIST FOR MAIN CONNECTIVE I.

SUBMAP, LEFT SIDES OF I.

SUBMAP, RIGHT SIDES OF I.

AXIOM, VARIABLE ON LEFT OF I.

MAPS LIST FOR I ON LEFT OF I.

MAPS LIST FOR V ON LEFT OF I.

AXIOM, VARIABLE ON RIGHT OF I.

MAPS LIST FOR I ON RIGHT OF I.

MAPS LIST FOR V ON RIGHT OF I.

SUBMAP, LEFT OF I ON LEFT OF I.

SUBMAP, RIGHT OF I ON LEFT OF I.

SUBMAP, LEFT OF V ON LEFT OF I.

SUBMAP, RIGHT OF V ON LEFT OF I.

SUBMAP, LEFT OF I ON RIGHT OF I.

SUBMAP, RIGHT OF I ON RIGHT OF I.

SUBMAP, LEFT OF V ON RIGHT OF I.

SUBMAP, RIGHT OF V ON RIGHT OF I.

AXIOM, VAR, LEFT OF I, LEFT OF I.

AXIOM, VAR, RIGHT OF I, LEFT OF I.

Fig. 23—Axiom Map in Keypunched Form
AXIOMS, VAR. LEFT OF V, LEFT OF I. 9-20

MAPS LIST, V LEFT OF V, LEFT OF I. 9-21

AXIOMS, VAR. RIGHT OF V, LEFT OF I. 9-22

MAPS LIST, V LEFT OF V, RIGHT OF I. 9-23

MAPS LIST, V RT. OF I, RT. OF I. 9-24

AXIOMS, VAR. LT. OF V, RT. OF I. 9-25

AXIOMS, VAR. RT. OF V, RT. OF I. 9-26

MAPS LIST, V RT. OF V, RT. OF I. 9-27

MAP, LT. OF V, LT. OF V, LT. OF I. 9-28

MAP, RT. OF V, LT. OF V, LT. OF I. 9-29

MAP, LT. OF V, LT. OF V, RT. OF I. 9-30

MAP, RT. OF V, LT. OF V, RT. OF I. 9-31

MAP, LT. OF V, RT. OF I, RT. OF I. 9-32

MAP, RT. OF V, RT. OF I, RT. OF I. 9-33

MAP, LT. OF V, RT. OF V, RT. OF I. 9-34

MAP, RT. OF V, RT. OF V, RT. OF I. 9-35

AXIOM, VAR. L OF V, L OF V, L OF I. 9-36

AXIOM, VAR. R OF V, L OF V, L OF I. 9-37

AXIOM, VAR. L OF V, L OF V, R OF I. 9-38

AXIOM, VAR. R OF V, L OF V, R OF I. 9-39

AXIOM, VAR. L OF V, R OF I, R OF I. 9-40

AXIOM, VAR. R OF V, R OF I, R OF I. 9-41

AXIOM, VAR. L OF V, R OF V, R OF I. 9-42

AXIOM, VAR. L OF V, R OF V, R OF I. 9-43

Fig. 23--(Continued)
Begin M54 [Add total expression (0) to map of expressions (1)]

Define context (40W0)

Find main segment (J81)
  if found → Quit, clear context (J30)
  if none → Add main segment (1) to map (0) (9-100)

Quit, clear context (J30)

Begin 9-100 of M54 [Add segment (1) to map (0)]

Define context (J43)

Test if simple variable (P8)
  if no → Find submaps list (J10)
  if yes → Add theorem name to list in map head

  → Quit, clear context (J33)

Find submaps list (J10)
  if found → Extend map
  if none →

Add 1st subsegment to 1st submap (9-100)

Test if another subsegment (P6)
  if yes → Add 2nd subsegment to 2nd submap (9-100)
  if no → Quit, clear context (J33)

Quit, clear context (J33)

Fig. 24--Add Expression to Map
M63 (Fig. 25) extracts names of feasible match expressions from the map by using a given problem expression as a guide. This is done by laying the problem expression over the map structure and collecting true expression names from overlaid map heads. Figure 26 shows a problem expression \([PVQ]I[FV[PVQ]]\), laid over the map so that Axioms \(*1.2\), \(*1.3\), \(*1.4\), and \(*1.5\) appear in overlaid map heads. Note that \(*1.5\) is not a feasible match for the given problem and that \(*1.6\) is never encountered.

M63 uses recursive processes M62 (Fig. 25) to extract a list of feasible match expressions from the map structure. It uses the following procedure to gather expression names from overlaid map heads and prevent unwanted expressions, such as \(*1.5\) of Fig. 26, from appearing in the output list.

For each map (submap) and corresponding segment (subsegment):

1) Recursively find all feasible matches for all subsegments from corresponding submaps; then

2) If the segment (subsegment) connective is unary (such as NOT), skip to step 3) below, otherwise find the intersection of matches obtained from all submaps; then

3) Append the results of step 2) to a copy of the list of expressions from the head of this map (submap); and

4) Output the result as a list of feasibles from this map (submap).

This procedure meets all requirements set forth previously. When applied to Fig. 26, it shows that Axiom \(*1.5\)
Begin M63 [Create list of true expressions for feasible match with total expression (1) from map (0)]

Find main expression of TEX (J81)
if found         if none            Create empty list (J90) for output, quit

Begin M62 [Create list of true expressions for feasible match with segment (0) from map (1)]

Copy list of theorems in head of map (J73)

Test if segment is a simple variable (P8)
if no            if yes

Locate first subsegment (J60)
if located       if none

Find list of appropriate submaps (J10)
if found         if none

Locate first submap (J60)
if located       if none

Create list of feasibles from first submap with first subsegment (M62)

Test if more subsegments (P6)
if yes            if no

Append first submap list to copy of map head list for output (J16)

Save first submap list as working list

Quit

Locate next subsegment (J60)
if located       if none

Append working list to copy of map head list for output (J63)

Quit

Locate next submap (J60)
if located       if none

Create list of feasibles from new submap with new subsegment (M62)

Get intersection of new submap list and working list (9-100)

Test if working list is not empty (J78)
if not empty      if empty

Erase empty work list (J72)

Quit, output copy of list from map head

Fig. 25--Create List of Feasible Expressions
L4: Axiom Map

Problem

\[ [P\nu Q]_1 [P\nu [P\nu Q]] \]

Fig. 26--Axiom Map with Overlaid Problem
is eliminated by step 2) because its name doesn't appear in the map head on the right of OR on the left of main connective IMPLIES. Step 3) provides viewing as contracted at every segment and subsegment place. The procedure completely avoids consideration of #1.6 because #1.6 doesn't have the right subsegment connectives.

The ability to completely avoid consideration of certain types of unlikely match candidates is one of the most important features of this retrieval system because each method (M1-M16) deals with a different part of the true expressions it uses. M12 matches whole expressions, so it uses the whole map. M11 matches whole problem expressions to the right sides of true expressions that have main connective IMPLIES, so it uses only the submap of right sides of main connective IMPLIES. M13, M16, and M17 use only the submaps of DEFINITIONAL EQUIVALENCE. M14 and M15 use whatever section of the map is appropriate for the problem at hand.

M14, M15, M16, and M17 use problem subsegments to search appropriate submaps, thus taking full advantage of the ability to ignore all irrelevant parts of the map. This ability becomes more valuable as the number of true expressions becomes large. Thus, this new theorem storage and retrieval system at least partially solves the old problem of what to do with too much information. Obviously irrelevant true expressions no longer get in the way.

Although the expression map technique can be used elsewhere, it is used in this version of LT only to organize true expressions. In particular, it can be applied to the found problems list which is currently
organized in terms of number of levels, number of variables, and number of variable places. However, the payoff will be smaller because the found problems match process (M40) doesn't use substitution, so there is no need for viewing as contracted at any level. In fact, a map search routine for a found problems map should specifically avoid viewing as contracted.

Reorganization of the untried problems with a map structure would be more interesting. Some modification of the structure would be required because the viewing as contracted problem is inverted. Once this is done, routines could be devised to search both the true expressions map and the untried problems map at the same time. The result of such a development might enable LT to "see" a larger part of its problem at a time by giving it some ability to scan the problem and its context as a unit.

This sort of thing should lead to more sophisticated ways to select and apply methods to problem theorem pairs. The problem executives (M1, M2) would need expansion to explore problems and plan attacks. The lower-level routines would not need any significant modification.
IX. LOWER-LEVEL ROUTINES

The lower-level routines operate on expression structures and terms. The objective of segregating these processes from the higher-level routines is to generalize the program. In theory, the expression structures that LT handles can be modified by changing only the lower-level routines. In practice, a few changes would also be required in some of the higher-level routines.

Some lower-level routines have interesting characteristics. First among these are the "Q" routines. Each of the IPL-V symbols, Q1-Q19, can take on more than one meaning, depending on the context of its usage. For example, Q7 is the name of a routine (FIND EXTERNAL NAME) and is also the name of an attribute (EXTERNAL NAME) used on the description lists of total expressions, variable terms, and character symbols.

<table>
<thead>
<tr>
<th>COMMENTS</th>
<th>TYPE</th>
<th>NAME</th>
<th>PQ</th>
<th>SYMB</th>
<th>LINK</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROUTINE HEADER</td>
<td>5</td>
<td></td>
<td>00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FIND EXTERNAL NAME</td>
<td></td>
<td>Q7</td>
<td>10</td>
<td>Q7</td>
<td>J10</td>
</tr>
<tr>
<td>DATA HEADER</td>
<td>5</td>
<td></td>
<td>01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SYMBOL FOR CHARACTER X</td>
<td></td>
<td>X0</td>
<td>9-1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td></td>
<td>9-1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATTRIBUTE-EXTERNAL NAME</td>
<td></td>
<td>Q7</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>9-2</td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>VALUE-ALPHANUMERIC DATA TERM</td>
<td>9-2</td>
<td>21</td>
<td></td>
<td>X</td>
<td></td>
</tr>
</tbody>
</table>

There is no conflict of usage because the description list processes never operate on the contents of cells whose names are used as attributes. If, by private convention, nothing is ever done with the value of an attribute, the attribute symbol may also serve as its own dummy value,
as well as for the name of the routine used to find it. Q5, Q6, and Q7 are examples of this usage. (See Sec. XV for code listings.)

There is no reason for "find" routines to be short and simple. Q2, Q3, and Q4 are examples of what is sometimes called an active attribute. Routine Q2 (Figs. 27, 28), when applied to a total expression, tries to find the number of levels in the expression as the value of attribute Q2 on the description list. If it doesn't find a value there, the routine counts levels, assigns the count as the value of attribute Q2, and outputs the value. H5 is set minus if a value cannot be found. This can only happen if the expression structure is faulty so that the number of levels is undefined. Q3 (Fig. 29) and Q4 work in a similar fashion to find the number of distinct variables and number of variable places.

Some interesting generators can be found among the "P" routines. P29 (Fig. 30) is interesting because it uses itself recursively with J18 as a subprocess to search the lower levels of expressions. P29 generates bound variable locations from expressions. P28 is of the same form, but generates free variable locations.

P26 (Figs. 31, 32) has a more difficult task. It is used to generate subsegment locations from a given level of a given expression. To do this, it uses local subgenerator 9-200 (Fig. 32), which counts expression sublevels as it descends into the expression to generate subsegment locations. The most interesting feature of P26 is the way subgenerator 9-200 is executed by subprocesses 9-100, with 9-100 as the subprocess.
Begin Q2 [Find number of levels of total expression (0)]

Preserve name of TEX (40H0)

Find number of levels on description list (J10),
  if failed → if found → Discard TEX (J8), quit +

Create local data term for description list (J120)

Create counter (J120)

Define context and save counter (J52)

Find main expression (J81)
  if none → if found →
    Count levels of main segment (9-100, Fig. 28)

Erase counter (J9)

Set up level for output (11W1)
  if H5+ → if H5-
    Erase level (J9), clear context (J32), quit -

Assign level as value of Q2 on description list (J11)

Clear context (J32), quit +

Fig. 27--Find Number of Levels
Begin 9-100 of Q2 [Count sublevels recursively]

Copy counter (J120)

Tally new counter (J125)

Define context and save new counter (J50)

Test if segment is variable (P8)
  if yes
  if no
    Generate subsegments (J100) for counting by (9-100)
    Erase counter (J9), clear context (30W0), quit
  Test if counter greater than level (J115)
    if no
    if yes
      Set level equal to counter (J121)
    Erase counter (J9), clear context (30W0), quit + (J4)

Fig. 28--Count Sublevels Recursively
Begin Q3 [Find number of distinct variables in total expression (0)]

Preserve name of TEX (40H0)

Find number of distinct variables on description list (J10)
  if none
  if found → Discard TEX (J8), quit +

Define context (J42)

Create list of free variables (P30)

Count free list (J126)

Mark count local for description list (J136)

Erase free list (J71)

Create bound list (P31)

Count bound list (J126)

Add bound count to free count (J110)

Erase bound list (J71)

Erase bound count (J9)

Assign counter as value of Q3 on description list (J11)

Clear context (J32), output counter and quit +
Begin P29  [Generate locations of bound variables]

Define context  (P17)

Test if input segment is a variable  (P8)
  if no  if yes  Clear context, quit +  (J19)

Locate next subsegment  (J60)
  if located  if none  Clear context, quit +  (J19)

Test if a free variable location  (P9)
  if yes  if no

Test if a bound variable location  (P8)
  if no  if yes

Recursively generate  (P29) from subsegment for subprocess J18
  if H5+  if H5-

Apply subprocess  (J18)
  if H5-  if H5+

Clear context, quit -  (J19)

Fig. 30--Generate Locations of Bound Variables
Begin P26 [Generate segment locations from level (2)
of total expression (1) for process (0)]

Define context (J17)

Find main expression (J81)
  if found
    if none ➔ Clear context (J19), quit +

Test if variable (P8)

Create zero valued counter (J120)

Save counter and level (J21)

Generate subsegment locations from main segment (9-200, Fig. 32)

---[Subprocess 9-100]---

Test if at proper level (J114)
  if no
    if yes ➔ Fire subprocess of P26 (J18)
      Quit + or - for generator

Generate subsegment locations for subprocess 9-100 (9-200)
  Quit + or - for generator

Erase counter (J9)

Clear context (J19) and quit + or -

Fig. 31--Generate Segment Locations from
  Given Level of Expression
Begin 9-200 of P26 [Generate subsegment locations from segment (1) for subprocess (0)]

Define context (J17)

Create new sublevel counter (J120)

Bump counter (J125)

Locate next segment place (J60)
if located
Test if variable (P8)
if yes
Restore H0 (40H0)

if no
Fire subprocess (J18)
if H5+

if H5-
Erase counter (J9)

Clear context (J19), quit + or -

Fig. 32--Generate Subsegment Locations from Segment
P50, P51, and P52 also form an interesting group of routines. They are used to convert expressions from list form to tree form. Previous discussions have dealt only with tree form expressions. LT uses list form expressions to make inputting of expressions easier.

Expressions can be put into LT in any of three different forms. Tree forms and list forms can be directly loaded or M89 may be used to read Hollerith records with the line-read primitives (J180's). M89 will be discussed in the next section.

List form expressions have no tree structure so they are easier to write down on code sheets. The expression is carried as a simple list of character symbols, as shown below. Entire expressions must be parenthesized and redundant parentheses are not allowed.

\[
\begin{array}{c}
\text{Head} \\
\{
\begin{array}{c}
\times 208 \\
(0 \\
P0 \\
I0 \\
P0 \\
)0 \\
0 \\
Q7 \\
9-2 \\
0
\end{array}
\end{array}
\]

P50 converts expressions from list form to tree form only if they are not yet in tree form. Tree form expressions have attribute Q15, while list forms do not.

P50 is a simple routine which first tests for tree form. If the given expression is not in tree form, P50 replaces delimited external connectives (P51); creates a
tree form from the list form (P52); discards the old list form, saving its head and description list (J75,J71); inserts the new tree form main expression after the old head (J64); and assigns tree form attribute, Q15, to the expression (J11).

P51 replaces delimited connectives such as .=. by scanning the list form for the delimiter symbol. When one is located, a check is made to be sure there is another delimiter symbol on the other side of the delimited symbol. If there is, all three symbols are replaced by the proper internal symbol obtained from the table of delimitable symbols. K7 holds the delimiter symbol and L8 is the name of the table of delimitable symbols.

P52 (Figs. 33, 34) creates a tree form main segment, complete with subsegments, from a given list expression without destroying the latter. Since a parenthesized list

---

Begin P52 [Create main segment from list (0)]

Define context (J41)

Create first segment from list 1WL (9-100, Fig. 34)

Clear context and quit + or - (J31)

---

Fig. 33--Create Main Segment from List Expression
Begin 9-100 of P52  [Create next segment from list expression 1W1]

Locate first symbol of list expression 1W1 (J60)
  if located
  if none

Test if opening paren (J2)
  if yes
  if no

  Test if NOT (J2)
  if yes
  if no

  Test if variable (P8)
  if yes
  if no

Set up empty segment (J90, J50)

Create NOTed subsegment (9-100)
  if bad
  if good

Insert subsegment into segment 1WO (J65)

Mark segment 1WO local (J136)

Output segment 1WO, clear setup (J30), and quit +

Set up empty segment (J90, J50)

Create first subsegment (9-100)
  if good
  if bad

Insert first subsegment into segment 1WO (J65)

Locate next symbol of list expression 1W1 (J60)

Test if connective (P7)
  if yes
  if no

  Insert dummy (/14) character (J63)

Create second subsegment (9-100)
  if good
  if bad

Insert second subsegment into segment 1WO (J65)

Locate next symbol of list expression 1W1 (J60)

Test if closing paren (J2)
  if yes
  if no

  Insert dummy (/14) character (J63)

Output good created segment 1WO

Clear setup (J30) and quit +

Clear setup (J30) and quit -

Fig. 34--Create Next Segment from List Expression
expression represents a tree structure, P52 uses a recursive subprocess (9-100) to translate the list expression into its corresponding main expression.

If the list expression is faulty, P52 inserts a dummy character symbol (/14) into the list expression. When the bad list expression is printed by M88, the dummy character appears at the place where trouble was detected.

The remaining lower-level "P" and "Q" routines are relatively trivial and can be studied directly from the code.
X. INPUT-OUTPUT ROUTINES

The input-output routines deal directly with print line primitives (J150's) and read line primitives (J180's).

As mentioned in the preceding section, expressions may be put into LT by loading them with J165 or by reading them with M89. J165 can load tree form expressions or list form expressions, while M89 can read Hollerith records with the following format:

Beginning in column 1: Any number of blanks (including none); followed by a regional symbol expression name; followed by at least 1 blank; followed by an expression string (enclosed in parentheses with no imbedded blanks); followed by at least 1 blank; followed by an optimal suffix (up to 5 characters); followed by at least 1 blank.

Example: */1.01 ((AIB)=.-AVB) DEF.

M89 (Fig. 35) reads one record from unit 1W18 and then scans it in read line buffer 1W24, using the read line primitives. If the record is an End-of-File (E-O-F) or is totally blank, M89 quits with no output and sets H5-. If the record is faulty in some obvious respects, it is skipped and the next record from read unit 1W18 is tried. This procedure continues until an acceptable expression is found or an empty record (or an E-O-F) is found.

In the first part of the scan procedure, the name of the expression is located and checked to see if it is a regional symbol other than a standard character symbol. This is tested with J130 and P18. If the name is okay,
the main expression string is located and used to build a list form expression of variables, connectives, and parentheses. If a suffix can be found, it is assigned as the data term value of Q18 on the list expressions' description list.

M88 is the only other input-output routine that deals with list form expressions. It is used to print expressions that fail in the conversion process.

Routines M70-M82 are used to print various tree form expressions on unit LW19. (See the vocabulary, Sec. XIV, for a list of these routines.) M73 is the central routine in this group because it is used by other routines to enter tree form expressions in the print line buffer. M73 works recursively to build a character string in the print line from a tree form expression. The process is similar to, but opposite from, that of P52. If M73 gets in trouble, it enters the external name of /14 to indicate where the expression is faulty and then continues to enter the rest of the given expression.

M71, which prints whole proof sequences, uses the derivation information associated with each problem (attributes Q10 through Q14) to trace out the successful sequence.

Two other output routines are worth mentioning. M76 is a simple routine which enters data term "text lists" by generating from them for subprocess J157.

M79 is more sophisticated. It can be used to enter the name of anything. First it tries to enter the external name (Q7); if it cannot, it tries to enter the problem number (Q8); if neither can be found, it enters
Begin M89 [Read next logic expression]

Define context (preserve W0, W25, W30)

Clear buffer 1W24 (J154)

Fill buffer from 1W18 (J180)
  if O.K.  if end-of-file Clear context quit - (9-0)

Create column counter (J120)

Locate first character of expression name (J184)
  if O.K.  if none Clear context quit - (9-1)

Set extent data term (J183)
  if N.G.  if O.K.

Reset for next (9-2)

Find name of expression (J181)

Test if name is O.K. (J130, P18)
  if N.G.  if O.K.

Reset for next (9-4)

Set data term to external name (J182)

Assign external name as value of Q7 (J11)

Locate first of expression (J184)
  if none  if located

Reset for next (9-5)

Fig. 35--Read Next Logic Expression
Find next character symbol (J186)
  if O.K.  if none
  
  Replace character symbol if necessary (P19)
  
  Add at end of list form expression (J65)

Locate first of suffix, if any (J184)
  if none  if located
  
  Determine extent (J183)
    if none  if O.K.
    
    Create prototype data term (J120)
    
    Set prototype to suffix (J182)
    
    Assign suffix as value of Q18 (J11)

Adjust for extra symbol in H0

Set H5+ (J4)

Erase column data term (J9)

Erase extent data term (J9)

Restore W25, W30, and W0 (J30)

Quit with H5+ or -

Fig. 35--(Continued)
the given symbol with J156 which enters it in regional or internal form.

The remaining input-output routines are fairly simple examples of how to use the print line primitives. The student should be able to decipher them on his own.
XI. RUN EXECUTIVES AND DEBUGGING Routines

There are two run executives. The first is X9 which creates a restart tape holding the main body of LT's routines and data. X9 terminates with J165 to facilitate loading of additional data and/or routines at run time.

The second run executive is X1. It is the highest-level executive for a theorem-proving run. After initializing some debugging devices, which will be discussed later, X1 reads a set of true expressions for the run converting each expression to tree form and adding it to the map of true expressions. The set of true expressions immediately follows the start card for X1 and ends with the first totally blank card. Any expression that fails in its conversion step (P50) is omitted from the expression set and is printed with M88 to report the difficulty.

Next, X1 reads a set of problem expressions. The problems are converted to tree form and added to the list of unproved expressions (L3). Upon encountering the end of the problem set, X1 links to multiple problem executive M2 which feeds the problems, one at a time, to the single problem executive M1 (Fig. 1).

Executive M2 tries for proofs under two different conditions. If K30 holds "RO," proved theorems are to be added to the set of true expressions, thus retaining results of past efforts for future application. If K30 does not hold "RO", proved theorems are to be forgotten and each new problem is to be started from scratch.

† See Ref. 12.
This procedure for remembering is not the only possible one. Each subproblem in a successful proof sequence is a true theorem in its own right and could be added to the set of true expressions. Modification of LT to incorporate this other kind of remembering might make a good student project. Many experiments might be designed around this part of LT.

Symbols X10 through X19 are used for debugging routines, and X20 through X29 are used for debugging lists.

When X10 is used as the trap value of attribute H3 on the description list of W26, X10 will be executed when the cycle count in H3 becomes equal to W33. X10 sets IW31 to full trace mode (IW31 = 1) and forces a monitor point with X19 to invoke the change immediately. The trace mode invoked by X10 can be revoked by executing X11 which pops W31 if it has been pushed down and executes X19 to make the change effective immediately.

X10 and X11 are useful when a full trace is desired of a small part of the program that is executed only after a considerable running time. By initially setting W33 to the desired value, the program can run in no-trace mode until W33 equals H3 and then full trace can be invoked for a short time, after which it can again be revoked.

Other similar routines can be designed to meet special needs when they arise. It is difficult to anticipate exactly what will prove useful in all situations.

Routine X13 is used to "snap" (0) at monitor points by loading X13 into W12 and W13. There is some danger in using J150 this way because (0) is not always guaranteed
to be a proper list structure. For example, (0) is sometimes a generator subprocess.

Routine X14 is used to obtain a restart tape on an operator signal from the computer console by loading X14 into W14. X14 uses the fact that J166 sets H5+ and the restart mechanism sets H5- in order to stop with a post mortem if just saved or to go on with the program if restarting.

Routine X15 is used to extend the post mortem by loading X15 into W15. The listing in Sec. XV shows that X15 is a simple routine to print L4 (the map of all true expressions).

Routine X19, as mentioned in connection with X10 and X11, is used to force a monitor point in order to make changes in trace mode effective immediately.

The debugging lists (X21-X23) are used by X1 to mark routines for tracing and to set up the trap attribute-value pairs of W26. The technique of marking routines to trace from a given list is simple but effective. The other way to mark for trace is to reload the whole routine with a trace mark (Q = 3 or 4) in its head.

The above mentioned debugging aids were the only ones used to check out the program. Other more sophisticated tools might be designed, but LT's bugs have not required them.
XII. A SAMPLE RUN

This section includes a listing of an input deck for a run and a listing of the resultant output.

INPUT DECK

The input deck includes a modified routine, a modified data structure, and a collection of run parameters and lists. Of particular interest are L6 and L7 which control the order of application of the methods. This run has relatively low limits (K20-K22) and calls for printing of rejected problems (K31) and remembering proved problems (K30).

Following the KICK OFF FOR PROVING THEOREMS are three sets of logic expressions. The first set is to be used as true expressions and includes definitions and axioms. Definition *2.33 will fail in conversion because LT cannot handle expressions such as (PVQVR). The second set contains the theorems to be proved (problems) in this run. The third set will be ignored because it follows the second blank card. It was placed here in order to include in this Memorandum all other theorems from *2, *3, *4, and *5 of Principia Mathematica.
JOB  8168,LTNEW1,EAS826,5MIN,0,C99,C  STEFFERUD
ASSIGN  A6=SYSAR2
ASSIGN  B6=SYSAR3
IPL
LOGIC THEORIST TEST
RELOAD FROM TAPE 2
MODIFIED ROUTINES

Q17  FIND LEVEL OF SUBSEGMENT
     REPLACEMENT IN TEX (0).

FIND CURRENT LEVEL.
IF NONE,
FIND NUMBER OF LEVELS,
IF NONE, QUIT .
COPY,
SAVE ONE FOR OUTPUT,

AND ASSIGN AS CURRENT LEVEL.
MODIFIED DATA
/16 DUMMY EXPRESSION --
'DEFINITIONS'.

/16 9-1 01
9-2 0 0
Q15 Q15
Q7 Q7

EXTERNAL NAME
CONNECTIVE 'I'.

/16 9-2 21
10
9-10
9-20 0

DUMMY VARIABLE 'DEFI'.

/16 9-10 0
0
Q5
Q5
Q9
Q7

EXTERNAL NAME
DUMMY VARIABLE 'TIONS'.

/16 9-20 21DEFI
0
Q5
Q5
Q9
Q7

EXTERNAL NAME
RUN DATA HEADER

/16 5 01 21TIONS
0

LIMIT ON NUMBER OF SUBPROBLEMS K20 + 1 50
LIMIT ON NUMBER OF SUBSTITUTIONS K21 + 1 50
LIMIT ON EFFORT K22 + 1 2000

R = ADD PROVED THEOREMS TO THEOREMS
Y = PRINT REJECTED SUBPROBLEMS.
L6 LIST OF METHODS FOR ORIG PROBS
L6
M16
M17

L7 LIST OF METHODS FOR PROBLEMS
L7
M13
M11
M14
M15

W12 SET-UP ENTRY SNAP ACTION
W12 X13 0
W13 SET-UP EXIT SNAP ACTION
W13 X13 0
W15 X15 0
W21 X21 0
W22 X22 0
W23 X23 0

DESCRIPTION LIST OF TRAP ACTIONS
W012 X012 0
W013 X013 0
W015 X015 0
X021 X021 0
X022 X022 0
X023 X023 0
KICK OFF FOR PROVING THEOREMS*

*1.01 ((P|Q)\rightarrow (-P|Q)) DEF*
*2.33 ((P|Q|V|R)\rightarrow ((P|Q)V|R)) DEF*
*3.01 ((P\&Q)\rightarrow (-P-Q)) DEF*
*4.01 ((P\&Q)\rightarrow ((P|Q)*(Q|P))) DEF*
*1.02 (B\&I|A|A)
*1.3 (B\&I|A|B)
*1.4 ((A|B|I)*(E|V|A))
*1.5 ((A|B|V|C|I)*(E|B|A|C|V))
*1.6 ((B|I|C|I)*(A|B|I|A|V|C))

*2.01 ((PI|P)I-P)
*2.02 (QI|PI|Q)
*2.03 ((PI-Q)I(QI-P))
*2.04 ((P|Q|I|R)I(QI|P|R))
*2.05 ((QI|P)I((PI|Q)I(P|R)))
*2.06 ((PI|Q)I((QI|R)I(P|R)))
*2.07 (PI|PVP)
*2.08 (PI|P)
*2.10 (-PVP)
*2.11 (PV-P)
*2.12 (PI=-P)
*2.13 (PV-P)
*2.14 (-PIPI)
*2.15 ((-PI|Q)I(-QI|P))
*2.20 (PI|P|Q)
*2.21 (-PI|P|Q)
*2.24 (PI|=-I|Q)
*3.13 ((-PI|Q)I(-P|V|Q))
*3.14 ((-P|V-Q)I(-P|Q))
*3.24 (=-P|Q)
*4.13 (P=-P)
*4.20 (P=P)
*4.24 (P=(P|P))
*4.25 (P=(P|P|V|P))

*3.02 ((PI|Q|R)\rightarrow ((PI|Q)*(Q|I|R))) DEF*
*4.02 ((P=Q=R)\rightarrow ((P|Q)*(Q=|R))) DEF*
*4.34 ((P\&Q*R)\rightarrow ((P|Q)*(Q*R))) DEF*
*2.16 ((PI|C)I(-Q|P))
*2.17 ((-Q|I=P)I(P|I|Q))
*2.18 ((-PI|P)I|P)
*2.25 (PV((P|Q|I|Q))
*2.26 (-PV((P|I|Q|I|Q))
*2.27 (PI((PI|Q)|I|Q))
*2.30 ((PV|Q|V|R)I(PV)(R|V|G))
*2.31 ((PV|Q|V|R)I(PV|QV|R))
*2.32 ((PV|Q|V|R)I(PV|QV|R))
*2.36 ((Q|I|P)I((PV)(R|P)))
*2.37 ((Q|I|P)I((QV)(P)))
*2.37 ((Q|I)I((QV)(R|P)))
*2.38 ((Q|I)I((QV)(P|V|P)))
*2.40  ((PV(PVQ))I(PVQ))
*2.41  ((QV(PVQ))I(PVQ))
*2.42  ((-PV(PIQ))I(PIQ))
*2.43  ((PI(PIQ))I(PIQ))
*2.45  (-PVQ)I-P)
*2.46  (-PVQ)I-Q)
*2.47  (-PVQ)I(-PVQ))
*2.48  (-PVQ)I(PV-Q)
*2.49  (-PVQ)I(-PV-Q)
*2.50  (-PIQ)I(-PIQ))
*2.51  (-PIQ)I(PQ-I)
*2.52  (-PIQ)I(-PI-Q))
*2.51  (-PIQ)I(QIP))
*2.53  ((PVQ)I(-PIQ))
*2.54  ((-PIQ)I(PVQ))
*2.55  ((PIQ)I(PVQ))
*2.56  (-QI(PVQ))IP)
*2.57  ((-PIQ)I((PIQ)IQ))
*2.58  ((PIQ)I((-PIQ)IQ))
*2.59  ((PVQ)I((PIQ)IQ))
*2.60  ((PIQ)I((PVQ)IQ))
*2.61  ((PVQ)I((-PVQ)IQ))
*2.62  ((PVQ)I((PVQ)IQ))

*2.63  ((PVQ)I(-PVQ)IP))
*2.64  ((PVQ)I(PV-Q)IP))
*2.65  ((PIQ)I((PI-Q)I-P))
*2.66  ((PIQ)IQ I(PIQ))
*2.67  ((PIQ)IQ I(PVQ))
*2.68  ((PIQ)IQ I(QIP))
*2.69  ((PIQ)IQ I((QIP)IF))

*2.73  ((PIQ)I((PVQ)I(PVQ)) I(QVR))
*2.74  ((QIP)I((PVQVR)I(PVQ)))
*2.75  ((PVQ)I((PVQ)I(PVQ)))
*2.76  ((PVQ)I((PVQ)I(PVQ)))
*2.77  ((PIQ)I((PIQ)I(PQ)))
*2.78  ((PVQ)I((PVQ)I(PVQ)))
*2.79  ((PVQ)I((PVQ)I(PVQ)))
*2.80  ((QVR)I((QVR)I(QVS))
*2.81  ((GI(RIS))I((PVQ)I((PVQ)I(PVQ))))
*2.82  ((PVQVR)I((PVQVR)I(PVQ)))
*2.83  ((PQ)I((PQ)I(PQ)))
*2.84  ((PQ)I((PQ)I(PQ)))
*2.85  ((PQ)I((PQ)I(PQ)))
*2.86  ((PIQ)I((PIQ)I(PQ))

*3.10  ((P*Q)I(-PV-Q))
*3.11  ((-PV-Q)I(P*Q))
*3.12  ((-PV-Q)I(P*Q))
*3.13  ((-PV-Q)I(P*Q))
*3.14  ((-PV-Q)I(P*Q))
*3.15  ((-PV-Q)I(P*Q))
*3.16  ((-PV-Q)I(P*Q))
*3.17  ((-PV-Q)I(P*Q))
*3.18  ((-PV-Q)I(P*Q))

*3.20  ((PIQ)I(P*Q))
*3.21  ((PIQ)I(P*Q))
*3.22  ((PIQ)I(P*Q))
*3.23  ((PIQ)I(P*Q))
*3.24  ((PIQ)I(P*Q))
*3.25  ((PIQ)I(P*Q))
*3.26  ((PIQ)I(P*Q))
*3.27  ((PIQ)I(P*Q))
*3.28  ((PIQ)I(P*Q))
*3.29  ((PIQ)I(P*Q))
*3.30  ((PIQ)I(P*Q))
*3.31  ((PIQ)I(P*Q))
*3.32  ((PIQ)I(P*Q))
*3.33  ((PIQ)I(P*Q))
*3.34  ((PIQ)I(P*Q))
*3.35  ((PIQ)I(P*Q))
*3.37  ((P*Q)IR I((P*R)I-G))
*3.38  ((P*Q)I(PIQ))
*3.39  ((PIR)I((P*Q)IR))
*3.40  ((QIR)I((P*Q)IR))
*3.41  (((PIQ)*(PIR))I((PI(Q*R))})
*3.42  (((QIR)*RIP))I((QVR)IP})
*3.43  ((PIQ)I((P*R)I I(R*Q)))
*3.44  ((PIR)*(QIS))I((P*Q)I(R*IS)))
*3.45  ((PIR)*(QIS))I((P*Q)I(R*IS)))
*3.46  ((PIR)*(QIS))I((P*Q)I(R*IS)))
*4.10  ((PIQ)=(-Q|P))
*4.11  ((P=Q)=(-P=-Q))
*4.12  ((P=-Q)=(-Q=-P))
*4.13  ((P*Q)|R|=((P*R)I-Q))
*4.14  ((P*Q)=R|=((P*R)I-Q))
*4.15  ((P*Q)=R|=((Q*R)I-P))
*4.16  ((P=Q)=(Q=P))
*4.17  ((P*Q)=(Q=R)))I((P=R))
*4.18  ((P*Q)=(Q=P))
*4.19  (((PVQ)=((QVQ))
*4.20  (((PVQ)=((QVQ))
*4.21  (((PVQ)=((QVQ))
*4.22  (((PVQ)=((QVQ))
*4.23  (((PVQ)=((QVQ))
*4.24  (((PVQ)=((QVQ))
*4.25  (((PVQ)=((QVQ))
*4.26  (((PVQ)=((QVQ))
*4.27  (((PVQ)=((QVQ))
*4.28  (((PVQ)=((QVQ))
*4.29  (((PVQ)=((QVQ))
*4.30  (((PVQ)=((QVQ))
*4.31  (((PVQ)=((QVQ))
*4.32  (((PVQ)=((QVQ))
*4.33  (((PVQ)=((QVQ))
*4.34  (((PVQ)=((QVQ))
*4.35  (((PVQ)=((QVQ))
*4.36  (((PVQ)=((QVQ))
*4.37  (((PVQ)=((QVQ))
*4.38  (((PVQ)=((QVQ))
*4.39  (((PVQ)=((QVQ))
*4.40  (((PVQ)=((QVQ))
*4.41  (((PVQ)=((QVQ))
*4.42  (((PVQ)=((QVQ))
*4.43  (((PVQ)=((QVQ))
*4.44  (((PVQ)=((QVQ))
*4.45  (((PVQ)=((QVQ))
*4.46  (((PVQ)=((QVQ))
*4.47  (((PVQ)=((QVQ))
*4.48  (((PVQ)=((QVQ))
*4.49  (((PVQ)=((QVQ))
*4.50  (((PVQ)=((QVQ))
*4.51  (((PVQ)=((QVQ))
*4.52  (((PVQ)=((QVQ))
*4.53  (((PVQ)=((QVQ))
*4.54  (((PVQ)=((QVQ))
*4.55  (((PVQ)=((QVQ))
*4.56  (((PVQ)=((QVQ))
*4.57  (((PVQ)=((QVQ))
*4.58  (((PVQ)=((QVQ))
*4.59  (((PVQ)=((QVQ))
*4.60  (((PVQ)=((QVQ))
*4.61  (((PVQ)=((QVQ))
*4.62  (((PVQ)=((QVQ))
*4.63  (((PVQ)=((QVQ))
*4.64  (((PVQ)=((QVQ))
*4.65  (((PVQ)=((QVQ))
*4.66  (((PVQ)=((QVQ))
*4.67  (((PVQ)=((QVQ))
*4.68  (((PVQ)=((QVQ))
*4.69  (((PVQ)=((QVQ))
*4.70  (((PVQ)=((QVQ))
*4.71  (((PVQ)=((QVQ))
*4.72  (((PVQ)=((QVQ))
*4.73  (Q1(P=(P*Q)))
*4.74  (-PI(Q=(PVQ)))
*4.76  (**(PIQ)*(PIR))**(P**(Q*R))**
*4.77  (**(QIP)*(RIP))**(**(QVR)**IP)**
*4.78  (**(PIQ)*(PIR))**(PI**(QVR))**
*4.79  (**(QIP)*(VIP))**(Q**(R)*IP)**
*4.80  ((PI-P)=P)
*4.81  (**(-PIP)=P)**
*4.82  (**(PIQ)*(PI-Q))=P**
*4.83  (**(QIP)*(QPI))=Q**
*4.84  ((P=Q)**{{(PIR)}=(QIR)}**)
*4.85  ((P=Q)**{{(QIR)}=(RIR)}**)
*4.86  ((P=Q)**{{(P=R)}=(Q=R)}**)
*4.87  (**(PQ)*IR)**{PI**(QIR)}**(QI**(PIR)**(Q**(P)*IR))**
*5.30  (**(PQ)*IR)**{(PQ)*IP**(P*R)}**
*5.31  (**(RP)*Q)**{FI**(Q*R)}**
*5.32  (**(PI**(Q=R)**)=**(PQ)**(P*R)**)
*5.33  (**(P**(QIR))**(P**(PQ)**IR)**)
*5.35  (**(PIQ)**{PIR)**{PI**(QIR)}**)
*5.36  (**(P**(PQ)=Q)**(Q**(P=Q))**
*5.40  (**(PIPIQ)**{PIQ)**
*5.44  (**(PIQ)**{PIR)**{PI**(Q*R)}**)
*5.45  (**(PIQ)**{(PIQ)**}
*5.50  (**(PIQ)**{P(Q)}**
*5.51  (**(PIQ)**{P(Q)}**
*5.52  (**(IPQ)**{QIS)**{QIS)**(RIS)**}
*5.54  (**(PQ)**{P(V**(PQ)**Q)**
*5.55  (**(PVQ)**{P(V**(PVQ)**Q)**
*5.56  (**(P**(Q)**Q)**{P**(Q)**Q)**
*5.57  (**(P**(Q)**Q)**{P**(PVQ)**Q)**
*5.58  (**(PVQ)**{(PV**(PQ)**Q)**
*5.59  (**(P**(Q)**Q)**{P**(PVQ)**Q)**
*5.60  (**(QI-R)**{Q**(PVQ)****(P**(R)**)
*5.61  (**(QIR)**{Q**(PIQ)****(P**(R)**)
*5.62  (**(QI-R)**{Q**(PVQ)****(P**(Q)**}
*5.70  (**(QVR)**{(QVR)**{R**(P**(Q)**
*5.71  (**(PI**(Q)**R)**{P**(PVQ)****(P**(R)**)
*5.74  (**(QI**(Q)**R)**{P**(PIQ)****(P**(R)**)
*5.75  (**(R**(Q)**R)**{P**(QVR)****(P**(Q)**}**
OUTPUT

The output from the sample run shows that the run commences from a restart. After the loader listings, the true expressions and problems for the run are listed. Note that *2.33 failed and that all true expressions have had their bound variable replaced by free variables.

Next come the proof printouts. First the problem is printed, then the subproblems are printed in the order of generation. Then, if a proof is found, it is printed along with a statement of effort applied. Finally, each proved theorem that is remembered is printed as it appears with its bound variables replaced.

At the end of the run, an IPL-V post mortem printout appears with a printout of the true expressions map as it appeared at the conclusion of the run. At the end of this sample run the map uses approximately 500 cells and holds approximately 30 expressions, which is about 17 cells per expression. With more expressions in the map, it becomes considerably more efficient.
JOB 8168,LTNE1,8A5826,5MIN,0,073,C STEFFERUD 1 003650 06/05/63

ASSIGN A6=SYSARZ
ASSIGN B6=SYSARZ
IPL

LOGIC THEORIST TEST 9

RELOAD FROM TAPE 2 5 4 2
MEMORY RELOADED FROM TAPE A6, 3956 0 0 J166 J165
MODIFIED ROUTINES

671 4 0 24587 3976        Q17 4 0 W0  Q17 FIND LEVEL OF SUBSEGMENT
3976 6 0 24587 3975        6 0 W0  REPLACEMENT IN TEXT (0).
3977 1 0 671 3977        1 0 Q17  Q017R010
3977 0 0 24770 3978        1 J10  Q017R020
3978 7 0 3979 24790        7 0  J30  FIND CURRENT LEVEL.
3979 1 1 24587 3980        7 0  J30  IF NONE.
3980 0 0 656 3981        0 0  J30  Q017R030
3981 0 0 24790 3982        7 0  J30  Q017R040
3982 0 0 24880 3983        0 0  J30  Q017R050
3983 4 0 24574 3984        7 0  J30  Q017R060
3984 1 1 24587 3985        0 0  J30  COPY.
3985 0 0 24766 3986        0 0  J30  Q017R070
3986 1 0 671 3987        1 0  Q17  Q017R080
3987 3 0 24587 24771        1 0  Q17  Q017R090

AND ASSIGN AS CURRENT LEVEL.
RUN DATA HEADER

164 0 1 50 K20 + 1 50 LIMIT ON NUMBER OF SUBPROBLEMS K020000
165 0 1 50 K21 + 1 50 LIMIT ON NUMBER OF SUBSTITUTIONS K021000
166 0 1 000000 K22 + 1 20 0000 LIMIT ON EFFORT K022000
174 0 0 704 0 K30 R K= ADD PROVED THEOREMS TO THEOREMS K030000
175 0 0 334 0 K31 YES Y = PRINT REJECTED SUBPROBLEMS K031000
250 0 4 0 4016 L6 0 L6 LIST OF METHODS FOR ORIG PROBS L006000
4016 0 0 310 4017 M16 L006010
4017 0 0 311 0 M17 L006020
251 0 4 0 4018 L7 0 L7 LIST OF METHODS FOR PROBLEMS L007000
4018 0 0 307 4019 M13 REPLACEMENT L007010
4019 0 0 305 4020 M11 DETACHMENT L007010
4020 0 0 308 4021 M14 FORWARD CHAINING L007010
4021 0 0 309 0 M15 BACKWARD CHAINING L007010
24599 0 0 797 0 W12 X13 0 W12 SET-UP ENTRY SNAP ACTION W012000
24600 0 0 797 0 W13 X13 0 W13 SET-UP EXIT SNAP ACTION W013000
24602 0 0 799 0 W15 X15 0 W015000
805 0 4 0 0 X21 0 X210000
806 0 4 0 0 X22 0 X220000
807 0 4 0 0 X23 0 DESCRIPTION LIST OF TRAP ACTIONS X023000

KICK OFF FOR PROVING THEOREMS. 5 X1

END OF LOADING. PROGRAM STARTS AT X1 1 1 W26 3921

NUMBER OF CELLS ON AVAILABLE SPACE=17978
*1.01  (AIB) = -(AVB) DEF.
BAD EXPRESSION  ((PVQUINARY) = -(PVQV))
*3.01  (A+B) = -(1-AV-B) DEF.
*4.01  (A+B) = -(AIB+1IA) DEF.
*1.2   (AVA) I A
*1.3   B(AVR)
*1.4   (AVB) (A)A
*1.5   (AV(BVCD)) (B)(AVC)
*1.6   (B(C)) (A)(AVB) (AVC)

*2.01  (PI-P) I I = P
*2.02  QI(PQI)
*2.03  (PI-Q) I (Q-P)
*2.04  (PI(Q)I QI(P))
*2.05  (QI)I (I(QP)I(P))
*2.06  (QI)I (I(QR)I(Q))
*2.07  PI(PQP)
*2.08  PIP
*2.10  -PVP
*2.11  PV-P
*2.12  P--P
*2.13  PV--P
*2.14  --PIP
*2.15  (-PI(QI)I-QI)
*2.20  PI(PVQ)
*2.21  -PI(PIQ)
*2.24  PI(-P)
*3.13  (-PVQ)I(-PVQ)
*3.24  -P-P
*4.13  P=P
*4.24  P=(P=P)
*4.25  P=(PQP)
TO PROVE
2.01 (PI-P1I-P)
 1. (-PV-P)I-P

PROCEDUDE:

GIVEN
SUBSTITUTION
GIVEN
SUBLEVEL REPLACEMENT
Q.E.D.

*1.2 (AUA)I-A
1. (-PV-P)I-P
*2.01 DEFINITIONS
   (PI-P)I-P

EFFORT
SUBPROBLEMS LIMIT 200000 ACTUAL 8062
SUBSTITUTIONS LIMIT 50 ACTUAL 1

REMEMBER PROVED THEOREM
2.01 (AI-A)I-A
TO PROVE
*2.02\hspace{1em} \text{Qi}(PiQ)
\hspace{1.5em} 1. \hspace{1em} \text{Qi}(\neg P\neg Q)
\hspace{1.5em} \text{, Sublevel Replacement}

PROOF FOUND.

\text{Given Substitution}
\text{Given Sublevel Replacement}
Q.E.D.

\text{*1.3} \hspace{1em} \text{Bi}(AvB)
\hspace{1.5em} 1. \hspace{1em} \text{Qi}(\neg P\neg Q)
\hspace{1.5em} \text{Definitions}

\text{*2.02} \hspace{1em} \text{Qi}(PiQ)

\text{Effort Subproblems Substitutions}
\text{Limit 200000 Limit 50 Limit 50}
\text{Actual 6041 Actual 1 Actual 2}

\text{Remember Proved Theorem}

\text{*2.02} \hspace{1em} \text{Ai}(BiA)
TO PROVE
*2.03 [PI-Q][I(QI-P)]
  1. (-PV-Q)[I(-QV-P)], SUBLEVEL REPLACEMENT

PROOF FOUND.
GIVEN
  *1.4 (AVB)(BVA)
  SUBSTITUTION
  GIVEN
  1. (-PV-Q)[I(-QV-P)], DEFINITIONS
  SUBLEVEL REPLACEMENT
  *2.03 (PI-Q)[I(QI-P)]
  G.E.Q.

EFFORT
SUBPROBLEMS LIMIT 200000 ACTUAL 11904
SUBSTITUTIONS LIMIT 50 ACTUAL 1

REMEMBER PROVED THEOREM
*2.03 (AI-B)[I(BI-A)]
TO PROVE
*2.04 (PI(QIR))((QI(PIR))
  1. (PI(QVR))((QI(PVR))
  2. (-PV(QIR))((QV(PIR))
  3. (-PV(QVR))((QV(PVR))

; SUBLEVEL REPLACEMENT
; SUBLEVEL REPLACEMENT
; SUBLEVEL REPLACEMENT

PROOF FOUND.

GIVEN

SUBSTITUTION

*1.5 (AV(BVC))((AV(CV))

GIVEN

DEFINITIONS

SUBLEVEL REPLACEMENT

*2.04 (-PV(QVR))((QV(PVR))

G.E.O.

*2.04 (PI(QIR))((QI(PIR))

EFFORT

SUBPROBLEMS

LIMIT 200000

LIMIT 50

ACTUAL 32262

ACTUAL 3

SUBSTITUTIONS

LIMIT 50

ACTUAL 4

REMEMBER PROVED THEOREM

*2.04 (AI(BIC))((BI(AIC))
IC PROVE
*2.05 (QIR)((PIQ)((PIR)))
  1. (QIR)(((PVQ))) (PVR))

SUBLEVEL REPLACEMENT

PROOF FOUND.

GIVEN

SUBSTITUTION

GIVEN

SUBLLEVEL REPLACEMENT

Q.E.D.

EFFECT

LIMIT 200000

SUBPROBLEMS

LIMIT 50

SUBSTITUTIONS

LIMIT 50

ACTUAL 13440

ACTUAL 1

ACTUAL 2

REMEMBER PROVED THEOREM

*2.05 (AIB)((CIA)((CIB)))
IC PROVE

*2.06 (PIQ) I I I (QIR) I I (PIR)

1. (PIQ) I I I (QIR) I I (PIR)  
2. (-PIQ) I I I (QIR) I V(PIR)  
3. (-PIQ) I I I (QIR) I W(-PIR)  
4. (PIQ) I V(QIR) I I (PIR)  
5. (QIR) I I I (PIQ) I I (PIR)  

SUBLEVEL REPLACEMENT
SUBLEVEL REPLACEMENT
SUBLEVEL REPLACEMENT
1.01, REPLACEMENT
2.04, DETACHMENT

PROOF FOUND.

GIVEN
SUBSTITUTION
GIVEN
DETACHMENT

*2.05 (AIB) I I I (CIA) I I (CIR)
5. (QIR) I I I (PIQ) I I (PIR)
*2.04 (AIB) I I I (BIC) I I I (AIC)
*2.06 (PIQ) I I I (QIR) I I (PIR)

G.E.D.

EFFORT
SUBPROBLEMS LIMIT 200000 ACTUAL 46665
SUBSTITUTIONS LIMIT 50 ACTUAL 5

REMEMBER PROVED THEOREM

*2.06 (AIB) I I I (BIC) I I I (AIC)
TO PROVE
*2.07  P1(PVP)

PROOF FOUND.

GIVEN

SUBSTITUTION

Q.E.D.

*1.3  BI(AVR)

*2.07  P1(PVP)

EFFECT

LIMIT 200000

ACTUAL 2623

SUBPROBLEMS

LIMIT 50

ACTUAL 0

SUBSTITUTIONS

LIMIT 50

ACTUAL 1

REMEMBER PROVED THEOREM

*2.07  AI(AVA)
TC PROVE
*2.08 PIP
1. ~PVP
   5088 P
   5193 (PIP)~(PIP)
2. (PVP)IP
   *1.01, REPLACEMENT
   *2.02, DETACHMENT, REJECTED PROBLEM
   *1.02, DETACHMENT, REJECTED PROBLEM
   *2.07, FORWARD CHAINING

PROOF FOUND.
GIVEN
SUBSTITUTION
GIVEN
FORWARD CHAINING
Q.E.D.

EFFORT LIMIT 200000 ACTUAL 8998
SUBPROBLEMS LIMIT 50 ACTUAL 2
SUBSTITUTIONS LIMIT 50 ACTUAL 3

REMEMBER PROVED THEOREM
*2.09 AIA

*1.2 (AVA)IA
*1.2 (PVP)IP
*2.07 AI(AVA)
*2.08 PIP
TO PROVE
*2.10 -PVP
  1. PIP

PROOF FOUND.
GIVEN
  1. PIP
STUBSTITUTION
GIVEN
  *1.01 (A|B)|.|(-A|B) DEF.
REPLACEMENT
G.E.D.

*2.08 AIA
*2.10 -PVP

EFFORT
SUBPROBLEMS LIMIT 200000 ACTUAL 5150
SUBSTITUTIONS LIMIT 50 ACTUAL 1

REMEMBER PROVED THEOREM
*2.10 -AVA
To prove

*2.11: PV-P

1. -PV

*1.4* DETACHMENT

proof found.

GIVEN

SUBSTITUTION

GIVEN

DETACHMENT

Q.E.D.

*2.10: AVA

1. -PV

*1.4: (AVA)(PVA)

*2.11: PV-P

Effort

<table>
<thead>
<tr>
<th>Limit</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>200000</td>
<td>6190</td>
</tr>
</tbody>
</table>

Subproblems

<table>
<thead>
<tr>
<th>Limit</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>1</td>
</tr>
</tbody>
</table>

Substitutions

<table>
<thead>
<tr>
<th>Limit</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>2</td>
</tr>
</tbody>
</table>
TO PROVE
*2.12  P1--p                   *1.01, REPLACEMENT
  1.  -Pv--P

PROOF FOUND.

GIVEN

SUBSTITUTION

GIVEN

REPLACEMENT

Q.E.D.

*2.11  AV--A

1.  -Pv--p

*1.01  (A&V)&=(A&V) DEF.

*2.12  P1--p

EFFECT

SUBPROBLEMS  LIMIT 200000  ACTUAL 8215

SUBSTITUTIONS  LIMIT 50  ACTUAL 1

REMENBER PROVED THEOREM

*2.12  A1--A

EFFECT

SUBPROBLEMS  LIMIT 50  ACTUAL 2
TO PROVE

\[ PV \rightarrow P \]

1. \[ \text{---}PV \]
   3062 \[ \text{---}P \]
   5304 \[ PV \rightarrow P \]
   5199 \[ (PV \rightarrow P) \lor (PV \rightarrow P) \]

2. \[ \text{---}P \]

\[ \text{---}P \]

*1.4, DETACHMENT
*1.3, DETACHMENT, REJECTED PROBLEM
*2.09x, DETACHMENT, REJECTED PROBLEM
*1.2, DETACHMENT, REJECTED PROBLEM
*2.11, FORWARD CHAINING

PROOF FOUND.

GIVEN
SUBSTITUTION
GIVEN
FORWARD CHAINING
Q.E.D.

\[ *2.12 \]
\[ AI \rightarrow A \]
\[ *2.11 \]
\[ AV \rightarrow A \]
\[ *2.13 \]
\[ PV \rightarrow P \]

EFFECT

LIMIT 200000
SUBPROBLEMS
LIMIT 50
SUBSTITUTIONS
LIMIT 50

ACTUAL 18263
ACTUAL 2
ACTUAL 3

REMEMBER PROVED THEOREM

\[ *2.13 \]
\[ AV \rightarrow A \]
TO PROVE
*2.14  --PVP

1. --PVP  *1.01. REPLACEMENT
  5187  --PVP  *2.02. DETACHMENT. REJECTED PROBLEM
  5090  --PVP  *2.08. DETACHMENT. REJECTED PROBLEM
  5149  [--PPI]V(--PPI)  *1.2. DETACHMENT. REJECTED PROBLEM
  3. [--PV--PPI]P  *2.12. FORWARD CHAINING
  5339  --PVP  *2.08. FORWARD CHAINING. REJECTED PROBLEM
  4. [B1--PI]P(PVP)  *2.07. FORWARD CHAINING
  5379  --PPI  *1.3. FORWARD CHAINING
  6. --PI[PVP]  *2.08. BACKWARD CHAINING. REJECTED PROBLEM

6. --PI[PVP]

7. --PVP(PVP)  *1.01. REPLACEMENT
  5051  PVP  *2.02. DETACHMENT. REJECTED PROBLEM
  5245  --PI[PVP]  *2.08. DETACHMENT. REJECTED PROBLEM
  5552  ([--PPI]P)IV(--PI[PVP])  *1.2. DETACHMENT. REJECTED PROBLEM
  8. --PI[PVP]  *2.12. FORWARD CHAINING
  5612  --PPI[PVP]  *2.08. FORWARD CHAINING. REJECTED PROBLEM
  9. [B1--PI]P(PVP)  *2.07. FORWARD CHAINING
  10. (AV--PI)P(PVP)  *1.3. FORWARD CHAINING
  11. (AV--PI)P(PVP)  *2.07. BACKWARD CHAINING. REJECTED PROBLEM
  5550  --PVP  *1.4. BACKWARD CHAINING. REJECTED PROBLEM
  5737  --PI(PVP)  *1.3. BACKWARD CHAINING. REJECTED PROBLEM
  5780  --PVP  *2.08. BACKWARD CHAINING. REJECTED PROBLEM
  5653  --PI(PVP)

12. --PII[PVP]IIV(PVP)

13. --PV(PVP)I(PVP)

5286  (PVP)IV(PVP)  *1.01. REPLACEMENT
  5162  --PII(PVP)  *2.02. DETACHMENT. REJECTED PROBLEM
  5854  ([--PPI]P)IIV([--PII(PVP)I(PVP)])I12. DETACHMENT. REJECTED PROBLEM
  14. --PII(PVP)I(PVP)  *2.12. FORWARD CHAINING
  5936  --PII(PVP)I(PVP)  *2.08. FORWARD CHAINING. REJECTED PROBLEM
  15. (B1--PII(PVP)I(PVP))  *2.07. FORWARD CHAINING
  16. (AV--PII(PVP)I(PVP))  *1.3. FORWARD CHAINING
  6031  --PII(PVP)
  18. --PII(PVP)I(PVP)  *2.08. BACKWARD CHAINING. REJECTED PROBLEM
  6059  --PII(PVP)I(PVP)
  6092  --PII(PVP)
  5995  --PII(PVP)I(PVP)IIV(PVP)I(PVP))  *1.2. BACKWARD CHAINING

1. --PVP
  6102  --PVP  *1.01. REPLACEMENT. REJECTED PROBLEM
  20. PV---P  *1.4. DETACHMENT

PROOF FOUND.

GIVEN

AV---A  *2.13
PV---P  *2.14

SUBSTITUTION

GIVEN

DETACHMENT

GIVEN

REPLACEMENT

C.E.D.
<table>
<thead>
<tr>
<th>EFFORT</th>
<th>LIMIT 200000</th>
<th>ACTUAL 181261</th>
</tr>
</thead>
<tbody>
<tr>
<td>SUBPROBLEMS</td>
<td>LIMIT 50</td>
<td>ACTUAL 20</td>
</tr>
<tr>
<td>SUBSTITUTIONS</td>
<td>LIMIT 50</td>
<td>ACTUAL 21</td>
</tr>
</tbody>
</table>

REMEMBER PROVED THEOREM

#2.14 --AIA
IC PROVE

*2.19 (-P(Q)(I)(-Q)(I))

1. (--P(Q)(I)(--Q)(P)) SUBLEVEL REPLACEMENT
2. 5126 (--P(Q)(I)(--Q)(P)) SUBLEVEL REPLACEMENT. REJECTED PROBLEM
3. (-P(I)(-Q)(I)) 1.01, REPLACEMENT
4. -P(I) 2.04, DETACHMENT
5. -P(I) 2.02, DETACHMENT
6. -P(I) 2.14, DETACHMENT
7. (-P(Q)(I)(-Q)(I)) 2.08, DETACHMENT. REJECTED PROBLEM
8. 5273 (-P(Q)(I)(-Q)(I)) 1.2, DETACHMENT. REJECTED PROBLEM
9. (P(Q)(I)(-P)(I)(I)(-Q)(I)) 2.09, FORWARD CHAINING
10. (P(Q)(I)(-P)(I)(I)(-Q)(I)) 2.06, FORWARD CHAINING
11. (P(Q)(I)(-P)(I)(I)(-Q)(I)) 2.15, FORWARD CHAINING
12. (P(Q)(I)(-P)(I)(I)(-Q)(I)) 2.14, FORWARD CHAINING
13. (P(Q)(I)(-P)(I)(I)(-Q)(I)) 2.09, FORWARD CHAINING
14. (P(Q)(I)(-P)(I)(I)(-Q)(I)) 2.08, FORWARD CHAINING. REJECTED PROBLEM
15. (P(Q)(I)(-P)(I)(I)(-Q)(I)) 1.2, BACKWARD CHAINING

4. -P(I)

16. --Q(P) 1.01, REPLACEMENT
17. 5882 P 2.02, DETACHMENT. REJECTED PROBLEM
18. (-P(I)(P)) 2.14, DETACHMENT
19. 6171 -P(I) 2.08, DETACHMENT. REJECTED PROBLEM
20. (-P(I)(P)) 1.2, DETACHMENT. REJECTED PROBLEM
21. 5396 (-P)(I)(-Q)(I) 2.12, FORWARD CHAINING
22. -P(I) 2.08, FORWARD CHAINING. REJECTED PROBLEM
23. -P(I) 2.07, FORWARD CHAINING
24. -P(I) 2.02, FORWARD CHAINING
25. -P(I) 1.3, FORWARD CHAINING
26. -P(I) 2.14, BACKWARD CHAINING
27. -P(I) 2.08, BACKWARD CHAINING. REJECTED PROBLEM
28. -P(I) 1.2, BACKWARD CHAINING

23. -Q(P)(P)

24. --Q(V)(P)(P) 1.01, REPLACEMENT
25. 5836 -Q(V)(P)(P) 2.02, DETACHMENT. REJECTED PROBLEM
26. 6306 -Q(P)(P) 2.14, DETACHMENT
27. 6420 (-Q(P)(P))(--Q(P)(P)) 2.08, DETACHMENT. REJECTED PROBLEM
28. 6459 (-Q(P)(P)) 1.2, DETACHMENT. REJECTED PROBLEM
29. 6466 (-Q(P)(P)) 2.12, FORWARD CHAINING
30. 6459 (-Q(P)(P)) 2.08, FORWARD CHAINING. REJECTED PROBLEM
31. 6466 (-Q(P)(P)) 2.07, FORWARD CHAINING
32. 6580 (-Q(P)(P)) 1.3, FORWARD CHAINING
33. 6580 -Q(P)(P) 2.07, BACKWARD CHAINING. REJECTED PROBLEM
34. 6657 -Q(P)(P) 1.4, BACKWARD CHAINING.
35. 6657 -Q(P)(P) 1.3, BACKWARD CHAINING. REJECTED PROBLEM
36. 6657 -Q(P)(P) 2.14, BACKWARD CHAINING
37. 6657 -Q(P)(P) 2.08, BACKWARD CHAINING. REJECTED PROBLEM
38. 6657 -Q(P)(P) 1.2, BACKWARD CHAINING

NC PROOF FOUND

EFFGAT LIMIT 200000 ACTUAL 206728
SUBPROBLEMS LIMIT 50 ACTUAL 31
SUBSTITUTIONS LIMIT 50 ACTUAL 32
TO PROVE
*2.20 $\Pi(P\neg Q)$

1. $\neg\Pi(P\neg Q)$
2. $P\neg Q$
3. $\neg\left(\Pi(P\neg Q)\right)$
4. $\neg\Pi(P\neg Q)$
5. $(P\neg Q)\Pi(P\neg Q)$
6. $(B\Pi)(P\neg Q)$
7. $(A\Pi)(P\neg Q)$

*1.01, REPLACEMENT
*2.02, DETACHMENT
*2.14, DETACHMENT
*2.08, DETACHMENT, REJECTED PROBLEM
*1.12, DETACHMENT, REJECTED PROBLEM
*2.12, FORWARD CHAINING
*2.08, FORWARD CHAINING, REJECTED PROBLEM
*2.07, FORWARD CHAINING
*2.02, FORWARD CHAINING
*1.3, FORWARD CHAINING

PROOF FOUND.

GIVEN
SUBSTITUTION
GIVEN
FORWARD CHAINING
Q.E.D.

*1.4 $(A\Pi)(B\Pi)$
7. $(C\Pi)(P\Pi)$
*1.3 $B\Pi(A\Pi)$
*2.20 $\Pi(P\Pi)$

EFFORT LIMIT 200000 ACTUAL 38111
SLAPROBLEMS LIMIT 50 ACTUAL 7
SUBSTITUTIONS LIMIT 50 ACTUAL 8

REMEMBER PROVED THEOREM

*2.20 $A\Pi(A\Pi)$
IC PROVE
   *2.21  -P1(P1G)
      1.  -P1(-PVG)

   PROOF FOUND.

   GIVEN
   SUBSTITUTION
   GIVEN
   SUBLEVEL REPLACEMENT
   Q.E.D.

   *2.20  AI(AVB)
      1.  -P1(-PVG)
   DEFINITIONS
   *2.21  -P1(P1)

   EFFORT
   SUBPROBLEMS  LIMIT 200000  ACTUAL 8021
   SUBSTITUTIONS  LIMIT 50  ACTUAL 1

   REMEMBER PROVED THEOREM

   *2.21  -AI(AIB)
To prove \( P(-P(Q)) \)

1. \( P(-P(Q)) \)
2. \( -P(-P(Q)) \)
3. \( -P(P(Q)) \)

\underline{Sublevel Replacement}  
\underline{Sublevel Replacement. Rejected Problem}  
\underline{*1.01, Replacement}  
\underline{*2.21, Detachment. Rejected Problem}  
\underline{*2.04, Detachment}

**Proof Found.**

1. Given
2. Substitution
3. Given
4. Detachment

\( \text{Q.E.D.} \)

**Error**

Limit 200000  
Actual 21521

**Subproblems**

Limit 50  
Actual 3

**Substitutions**

Limit 50  
Actual 4

Remember prove theorem

\( *2.24 \quad AI(AIB) \)
-101-

TO PROVE

*3.13  \((-P\land Q)\lor (-P\land Q)\)
    1.  \((-L\land P\land Q)\lor (-L\land P\land Q)\)

PROOF FOUND.

GIVEN

SUBSTITUTION

GIVEN

SUBLEVEL REPLACEMENT

Q.E.D.

* 2.14  --AIA

1.  \((-L\land P\land Q)\lor (-L\land P\land Q)\)

DEFINITIONS

* 3.13  \((-P\land Q)\lor (-P\land Q)\)

EFFECT

LIMIT 200000  ACTUAL 13293

SUBPROBLEMS

LIMIT 50  ACTUAL 1

SUBSTITUTIONS

LIMIT 50  ACTUAL 2

REMEMBER PROVED THEOREM

*3.13  \((-A\land B)\lor (-A\land B)\)
TO PROVE
*3.4 \((P \lor Q) \land \neg (P \land Q)\)
\[1. \quad \neg (P \lor Q) \land \neg \neg (P \lor Q)\]  
SUBLEVEL REPLACEMENT

PROOF FOUND.

\text{GIVEN}
\text{SUBSTITUTION}
\text{GIVEN}
\text{SUBLEVEL REPLACEMENT}
Q.E.D.

\text{EFFORT} \quad \text{LIMIT 200000} \quad \text{ACTUAL 12753}
\text{SUBPROBLEMS} \quad \text{LIMIT 50} \quad \text{ACTUAL 1}
\text{SUBSTITUTIONS} \quad \text{LIMIT 50} \quad \text{ACTUAL 2}

\text{REMEMBER PROVED THEOREM}

*3.4 \((A \lor \neg B) \land (A \land \neg B)\)
TO PROVE

\*3.24 \ - (P \* P)

1. \ - (\neg (\neg P \* P))
2. S417 \ - (\neg (\neg P \* P))

\* SUBLEVEL REPLACEMENT

\*3.14, DETACHMENT

REJECTED PROBLEM

PROOF FOUND.

GIVEN

\*2.11 \ A \* A
2. \ - P \* P
3. S14 \ (\neg A \* B) \ (\neg (A \* B))
4. \*3.24 \ (P \* P)

GIVEN

DETACHMENT

Q.E.D.

EFFORT LIMIT 200000 ACTUAL 4541

SUBPROBLEMS LIMIT 50 ACTUAL 2

SUBSTITUTIONS LIMIT 50 ACTUAL 3

REMEMBER PROVED THEOREM

\*3.24 \ - (A \* A)
TO PROVE
*4.13  P==P
1. (P==P)\land\neg(P==P)
2. \neg\neg(P==P)
5588  P==P
5260  (P==P)\land(P==P)
3.  P==P

*4.01, REPLACEMENT
*2.14, DETACHMENT
*2.08, DETACHMENT, REJECTED PROBLEM
*1.2, DETACHMENT, REJECTED PROBLEM
*2.12, BACKWARD CHAINING

PROOF FOUND.

GIVEN

AIA
*2.08

SUBSTITUTION
3.  P==P

GIVEN

*2.12  A1==A

BACKWARD CHAINING

*4.13  P==P

Q.E.D.

EFFORT

LIMIT 200000

ACTUAL 15965

SUBPROBLEMS

LIMIT 50

ACTUAL 3

SUBSTITUTIONS

LIMIT 50

ACTUAL 4

REMEMBER PROVED THEOREM

*4.13  A==A
TC PROVE
*4.20 P=P
1. (P\&P)\&(P\&P)
2. \neg(\neg(P\&P))
6082 P=P
5765 (P\&P)\&(P\&P)
3. \neg P\&P

*4.01, REPLACEMENT
*2.14, DETACHMENT
*2.08, DETACHMENT, REJECTED PROBLEM
*1.2, DETACHMENT, REJECTED PROBLEM
*4.13, FORWARD CHAINING

PROOF FOUND.
GIVEN
SUBSTITUTION
GIVEN
FORWARD CHAINING
Q.E.D.

*2.14 -- A\&A
3. -- P\&P
*4.13 -- A\&A
*4.20 P=P

EFFORT
SUBPROBLEMS LIMIT 200000 ACTUAL 12404
SUBSTITUTIONS LIMIT 50 ACTUAL 3

REMEMBER PROVED THEOREM
*4.20 A=A
TO PROVE

*4.24  P=(P*P)

\[\begin{align*}
1. & \quad P=(-(P*P)) \\
2. & \quad P=(P*P) \\
3. & \quad -(P*P) \\
4. & \quad P=(P*P) \\
5. & \quad P=(-(P*P)) \\
6. & \quad P=(P*P) \\
7. & \quad P=(P*P) \\
8. & \quad P=(P*P) \\
9. & \quad P=(P*P) \\
10. & \quad -(P=(P*P)) \\
11. & \quad -(P=(P*P)) \\
12. & \quad -(P=(P*P)) \\
13. & \quad -(P=(P*P)) \\
14. & \quad -(P=(P*P)) \\
15. & \quad -(P=(P*P)) \\
16. & \quad -(P=(P*P)) \\
17. & \quad P=(P*P)
\end{align*}\]

*4.25  SUBLEVEL REPLACEMENT

SUBLEVEL REPLACEMENT. REJECTED PROBLEM

*4.01  REPLACEMENT

*2.14  DETACHMENT

*2.08  DETACHMENT. REJECTED PROBLEM

*1.2  DETACHMENT. REJECTED PROBLEM

*4.20  FORWARD CHAINING

*4.13  FORWARD CHAINING

*2.14  BACKWARD CHAINING

*2.08  BACKWARD CHAINING. REJECTED PROBLEM

*1.2  BACKWARD CHAINING

**PROOF FOUND.

GIVEN

SUBSTITUTION

GIVEN

BACKWARD CHAINING

GIVEN

DETACHMENT

GIVEN

FORWARD CHAINING

Q.E.D.

EFFORT

SUBPROBLEMS

LIMIT 200000

ACTUAL 92331

SUBSTITUTIONS

LIMIT 50

ACTUAL 17

ACTUAL 18

REMEMBER PROVED THEOREM

*4.24  A=(A*A)
TO PROVE
*4.25  P=(PVP)
  1.  (PI(PVP))\*((PVP)IP)
  2.  -(P=(P(VP))
  5287  P=(PVP)
  3.  (P=(PVP))\*VP=(PVP)
  4.  PI(PVP)

PROOF FOUND.

GIVEN
SUBSTITUTION
GIVEN
FORWARD CHAINING
C.E.D.

EFFECT
SUBPROBLEMS  LIMIT 200000
SUBSTITUTIONS  LIMIT 50

ACTUAL 20788
ACTUAL 4
ACTUAL 9

REMEMBER PROVED THEOREM

*4.25  A=\{AVA\}
PROGRAM RAN TO COMPLETION.

IPL-V POST-MRTERM

H0   #1  0
    #2  J7
    #3
    #4
H1
H2  16207 WORDS
H3  0
H4  J4
H5  0
H6
H7
H8  0
H9  0
H10 0
H11 0
H12 0
W0  0
W1  0  24627
W2  0
W3  0
W4  0
W5  0
W6  0
W7  0
W8  0
W9  0
W10 24627  1
W11  0
W12 X13
W13 X13
W14 J0
W15 X15
W16
W17
W18  24628  0
W19  24629  0
W20  28153  0
W21  24630  0
W22  24631  1
W23  24668  1
W24  24632  1
W25  28090  1
W26  24633
W27  24754  16
W28  24756
W29  0
W30  24755  1
W31  24759  2
THE FOLLOWING RESULTED FROM EXECUTING 1W15...

L4
  0
  -0
  9-1
  9-2
  V0
  9-3
  10
  9-4
  *1
  9-5
  5916* 9-1
  0
  9-6
  9-7
  4849* 9-2
  0
  9-8
  5095* 9-3
  0
  9-9
  9-10
  4247* 9-4
  0
  9-11
  9-12
  4054* 9-5
  0
  9-13
  9-14
  5707* 9-6
  9-15
  5719* 9-7
  9-16
  V0
  9-17
  *0
  9-18
  9-19
  5398* 9-8
  0
  *0
  9-20
  5059* 9-9
  9-21
  -0
  9-22
  5138* 9-10
  9-23
  -0
5789  9-17  9-74  9-75  9-76  9-77  9-78  9-79  9-80  9-81  9-82
5765  9-18  9-83  9-84  9-85  9-86  9-87  9-88  9-89  9-90  9-91

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
| 5109 | 9-21 | 0 |   |   |   |   |   |   | 0 |   |   |   |   | 9-48 |   |   |   |   |   |
| 4368 | 9-22 | 0 |   |   |   |   |   |   | 0 |   |   |   | 9-49 |   | 9-50 |   |   |   |   |   |
| 4258 | 9-23 | 0 |   |   |   |   |   |   | 0 | 9-51 |   |   |   | 9-52 |   |   |   |   |   |   |
| 4242 | 9-24 | 0 |   |   |   |   |   |   | 0 |   |   |   |   |   |   |   |   |   |   |   |
| 4875 | 9-25 | 0 |   |   |   |   |   |   | 0 |   |   |   | 9-53 | 9-54 |   |   |   |   |   |   |
| 4371 | 9-26 | 0 |   |   |   |   |   |   | 0 |   |   | 9-55 |   |   |   |   |   |   |   |   |
| 4253 | 9-27 | 0 |   |   |   |   |   |   | 0 | 9-56 |   |   |   |   |   |   |   |   |   |   |
| 4101 | 9-28 | 0 |   |   |   |   |   |   | 0 |   |   |   | 9-57 |   |   |   |   |   |   |   |
| 4131 | 9-29 | 0 |   |   |   |   |   |   | 0 |   |   |   |   | 9-58 |   |   |   |   |   |   |
| 4264 | 9-30 | 0 |   |   |   |   |   |   | 0 |   |   |   |   |   | 9-59 |   |   |   |   |   |
| 4204 | 9-31 | 0 |   |   |   |   |   |   | 0 |   |   |   |   |   |   | 9-60 |   |   |   |   |
| 4113 | 9-32 | 0 |   |   |   |   |   |   | 0 |   |   |   |   |   |   |   | 9-61 |   |   |   |
| 4069 | 9-33 | 0 |   |   |   |   |   |   | 0 |   |   |   |   |   |   |   |   | 9-62 |   |   |
| 5585 | 9-34 | 0 |   |   |   |   |   |   | 0 |   |   |   |   |   |   |   |   |   | 9-63 |   |
| 5171 | 9-35 | 0 |   |   |   |   |   |   | 0 |   |   |   |   |   |   |   |   |   |   | 9-64 |
| 6331 | 9-36 | 0 |   |   |   |   |   |   | 0 |   |   |   |   |   |   |   |   |   |   |   | 9-65 |
| 6109 | 9-37 | 0 |   |   |   |   |   |   | 0 |   |   |   |   |   |   |   |   |   |   |   | 9-66 |
| 5524 | 9-38 | 0 |   |   |   |   |   |   | 0 |   |   |   |   |   |   |   |   |   |   |   | 9-67 |
| 5819 | 9-39 | 0 |   |   |   |   |   |   | 0 |   |   |   |   |   |   |   |   |   |   |   | 9-68 |
| 5411 | 9-40 | 0 |   |   |   |   |   |   | 0 |   |   |   |   |   |   |   |   |   |   |   | 9-69 |
| 5010 | 9-41 | 0 |   |   |   |   |   |   | 0 |   |   |   |   |   |   |   |   |   |   |   | 9-70 |
| 4983 | 9-42 | 0 |   |   |   |   |   |   | 0 |   |   |   |   |   |   |   |   |   |   |   | 9-71 |
| 5217 | 9-43 | 0 |   |   |   |   |   |   | 0 |   |   |   |   |   |   |   |   |   |   |   | 9-72 |
| 5217 | 9-44 | 0 |   |   |   |   |   |   | 0 |   |   |   |   |   |   |   |   |   |   |   | 9-73 |
| 5410 | 9-45 | 0 |   |   |   |   |   |   | 0 |   |   |   |   |   |   |   |   |   |   |   | 9-74 |
| 5010 | 9-46 | 0 |   |   |   |   |   |   | 0 |   |   |   |   |   |   |   |   |   |   |   | 9-75 |
| 4983 | 9-47 | 0 |   |   |   |   |   |   | 0 |   |   |   |   |   |   |   |   |   |   |   | 9-76 |
| 5217 | 9-48 | 0 |   |   |   |   |   |   | 0 |   |   |   |   |   |   |   |   |   |   |   | 9-77 |
| 5217 | 9-49 | 0 |   |   |   |   |   |   | 0 |   |   |   |   |   |   |   |   |   |   |   | 9-78 |
| 5217 | 9-50 | 0 |   |   |   |   |   |   | 0 |   |   |   |   |   |   |   |   |   |   |   | 9-79 |
4073* 9-67  0
   -0
   9-116
4082* 9-68  9-117
5266* 9-69  0
   *425
5274* 9-70  0
   *425
5609* 9-71  0
   *424
6002* 9-72  0
   *424
6077* 9-73  0
   9-118
6107* 9-74  0
   324
5174* 9-75  0
   9-119
5136* 9-76  0
4995* 9-77  0
   *210
5080* 9-78  0
   *211
5419* 9-79  0
   9-120
5608* 9-80  0
   *221
5089* 9-81  0
   9-121
4369* 9-82  0
   *206
   *205
   *204
   *203
   *201
   *16
4257* 9-83  0
   *206
   *205
   *16
5054* 9-84  0
   9-124
   9-125
4956* 9-85  0
   9-126
4254* 9-86  0
   *15
   *14
   12
5821* 9-87  0
   9-127
4256* 9-88  0
   *14
   *12
4795* 9-89  0
   9-128
4295* 9-90  0
<p>| 4050 &amp; 9-111 &amp; 0 &amp; 0.301 |
| 4070 &amp; 9-112 &amp; 0 &amp; 0.101 |
| 4210 &amp; 9-113 &amp; 0 &amp; 0.101 |
| 4215 &amp; 9-114 &amp; 9-148 &amp; 9-149 |
| 4216 &amp; 9-115 &amp; 0 &amp; 9-150 |
| 4148 &amp; 9-116 &amp; 0 &amp; 9-151 |
| 4076 &amp; 9-117 &amp; 0 &amp; 9-152 |
| 4075 &amp; 9-118 &amp; 0 &amp; 9-153 |
| 4075 &amp; 9-119 &amp; 0 &amp; 9-154 |
| 4075 &amp; 9-120 &amp; 0 &amp; 9-155 |
| 5854 &amp; 9-121 &amp; 0 &amp; 9-156 |
| 5579 &amp; 9-122 &amp; 0 &amp; 9-157 |
| 5604 &amp; 9-123 &amp; 9-158 |
| 5653 &amp; 9-124 &amp; 9-159 |
| 6080 &amp; 9-125 &amp; 9-160 |
| 4839 &amp; 9-126 &amp; 9-161 |
| 4947 &amp; 9-127 &amp; 9-162 |
| 4948 &amp; 9-128 &amp; 9-163 |
| 5122 &amp; 9-129 &amp; 9-164 |
| 5123 &amp; 9-130 &amp; 9-165 |
| 5379 &amp; 9-131 &amp; 9-166 |
| 5475 &amp; 9-132 &amp; 9-167 |
| 5542 &amp; 9-133 &amp; 9-168 |
| 5538 &amp; 9-134 &amp; 9-169 |
| 5538 &amp; 9-135 &amp; 9-170 |
| 6130 &amp; 9-136 &amp; 9-171 |
| 6125 &amp; 9-137 &amp; 9-172 |
| 5125 &amp; 9-138 &amp; 9-173 |
| 5125 &amp; 9-139 &amp; 9-174 |
| 4401 &amp; 9-140 &amp; 9-175 |
| 4401 &amp; 9-141 &amp; 9-176 |
| 4401 &amp; 9-142 &amp; 9-177 |
| 4401 &amp; 9-143 &amp; 9-178 |
| 4401 &amp; 9-144 &amp; 9-179 |
| 4401 &amp; 9-145 &amp; 9-180 |
| 5655 &amp; 9-146 &amp; 9-181 |
| 5655 &amp; 9-147 &amp; 9-182 |
| 4345 &amp; 9-148 &amp; 9-183 |
| 4324 &amp; 9-149 &amp; 9-184 |
| 4213 &amp; 9-150 &amp; 9-185 |
| 4195 &amp; 9-151 &amp; 9-186 |
| 4214 &amp; 9-152 &amp; 9-187 |
| 4220 &amp; 9-153 &amp; 9-188 |
| 413 &amp; 9-154 &amp; 0 |
| 4145 &amp; 9-155 &amp; 0 |
| 4145 &amp; 9-156 &amp; 0 |</p>
<table>
<thead>
<tr>
<th>Code</th>
<th>Category</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>4421</td>
<td>9-180</td>
<td>0</td>
</tr>
<tr>
<td>4807</td>
<td>9-181</td>
<td>0</td>
</tr>
<tr>
<td>5917</td>
<td>9-182</td>
<td>0</td>
</tr>
<tr>
<td>4356</td>
<td>9-183</td>
<td>0</td>
</tr>
<tr>
<td>4356</td>
<td>9-184</td>
<td>0</td>
</tr>
<tr>
<td>4216</td>
<td>9-185</td>
<td>0</td>
</tr>
<tr>
<td>4216</td>
<td>9-186</td>
<td>0</td>
</tr>
<tr>
<td>4227</td>
<td>9-187</td>
<td>0</td>
</tr>
<tr>
<td>4229</td>
<td>9-188</td>
<td>0</td>
</tr>
<tr>
<td>4131</td>
<td>9-189</td>
<td>0</td>
</tr>
<tr>
<td>4155</td>
<td>9-190</td>
<td>9-193</td>
</tr>
<tr>
<td>4097</td>
<td>9-191</td>
<td>9-194</td>
</tr>
<tr>
<td>9207</td>
<td>9-192</td>
<td>9-195</td>
</tr>
<tr>
<td>4195</td>
<td>9-193</td>
<td>9-196</td>
</tr>
<tr>
<td>4195</td>
<td>9-194</td>
<td>9-197</td>
</tr>
<tr>
<td>5261</td>
<td>9-195</td>
<td>0</td>
</tr>
<tr>
<td>4161</td>
<td>9-196</td>
<td>0</td>
</tr>
<tr>
<td>4168</td>
<td>9-197</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>#301</td>
</tr>
</tbody>
</table>
ACCOUNTING SUMMARY

<table>
<thead>
<tr>
<th>PHASE</th>
<th>PROGRAM</th>
<th>TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>IPL</td>
<td>04MIN 30SEC</td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td>04MIN 30SEC</td>
</tr>
</tbody>
</table>

MEDIARY OUTPUT COUNT: 10 WORDS
END OF MOCK-DONALD SYSPIT
XIII. LEVELS OF VOCABULARY AND USE OF SYMBOLS

The following list summarizes LT's use of symbols for routines and data.

<table>
<thead>
<tr>
<th>Symbol(s)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AO, BO, ----, GO</td>
<td>Free variables</td>
</tr>
<tr>
<td>PO, QO, ----, TO</td>
<td>Bound variables</td>
</tr>
<tr>
<td>IO, VO, -0, *0, =0, =1</td>
<td>Connectives</td>
</tr>
<tr>
<td>Other zeroth symbols</td>
<td>Character symbols (except H0, J0, W0)</td>
</tr>
<tr>
<td>/1, /2, ----, /14</td>
<td>Substitute character symbols</td>
</tr>
<tr>
<td>N1, N2, ----, N10</td>
<td>Integer data terms 1, 2, ----, 0</td>
</tr>
<tr>
<td>K1-K99</td>
<td>Constants and parameters</td>
</tr>
<tr>
<td>L1-L49</td>
<td>Control and working lists</td>
</tr>
<tr>
<td>T1-T39</td>
<td>Text lists</td>
</tr>
<tr>
<td>Q1-Q49</td>
<td>Attributes of terms and expressions</td>
</tr>
<tr>
<td>P1-P99</td>
<td>Routines to operate on terms and expressions</td>
</tr>
<tr>
<td>M1-M199</td>
<td>Logic theorist routines</td>
</tr>
<tr>
<td>M1---M9</td>
<td>Executives, setup</td>
</tr>
<tr>
<td>M10--M19</td>
<td>Methods</td>
</tr>
<tr>
<td>M40--M49</td>
<td>Utility measures</td>
</tr>
<tr>
<td>M50--M59</td>
<td>Information recorders</td>
</tr>
<tr>
<td>M60--M69</td>
<td>Information retrievers</td>
</tr>
<tr>
<td>M70--M89</td>
<td>Input-output routines</td>
</tr>
<tr>
<td>M90--M99</td>
<td>Limit testers</td>
</tr>
<tr>
<td>M100-M119</td>
<td>Match processes</td>
</tr>
<tr>
<td>X1-X49</td>
<td>Run housekeeping and executives</td>
</tr>
<tr>
<td>X1---X9</td>
<td>Executives</td>
</tr>
<tr>
<td>X10--X19</td>
<td>Debugging routines</td>
</tr>
<tr>
<td>X20--X29</td>
<td>Debugging lists</td>
</tr>
</tbody>
</table>
LIST OF IPL-V BASIC PROCESSES

* Indicates processes which set H5

General Processes
J0 No operation
J1 Execute (0) after restoring H0
J2 TEST (0) * (1)
J3 Set H5
J4 Set H5+
J5 Reverse sense of H5
J6 Reverse (0) and (1)
J7 Halt, proceed on GO
J8 Restore H0
J9 ERASE cell (0)

Description Processes
J10 FIND value of attribute (0) of (1)
J11 Assign (1) as value of attribute (0) of (2)
J12 Add (1) at front of value list of attribute (0) of (2)
J13 Add (1) at end of value list of attribute (0) of (2)
J14 ERASE attribute (0) of (1)
J15 ERASE all attributes of (0)
J16 FIND attribute of (0) randomly

Generator Housekeeping Processes
J17 Gen set up; context (0) for subprocess (1)
J18 Execute subprocess of Gen
J19 Gen clean up

Working Storage Processes
J20 MOVE (0)-(n) into WO-Wn
J3n Restore WO-Wn
J4n Preserve WO-Wn
J5n Preserve WO-Wn; MOVE (0)-(n) into WO-Wn

List Processes
J60 LOCATE next symbol after cell (0)
J61 LOCATE last symbol on list (0)
J62 LOCATE (0) on list (1) (1st occurrence)
J63 INSERT before symbol in cell (1)
J64 INSERT after symbol in cell (1)
J65 INSERT (0) at end of list (1)
J66 INSERT (0) at end if not on list (1)
J67 Replace (1) by (0) on list (2) (1st occur.)
J68 DELETE symbol in cell (0)
J69 DELETE (0) from list (1) (1st occurrence)
J70 DELETE last symbol from list (0)
J71 ERASE list (0)
J72 ERASE list structure (0)
J73 COPY list (0)
J74 COPY list structure (0)
J75 Divide list after location (0); name of remainder is output (0)
J76 INSERT list (0) after (1), locate last symbol
J77 TEST if (0) is on list (1)
J78 TEST if list (0) is not empty
J79 TEST if cell (0) is not empty
J8n FIND the nth symbol on list (0)
J9n Create list of n symbols, (n-1) to (0)
J100 Gen symbols on list (1) for (0)
J101 Gen cells of list structure (1) for (0)
J102 Gen cells of tree (1) for (0)
J103 Gen cells of block (1) for (0)
J104

Auxiliary Storage Processes
J105 MOVE list structure (0) in from auxiliary
J106 FILE list structure (0) in fast-auxiliary
J107 FILE list structure (0) in slow-auxiliary
J108 TEST if list structure (0) is on auxiliary
J109 Compact auxiliary data storage system (0)

Arithmetic Processes
J110 X = (0), leave (0)
J111 (1) = (2) - (0), leave (0)
J112 (1) = (2), (0), leave (0)
J113 (1) / (2) = (0), leave (0)
J114 TEST if (0) = (1)
J115 TEST if (0) > (1)
J116 TEST if (0) < (1)
J117 TEST if (0) = (0)
J118 TEST if (0) > 0
J119 TEST if (0) < 0
J120 COPY (0)
J121 = (0) identical to (1), leave (0)
J122 Take absolute value of (0), leave (0)
J123 Take negative of (0), leave (0)
J124 Clear (0), leave (0)
J125 Tally 1 in (0), leave (0)
J126 Count list (0)
J127 TEST if data type (0) = data type (1)
J128 Translate (0) to be data type of (1)
J129 Produce random number between 0 and (0)

Data Prefix Processes
J130 TEST if (0) is regional symbol
J131 TEST if (0) names data term
J132 TEST if (0) is local symbol
J133 TEST if list (0) has been marked processed
J134 TEST if (0) is internal symbol
J135
J136 Make (0) local, leave (0)
J137 Mark list (0) processed, leave (0)
J138 Make (0) internal, leave (0)
J139

Read and Write Processes
J140 Read list structure
J141 Read symbol from console
J142 Write list structure (0)
J143 Read tape (0)
J144 Skip to next tape file
J145 Write end-of-file
J146 Write end-of-set

Monitor System
J147 Mark routine (0) to trace
J148 Mark routine (0) to propagate trace
J149 Mark routine (0) not to trace

Print Processes
J150 Print list structure (0)
J151 Print list (0)
J152 Print symbol (0)
J153 Print data term (0) w/ name or type
J154 Clear print line
J155 Print line
J156 Enter symbol (0) left-justified
J157 Enter data term (0) left-justified
J158 Enter symbol (0) right-justified
J159 Enter data term (0) right-justified
J160 Tab to column (0)
J161 Increment column by (0)
J162 Enter (0) according to format W43
J163 J164

In-process Loading
J165 Load routines and data
J166 Save on unit (0) for restart
J167 Skip list structure
J168 J169

Error Trap
J170 Trap on (0)

Block Handling Processes
J171 Return unused regionals to H2
J172 Make block (0) into a list
J173 Read into block (0)
J174 Write block (0)
J175 FIND region control word of regional symbol (0)
J176 Space (0) blocks on unit 1W19
J177 J178

Line Read Processes
J180 Read line
J181 Input line symbol
J182 Input line data term (0)
J183 Set (0) to next blank
J184 Set (0) to next non-blank
J185 Set (1) to next occurrence of character (0)
J186 Input line character
J187 J188

J189 Transfer field to line (0)

Partial Word Processes
J190 Input P of cell
J191 Input Q of cell (0)
J192 Input SYMB of cell (0)
J193 Input LINK of cell (0)
J194 Set (1) to be P of cell (0)
J195 Set (1) to be Q of cell (0)
J196 Set (1) to be SYMB of cell (0)
J197 Set (1) to be LINK of cell (0)
J198 J199

Miscellaneous Processes
J200 LOCATE (0)th symbol on list (1)
J201 ERASE routine (0)
J202 Print post mortem and continue
IPL INSTRUCTION: PQ SYMB LINK

P is operation code
P = 0 Execute S
P = 1 Input S (after preserving HO)
P = 2 Output to S (then restore HO)
P = 3 Restore (pop up) S
P = 4 Preserve (push down) S
P = 5 Replace (O) by S
P = 6 Copy (O) in S
P = 7 Branch to S if H5-

Q is designation code
Q = 0 S = SYMB
Q = 1 S = symbol in cell named SYMB
Q = 2 S = symbol in cell named in cell named SYMB
Q = 3 S = SYMB; start selective trace
Q = 4 S = SYMB; continue selective trace
Q = 5 Machine language routine
Q = 6 Routine in fast-aux. storage
Q = 7 Routine in slow-aux. storage
SYMB is symbol operated on by Q
LINK is address of next instruction
(0 for end of routine)

SYSTEM STORAGE CELLS
H0 Communication cell
H1 Current instruction address cell
H2 Available space list
H3 Tally of interpretation cycles
H4 Current auxiliary routine cell
H5 Test cell
W0-W9 Common working storage
W10 Random number control cell
W11 Integer division remainder
W12 Monitor start cell (Q = 3)
W13 Monitor end cell (Q = 3)
W14 External interrupt cell
W15 Post mortem routine cell
W16 Input mode cell
W17 Output mode cell
W18 Read unit cell
W19 Write unit cell
W20 Print unit cell
W21 Print column cell
W22 Print spacing cell
W23 Post mortem list cell
W24 Print line cell
W25 Print entry column cell
W26 Error trap cell
W27 Trap address cell
W28 Trap symbol cell
W29 Monitor point address cell
W30 Field length cell
W31 Trace mode cell
W32 Reserved available space cell
W33 Cycle count for trap cell
W34 Current available space cell
W35 Slow-aux. obsolete structure cell
W36 Used slow-auxiliary space cell
W37 Slow-auxiliary storage density cell
W38 Slow-auxiliary storage compacting routine cell
W39 Fast-aux. obsolete structure cell
W40 Used fast-auxiliary space cell
W41 Fast-auxiliary storage density cell
W42 Fast-auxiliary storage compacting routine cell
W43 Format cell

IPL DATA: PQ SYMB LINK

Q = 0 Standard list cell:
   P is irrelevant
   SYMB is symbol
   LINK is address of next list cell
   (0 for end of list)
Q = 1 Data term:
   ±PQ SYMB LINK
   Decimal integer 1 dddd dddd
   Floating point 11 dddd d d ee
   Alphanumeric 21 aaaa
   Octal 31 dddd dddd

TYPE CARDS
0 (blank) Routines and data
1 Comments
2 Region definition
   NAME = Regional symbol
   SYMB = Origin (if given)
   LINK = Size
3 Block reservation
   NAME = Block control word (if given)
   SYMB = Origin (if given)
   LINK = Size
   Q = 0 Reserve regional symbols
   Q = 1 Reserve print line
   Q = 2 Reserve block
   Q = 3 Reserve auxiliary buffer
   Q = 4 Specify available space
4 Listing cards
5 Main storage header
6 Fast-auxiliary storage header
7 Slow-auxiliary storage header
8 Editing header; Inhibits loading
   NAME = Name of storage block
   P = Input mode
   P = 0 IPL standard
       P = 1 IPL compressed
       P = 2 IPL binary
       P = 3 Machine code
       P = 4 Restart mode
   Q = Type of input
       Q = 0 Routines; internals
       Q = 1 Data; internals
       Q = 2 Routines; internals
       Q = 3 Data; internals symbolic;
           reset internal symbol table
       Q = 4 Routines; internals
           symbolic; reset internal
           symbol table
       Q = 5 Data; internals
           absolute
       Q = 6 Data; internals
           absolute
   SYMB = Alternate input unit
0 (blank) = controlling unit
1-10 = Internal tapes
   Regional SYMB names first
   routine (terminate loading)
LINK = Output mode: of form bbbcd
   b = Output unit: blank = unit
       1W1; 1-10 = unit 1-10
   c = 0 (blank) if assembly
       listing = 1 or any character if no
       assembly listing
   d = 0 (blank) if no output
       = 1 IPL compressed output
       = 2 IPL binary output
       = 3 Machine code output
       = 9 IPL standard output
9 First card
   SYMB = Controlling unit (0 or blank
   * normal input unit)
XIV. COMPLETE VOCABULARY LISTING

A complete vocabulary listing, as shown in this section, was kept up to date during conversion of LT into its present form.

The vocabulary is intended to serve as an extension of the List of Basic Processes in the IPL-V system.
<table>
<thead>
<tr>
<th>A0</th>
<th>FREE VARIABLE -A-</th>
<th>A000V000</th>
</tr>
</thead>
<tbody>
<tr>
<td>B0</td>
<td>FREE VARIABLE -B-</td>
<td>B000V000</td>
</tr>
<tr>
<td>C0</td>
<td>FREE VARIABLE -C-</td>
<td>C000V000</td>
</tr>
<tr>
<td>D0</td>
<td>FREE VARIABLE -D-</td>
<td>D000V000</td>
</tr>
<tr>
<td>E0</td>
<td>FREE VARIABLE -E-</td>
<td>E000V000</td>
</tr>
<tr>
<td>F0</td>
<td>FREE VARIABLE -F-</td>
<td>F000V000</td>
</tr>
<tr>
<td>G0</td>
<td>FREE VARIABLE -G-</td>
<td>G000V000</td>
</tr>
<tr>
<td>I0</td>
<td>LOGICAL CONNECTIVE -IMPLIES-</td>
<td>I000V000</td>
</tr>
<tr>
<td>K0</td>
<td>Symbol for character K</td>
<td>K000V000</td>
</tr>
<tr>
<td>K1</td>
<td>Holds -OR-</td>
<td>K001V000</td>
</tr>
<tr>
<td>K2</td>
<td>Holds -NOT-</td>
<td>K002V000</td>
</tr>
<tr>
<td>K3</td>
<td>Holds -AND-</td>
<td>K003V000</td>
</tr>
<tr>
<td>K4</td>
<td>Holds -PROVEN EQUIVALENCE-</td>
<td>K004V000</td>
</tr>
<tr>
<td>K5</td>
<td>Holds -DEFINITIONAL EQUIVALENCE</td>
<td>K005V000</td>
</tr>
<tr>
<td>K6</td>
<td>Holds -IMPLIES-</td>
<td>K006V000</td>
</tr>
<tr>
<td>K7</td>
<td>Holds DELIMITER SYMBOL FOR DELIMITABLE CHARACTERS</td>
<td>K007V000</td>
</tr>
<tr>
<td>K9</td>
<td>Previous subproblem number (data term)</td>
<td>K010V000</td>
</tr>
<tr>
<td>K.1</td>
<td>Substitution count</td>
<td>K011V000</td>
</tr>
<tr>
<td>K.2</td>
<td>Effort base (and total)</td>
<td>K012V000</td>
</tr>
<tr>
<td>K.20</td>
<td>Limit on no. of subproblems</td>
<td>K020V000</td>
</tr>
<tr>
<td>K.21</td>
<td>Limit on no. of substitutions</td>
<td>K021V000</td>
</tr>
<tr>
<td>K.22</td>
<td>Limit on effort</td>
<td>K022V000</td>
</tr>
<tr>
<td>K30</td>
<td>Holds ro if proved theorems are to be remembered</td>
<td>K030V000</td>
</tr>
<tr>
<td>K31</td>
<td>Holds yo if rejected subproblems are to be printed</td>
<td>K031V000</td>
</tr>
<tr>
<td>K41</td>
<td>D.T. COLUMN FOR PRINTING METHODS IN PROOF SEQUENCES</td>
<td>K041V000</td>
</tr>
<tr>
<td>K42</td>
<td>D.T. COLUMN FOR PRINTING NAMES IN PROOF SEQUENCES</td>
<td>K042V000</td>
</tr>
<tr>
<td>K43</td>
<td>D.T. COLUMN FOR PRINTING EXPRESSIONS IN PROOF SEQUENCES</td>
<td>K043V000</td>
</tr>
<tr>
<td>K44</td>
<td>D.T. COLUMN FOR PRINTING 'LIMIT'</td>
<td>K044V000</td>
</tr>
<tr>
<td>K45</td>
<td>D.T. COLUMN FOR PRINTING 'ACTUAL'</td>
<td>K045V000</td>
</tr>
<tr>
<td>K46</td>
<td>D.T. COLUMN FOR PRINTING 'REJECTED SUBPROBLEM'</td>
<td>K046V000</td>
</tr>
<tr>
<td>K47</td>
<td>D.T. COLUMN FOR PRINTING NAME OF NEW SUBPROBLEM</td>
<td>K047V000</td>
</tr>
<tr>
<td>K48</td>
<td>D.T. COLUMN FOR PRINTING 'THEOREM, METHOD' OF NEW PROBLEMS</td>
<td>K048V000</td>
</tr>
<tr>
<td>K51</td>
<td>Data term '1'</td>
<td>K051V000</td>
</tr>
<tr>
<td>K52</td>
<td>Data term '11'</td>
<td>K052V000</td>
</tr>
<tr>
<td>K53</td>
<td>Data term '111'</td>
<td>K053V000</td>
</tr>
<tr>
<td>K54</td>
<td>Data term '1111'</td>
<td>K054V000</td>
</tr>
<tr>
<td>L0</td>
<td>Symbol for character L</td>
<td>L000V000</td>
</tr>
<tr>
<td>L1</td>
<td>List of true expressions (axioms, definitions, theorems)</td>
<td>L021V000</td>
</tr>
<tr>
<td>L2</td>
<td>List of free variables</td>
<td>L022V000</td>
</tr>
<tr>
<td>L3</td>
<td>List of unproved expressions (problems)</td>
<td>L033V000</td>
</tr>
<tr>
<td>L4</td>
<td>List structure map of all true expressions</td>
<td>L044V000</td>
</tr>
<tr>
<td>L5</td>
<td>List structure map of true expressions</td>
<td>L055V000</td>
</tr>
<tr>
<td>L6</td>
<td>List of special methods for original problems</td>
<td>L066V000</td>
</tr>
<tr>
<td>L7</td>
<td>List of regular methods for all problems</td>
<td>L077V000</td>
</tr>
<tr>
<td>L8</td>
<td>Description list table of delimitable characters</td>
<td>L088V000</td>
</tr>
<tr>
<td>L9</td>
<td>Description list table of character symbols for reading text</td>
<td>L099V000</td>
</tr>
<tr>
<td>L10</td>
<td>List of untried problems</td>
<td>L010V000</td>
</tr>
<tr>
<td>L11</td>
<td>List of found problems</td>
<td>L011V000</td>
</tr>
<tr>
<td>M0</td>
<td>Symbol for character M</td>
<td>M000V000</td>
</tr>
<tr>
<td>M1</td>
<td>Single problem executive for problem (0)</td>
<td>M001V000</td>
</tr>
<tr>
<td>M2</td>
<td>Multiple problem executive for list L3</td>
<td>M002V000</td>
</tr>
</tbody>
</table>
M3 SET-UP FOR NEW PROBLEM.
M7 APPLY METHODS (1) TO PROBLEM (0)+ ERASE (1) WHEN THRU.
M8 CREATE A LIST OF METHODS FOR PROBLEM (0).
M11 DETACHMENT METHOD FOR PROBLEM (0). SETS H5+ AND OUTPUTS SUCCESSFUL SUBPROBLEM IF SOLUTION IS FOUND. SETS H5- IF NO SOLUTION. ADDS GOOD NEW SUBPROBLEMS TO UNTRIED LIST.
M12 SUBSTITUTION METHOD FOR PROBLEM (0). SETS H5+ AND OUTPUTS SUCCESSFUL SUBPROBLEM IF SOLUTION IS FOUND. SETS H5- IF NO OUTPUT.
M13 REPLACEMENT METHOD FOR SUBPROBLEM (0). SETS H5+ AND OUTPUTS SUCCESSFUL SUBPROBLEM IF SOLUTION IS FOUND. SETS H5- IF NO SOLUTION. ADDS GOOD NEW SUBPROBLEMS TO UNTRIED LIST.
M14 FORWARD CHAINING METHOD FOR PROBLEM (0). SETS H5+ AND OUTPUTS SUCCESSFUL SUBPROBLEM IF SOLUTION IS FOUND. SETS H5- IF NO SOLUTION. ADDS GOOD NEW SUBPROBLEMS TO UNTRIED LIST.
M15 BACKWARD CHAINING METHOD FOR PROBLEM (0). SETS H5+ AND OUTPUTS SUCCESSFUL SUBPROBLEM IF SOLUTION IS FOUND. SETS H5- IF NO SOLUTION. ADDS GOOD NEW SUBPROBLEMS TO UNTRIED LIST.
M16 SUBLEVEL REPLACEMENT METHOD FOR PROBLEM (0). SETS H5+ AND OUTPUTS SUCCESSFUL SUBPROBLEM IF SOLUTION IS FOUND. SETS H5- IF NO SOLUTION. ADDS GOOD NEW SUBPROBLEMS TO UNTRIED LIST. M16 TRIES FOR A SUBPROBLEM AT EACH SUBLEVEL OF PROBLEM (0).
M17 SUBLEVEL REPLACEMENT METHOD FOR PROBLEM (0). SETS H5+ AND OUTPUTS SUCCESSFUL SUBPROBLEM IF SOLUTION IS FOUND. SETS H5- IF NO SOLUTION. ADDS GOOD NEW SUBPROBLEMS TO UNTRIED LIST. M17 TRIES FOR ONLY ONE SUBPROBLEM BY TRYING REPLACEMENT ON ALL SUBLEVELS.
M19 FINISH BUILDING NEW SUBPROBLEM TEX (3) FROM (2) VIA THEOREM (1) BY METHOD (0). H5- MEANS NEW TEX WAS ERASED DUE TO LOW UTILITY. H5+ MEANS OUTPUT TEX (0) HAS SATISFACTORY UTILITY.
M40 TEST MATCH OF TOTAL EXPRESSIONS (0)+ (1) WITHOUT SUBSTITUTION.
M41 TEST MATCH OF SEGMENTS (0)+ (1) WITHOUT SUBSTITUTION.
M42 ADD PROBLEM (0) TO FOUND PROBLEMS LIST IF CAN.
M50 ADD TRUE EXPRESSION (0) TO TRUE EXPRESSIONS LIST AND MAP.
M51 ADD PROBLEM (0) TO UNTRIED PROBLEMS LIST.
M54 ADD EXPRESSION (0) TO MAP OF TRUE EXPRESSIONS.
M60 FIND AND REMOVE NEXT UNTRIED SUBPROBLEM ON UNTRIED LIST.
M62 CREATE A LIST OF FEASIBLE MATCHES FOR SEGMENT (0) FROM MAP (1).
M63 CREATE A LIST OF FEASIBLE MATCHES FOR TOTAL EXPRESSION (1) FROM MAP (0).
M70 PRINT EXPRESSION (0).
M71 PRINT PROOF SEQUENCE WITH SUCCESSFUL SUBPROBLEM (0).
M72 PRINT FAILURE TO FIND PROOF.
M73 ENTER SEGMENT (0).
M74 ENTER TOTAL EXPRESSION (0).
M75 PRINT NEW SUBPROBLEM (0).
M76 ENTER LIST OF DATA TERMS (0).
M77 PRINT CURRENT STATUS OF LIMITS.
M78 PRINT TO PROVE PROBLEM (0).
M79 ENTER NAME OF (0). USE EXTERNAL NAME IF CAN.
M80 PRINT PROOF LINE FOR METHOD (0) AND TEX (1).
M81 PRINT REJECTED PROBLEM.
M82 PRINT REMEMBER PROVED THEOREM.
M83 PRINT LIST FORM EXPRESSION (0).
M84 READ NEXT LOGIC EXPRESSION FROM UNIT 1W18.

M90 TEST IF A LIMIT HAS BEEN REACHED.
M100 MAKE FREE VARIABLES OF TOTAL EXPRESSIONS (0), (1) DISJOINT.
SUBSTITUTES NEW VARIABLES IN (1) AS REQUIRED.
M111 MATCH SEGMENTS (0), (1) WITH SUBSTITUTION AS REQUIRED.
H5+ MEANS OUTPUT (0) IS A SUBSTITUTION LIST OF PAIRS.
1ST IS A FREE VARIABLE, 2ND IS ITS SUBSTITUTE.
M112 EXPAND SUBSTITUTION LIST (0).
REPLACE EXPRESSIONS WITH COMPLETELY SUBSTITUTED LOCAL COPIES.
M113 MATCH SEGMENTS (0), (1) WITH SUBSTITUTION AS REQUIRED.
H5+ MEANS OUTPUT (0) IS EXPANDED SUBSTITUTION LIST.
M114 MATCH SEGMENTS (0), (1) WITH SUBSTITUTION AS REQUIRED.
NO OUTPUT, H5- MEANS THE MATCH FAILED.
M115 SUBSTITUTE IN SEGMENT (0) FROM SUBSTITUTION LIST (1).
M116 CREATE LIST OF FREE VARIABLES IN TOTAL EXPRESSION (0).
H5- MEANS NO OUTPUT.
M117 CREATE LIST OF BOUND VARIABLES IN TOTAL EXPRESSION (0).
H5- MEANS NO OUTPUT.

NO SYMBOL FOR CHARACTER N.
N1 INTEGER DATA TERM = 1
N2 INTEGER DATA TERM = 2
N3 INTEGER DATA TERM = 3
N4 INTEGER DATA TERM = 4
N5 INTEGER DATA TERM = 5
N6 INTEGER DATA TERM = 6
N7 INTEGER DATA TERM = 7
N8 INTEGER DATA TERM = 8
N9 INTEGER DATA TERM = 9
N10 INTEGER DATA TERM = 0
O0 SYMBOL FOR CHARACTER O.

P0 VARIABLE TERM -P-
P1 TEST IF (0) IS A BOUND VARIABLE.
P2 CLEAR DESCRIPTIONS OF EXPRESSION (0).
P3 GO THRU NOTS OF SEGMENT (0). LEAVE FIRST UNNOTED SEGMENT.
H5- MEANS NO OUTPUT DUE TO FAULTY EXPRESSION.
P4 TEST IF MAIN CONNECTIVE OF EXPRESSION (0) IS -IMPLIES-.
P5 TEST IF CONNECTIVE (0) IS NON-UNARY.
P6 TEST IF (0) IS A CONNECTIVE.
P7 TEST IF (0) IS VARIABLE TERM.
P8 TEST IF (0) IS FREE VARIABLE.
P9 FIND MAIN EXPRESSION OF TOTAL EXPRESSION (0).
P10 FIND LEFT SEGMENT OF TOTAL EXPRESSION (0).
P14  FIND RIGHT SEGMENT OF TOTAL EXPRESSION (0).  
P15  TEST IF TOTAL EXPRESSION (0) IS IN TREE FORM.  
P16  FIND MAIN CONNECTIVE OF TOTAL EXPRESSION (0).  
P17  COPY SEGMENT (0).  IF (0) IS A VARIABLE, OUTPUT THE INPUT.  
P18  TEST IF (0) IS A CHARACTER SYMBOL.  
P19  GET APPROPRIATE CHARACTER SYMBOL FOR (0).  
P20  MAKE LEFT SUBEXPRESSION OF EXPRESSION (0) INTO EXPRESSION,  
     H5- MEANS NO OUTPUT DUE TO FAULTY EXPRESSION.  
P21  MAKE RIGHT SUBEXPRESSION OF EXPRESSION (0) INTO EXPRESSION,  
     H5- MEANS NO OUTPUT DUE TO FAULTY EXPRESSION.  
P22  CREATE NEW PROBLEM WITH SEGMENT (0) AS LEFT SUBEXPRESSION,  
     SEGMENT (1) AS RIGHT SUBEXPRESSION; IMPLIES AS CONNECTIVE.  
P23  ERASE MADE EXPRESSION (0).  
P24  MAKE SEGMENT (0) INTO A NEW TOTAL EXPRESSION.  
P25  COPY TEX (0) FOR SUBSTITUTION.  
P26  GENERATE LOCATIONS OF NON-VARIABLE SEGMENTS FROM  
     TOTAL EXPRESSION (1) AT THE LEVEL OF DATA TERM(2)  
     FOR PROCESS (0).  
P27  REPLACE BOUND VARIABLES BY FREE VARIABLES IN TEX (0).  
P28  GENERATE LOCATIONS OF FREE VARIABLES IN SEGMENT (1) FOR (0).  
P29  GENERATE LOCATIONS OF BOUND VARIABLES IN SEGMENT (1) FOR (0).  
P30  CREATE LIST OF FREE VARIABLES IN EXPRESSION (0).  
P31  CREATE LIST OF BOUND VARIABLES IN EXPRESSION (0).  
P50  CONVERT LOGIC EXPRESSION (0) TO INTERNAL TREE FORM IF IN  
     EXTERNAL FORM; NO OUTPUT. H5- MEANS NO CONVERSION.  
     ENTIRE EXPRESSION MUST BE PARENTHESIZED.  
P51  REPLACE ALL DELIMITED SYMBOLS IN EXPRESSION (0) IF  
     (0) IS IN LIST FORM.  
P52  CREATE A TREE FORM MAIN SEGMENT FROM LIST FORM EXPRESSION (0).  
     H5- MEANS NO OUTPUT DUE TO A FAULTY INPUT EXPRESSION.  
P55  LOCATE LIST FOLLOWING DATA TERM (0) ON LIST (1).  
     H5- MEANS OUTPUT (0) IS CELL HOLDING SUBLIST.  
     H5- MEANS OUTPUT (0) IS CELL AFTER WHICH TO INSERT.  
Q0  VARIABLE TERM -Q-  
Q1  FIND CONNECTIVE OF SEGMENT (0).  
Q2  FIND NO. OF LEVELS OF EXPRESSION (0).  
Q3  FIND NO. OF DISTINCT VARIABLES OF EXPRESSION (0).  
Q4  FIND NO. OF VARIABLE PLACES OF EXPRESSION (0).  
Q5  ATTRIBUTE--VARIABLE TERM.  
Q6  ATTRIBUTE--FREE VARIABLE.  
Q7  ATTRIBUTE--EXTERNAL NAME.  
Q8  FIND PROBLEM NO. OF EXPRESSION (0).  
Q9  ATTRIBUTE--BOUND VARIABLE.  
Q10  FIND PROBLEM EXPRESSION (0) DERIVED FROM.  
Q11  FIND METHOD OF DERIVATION FOR EXPRESSION (0).  
Q12  FIND THEOREM USED TO DERIVE PROBLEM (0).  
Q13  FIND PROVING THEOREM FOR PROBLEM (0).  
Q14  FIND TYPE OF CONNECTIVE (0).  
Q15  ATTRIBUTE--INTERNAL FORM.  
Q16  FIND EXTERNAL NAME OF (0) IN TABLE T10.  
Q17  FIND CURRENT LEVEL OF SUBSEGMENT REPLACEMENT OF PROBLEM (0).
Q18  FIND SUFFIX OF EXPRESSION (0).
Q19  FIND APPROPRIATE CHARACTER SYMBOL FOR (0).
R0  VARIABLE TERM -R-
S0  VARIABLE TERM -S-
T0  VARIABLE TERM -T-
T1  TEXT LIST 'GIVEN'
T2  TEXT LIST 'PROOF FOUND'
T3  TEXT LIST 'SUBSTITUTION'
T4  TEXT LIST 'Q.E.D.'
T5  TEXT LIST OF 5 BLANK CHARACTERS.
T6  TEXT LIST 'NO PROOF FOUND'
T7  TEXT LIST 'EFFORT'
T8  TEXT LIST 'SUBPROBLEMS'
T9  TEXT LIST 'SUBSTITUTIONS'
T10 DESCRIPTION LIST TABLE OF NAMES.
T12 TEXT LIST 'DETACHMENT'
T13 TEXT LIST 'REPLACEMENT'
T14 TEXT LIST 'FORWARD CHAINING'
T15 TEXT LIST 'BACKWARD CHAINING'
T16 TEXT LIST 'SUBLEVEL REPLACEMENT'
T18 TEXT LIST 'REJECTED PROBLEM'
T20 TEXT LIST 'ACTUAL'
T21 TEXT LIST 'LIMIT'
T22 TEXT LIST 'TO PROVE'
T23 TEXT LIST 'REMEMBER PROVED THEOREM'
T24 TEXT LIST 'BAD EXPRESSION'
U0  SYMBOL FOR CHARACTER U.
V0  LOGICAL CONNECTIVE -OR-
X0  SYMBOL FOR CHARACTER X.
X1  RUN EXECUTIVE
X9  SAVE AND CONTINUE EXECUTIVE.
X21 LIST OF ROUTINES TO BE MARKED TO TRACE.
X22 LIST OF ROUTINES TO BE MARKED TO PROPAGATE TRACE.
X23 DESCRIPTION LIST OF TRAP ACTIONS. (ATTRIBUTE/VALUE PAIRS)
Y0  SYMBOL FOR CHARACTER Y.
Z0  SYMBOL FOR CHARACTER Z.
=0  LOGICAL CONNECTIVE -PROVEN EQUIVALENCE-
=1  LOGICAL CONNECTIVE -DEFINITIONAL EQUIVALENCE-
-0  LOGICAL CONNECTIVE -NOT-
*0  LOGICAL CONNECTIVE -AND-
00  SYMBOL FOR QUOTE MARK.
00  SYMBOL FOR PERIOD.
10  SYMBOL FOR RIGHT PAREN.
/0  SYMBOL FOR SLASH.
/1  SYMBOL FOR DIGIT 1.
/2  SYMBOL FOR DIGIT 2.
/3  SYMBOL FOR DIGIT 3.
/4  SYMBOL FOR DIGIT 4.
/5  SYMBOL FOR DIGIT 5.
/6  SYMBOL FOR DIGIT 6.
/7  SYMBOL FOR DIGIT 7.
/8 SYMBOL FOR DIGIT 8.
/9 SYMBOL FOR DIGIT 9.
/10 SYMBOL FOR DIGIT 0.
/11 SYMBOL FOR CHARACTER H.
/12 SYMBOL FOR CHARACTER J.
/13 SYMBOL FOR CHARACTER W.
/14 DUMMY CHARACTER SYMBOL WITH EXTERNAL NAME '/UGH/'
/16 DUMMY EXPRESSION TO SUPPLY TEXT 'THE DEFINITIONS'
+0 SYMBOL FOR COMMA.
+0 SYMBOL FOR LEFT PAREN.
+0 SYMBOL FOR PLUS SIGN.
$0 SYMBOL FOR DOLLAR SIGN.
XV. COMPLETE PROGRAM LISTING

The program listing has been carefully and extensively documented in the comment fields to enable students to work directly from it.

```
JO6 8160,LTNEW1,EAS826,5MIN,0,099,C
ASSIGN A6=SYSAR2
ASSIGN B6=SYSAR3
IPL
LOGIC THEORIST (IPL-V)

2 A 10 0000 000
2 B 10 0000 010
2 C 10 0000 020
2 D 10 0000 030
2 E 10 0000 040
2 F 10 0000 050
2 G 10 0000 060
2 H 10 0000 070
2 K 100 0000 080
2 L 50 0000 090
2 M 200 0000 100
2 N 50 0000 110
2 O 10 0000 120
2 P 100 0000 130
2 Q 50 0000 140
2 R 10 0000 150
2 S 10 0000 160
2 T 40 0000 170
2 U 10 0000 180
2 V 10 0000 190
2 X 50 0000 200
2 Y 10 0000 210
2 Z 50 0000 220
2 - 10 0000 230
2 * 600 0000 240
2 = 10 0000 260
2 + 10 0000 270
2 / 20 0000 280
2 + 10 0000 290
2 ( 10 0000 300
2 ) 10 0000 310
2 ) 10 0000 320
2 ) 10 0000 330
```
PROGRAM HEADER
M1 SINGLE PROBLEM EXECUTIVE FOR
PROBLEM (0). H5+ IF SUCCEEDS.

TEST UTILITY.
IF NO GOOD, QUIT.
PRINT 'TO PROVE' PROBLEM 1WO.
TRY SUBSTITUTION.
IF WORKED, PRINT PROOF.
CREATE LIST OF METHODS FOR PROB.
APPLY METHODS.
IF PROOF FOUND, PRINT IT.
TEST IF ANY LIMITS EXCEEDED.
IF YES, QUIT.
FIND NEXT SUBPROBLEM.
IF NONE, QUIT.
IF ONE,
PRINT SUBPROBLEM, TRY METHODS.
PRINT PROOF FROM (0).
AND QUIT +.
PRINT FAILURE, QUIT -.
PRINT REJECTED PROBLEM
AND QUIT -.

M2 MULTIPLE PROBLEM EXECUTIVE.
GENERATE PROBLEMS FOR PROOF.
CONVERT TO INTERNAL (TREE) FORM.
IF FAILED, TAKE ERROR ACTION.

TEST IF REMEMBERING.
INPUT PROBLEM FOR M1.
TRY FOR PROOF TO BE REMEMBERED.
H5- MEANS NO PROOF FOUND.
PRINT 'REMEMBER PROVED THEOREM'.
ADD TO TRUE EXPRESSIONS LIST.
TRY FOR PROOF TO BE FORGOTTEN.
CLEAN UP W0, H5+ FOR GEN.

TOTAL EXECUTION TIME.

M2 10L3 109-100 1100 0 10R
M1 11W0 1109-101 M1 11W0
M3 40WC 60WC 1WO=PRCB M82 9-101 M88 9-102
M43 709-4 11W0 11W0 M81 9-5 30W0 J3
M78 11W0 M7 9-2
M12 70 M7 9-2
M90 70 9-3
M60 709-3 M90 9-1
M70 30W0 J4
M71 9-2
M11 9-3
M72 9-5
M81 9-4
M9 9-4
M50 709-103 11K30 10R
M50 709-102 M1 11W0
M50 9-102 M82 9-102
M50 9-102 M50 9-102
M001R000 M001R010
M001R020 M001R030
M001R040 M001R050
M001R060 M001R070
M001R080 M001R090
M001R100 M001R110
M001R120 M001R130
M001R140 M001R150
M001R160 M001R170
M001R180 M001R190
M001R200 M001R210
M001R220 M001R230
M001R240 M001R250
M001R260 M002R000
M002R010 M002R020
M002R030 M002R040
M002R050 M002R060
M002R070 M002R080
M002R090 M002R100
M002R110 M002R120
M002R130 M002R140
M002R150 M002R160
M002R170 M002R180
M3 SET UP FOR NEW PROBLEM.

CLEAR UNTRIED PROBLEMS LIST.

CLEAR FOUND PROBLEMS LIST.

CLEAR PREVIOUS PROBLEM NUMBER.

CLEAR SUBSTITUTIONS COUNT.

SET EFFORT BASE.

M7 APPLY METHODS (1) TO PROBLEM (0), ERASE LIST (1) WHEN THRU.

H5+ MEANS OUTPUT (0) IS SOLN.

H5- MEANS NO OUTPUT, NO SOLN.

IF NO PROOF, ERASE METHOD LIST.

ELSE SAVE PROOF AND ERASE LIST.

9-10C SUBPROCESS, APPLY METHOD (0) TO PROBLEM 1WO.

REVERSE H5 FOR GENERATOR.

M8 CREATE METHOD LIST FOR (0).

COPY REGULAR LIST.

TEST IF ORIGINAL PROBLEM.

IF NOT, NO SPECIAL METHODS.

IF YES,

COPY SPECIAL LIST.

INSERT AFTER HEAD OF REGULAR LIST.

SET UP OUTPUT AND QUIT.

M11 DETACHMENT METHOD FOR PROB (0).

ADD NEW SUBPROBLEMS TO UNTRIED LIST IF CAN.

H5+ MEANS OUTPUT (0) IS A SOLUTION.

H5- MEANS NO SOLUTION, NO OUTPT.

FIND IMPLIES MAPS.

FIND MAP OF RIGHT SIDES OF MAIN CONNECTIVE IMPLIES.
GET LIST OF FEASIBLE THEOREMS.
SAVE LIST FOR ERASURE.

GENERATE FEASIBLE THEOREMS.

CLEAN UP AND
REVERSE HS AFTER GENERATOR.

9-100 SUBPROCESS,
TRY PROOF WITH THEOREM (0).

MAKE FREE VARIABLES DISJOINT
FIND RIGHT SUBSEGMENT OF THM

MATCH, MAKE THM RIGHT LIKE PROB.

FIND THM LEFT SUBSEGMENT.

MAKE NEW TEX FROM COPY OF THMLFT.

SUBSTITUTE INTO NEWTEX FROM (1)
ERASE SUBSTL.
ASSIGN DERIVATION

COMPLETE NEW SUBPROBLEM
DESCRIPTION AND MEASURE UTILITY.
HS- MEANS NEW PROBLEM WAS ERASED.
TRY SUBSTITUTION, OUTPUT SOLN.

M12 SUBSTITUTION METHOD FOR
PROBLEM (0). HS+ MEANS
OUTPUT (0) IS A SOLUTION
HS- MEANS NO OUTPUT.

TALLY SUBSTITUTION COUNTER.

GET A LIST OF FEASIBLE THEOREMS.
SAVE THE LIST FOR ERASURE.

GENERATE THEOREMS.

ERASE LIST OF THEOREMS.
CLEAN-UP AND REVERSE H5.
9-100 SUBPROCESS, TRY PROBLEM MEX
1W0 WITH THEOREM (0).
MAKE FREE VARIABLES DISJOINT.
FIND MAIN EXPRESSION OF THEOREM.
TEST FOR MATCH.
IF H5-+ SET + FOR GENERATOR.
OTHERWISE ASSIGN PROOF
AND OUTPUT SUCCESSFUL PROBLEM 1W1
AS SOLUTION. QUIT WITH H5- FOR GEN.
M13 REPLACEMENT METHOD FOR PROB (0) M13
ADD NEW SUBPROBLEMS TO UNTRIED
LIST IF CAN.
H5+ MEANS OUTPUT (0) IS A
SOLUTION.
H5- MEANS NO SOLN, NO OUTPUT.
FIND DEFINITIONAL EQUIVALENCE MAPS.
FIND MAP OF LEFT SIDES OF DEFEQ.
GET LIST OF FEASIBLE DEFINITIONS.
SAVE LIST FOR ERASURE.
TRY LEFT SIDES.
GENERATE FEASIBLE DEFINITIONS.
ERASE FEASIBLES LIST.
REVERSE H5 FROM GENERATOR.
QUIT IF PROOF IN HC, OR TRY RIGHTS.
FIND MAP OF RIGHT SIDES OF DEFEQ.
GET LIST OF FEASIBLE DEFINITIONS.
SAVE LIST.
TRY RIGHT SIDES.
GENERATE FEASIBLE DEFINITIONS.
ERASE FEASIBLES LIST.
REVERSE H5, CLEAN UP AND QUIT.
TRY LEFT SIDE OF (0) W/1W0.
9-100 60W2 1W2=THM M012R150
11W1 M012R160
11W2 M012R170
M110 M012R180
J81 M012R190
70J4 M012R200
11W0 M012R210
M114 M012R220
70J4 M012R230
11W1 M012R240
11W2 M012R250
10G13 M012R260
J11 M012R270
11W1 M012R280
J3 M012R290
1 R M013R000
M13 REPLACEMENT METHOD FOR PROB (0) M13
ADD NEW SUBPROBLEMS TO UNTRIED
LIST IF CAN.
H5+ MEANS OUTPUT (0) IS A
SOLUTION.
H5- MEANS NO SOLN, NO OUTPUT.
FIND DEFINITIONAL EQUIVALENCE MAPS.
FIND MAP OF LEFT SIDES OF DEFEQ.
GET LIST OF FEASIBLE DEFINITIONS.
SAVE LIST FOR ERASURE.
TRY LEFT SIDES.
GENERATE FEASIBLE DEFINITIONS.
ERASE FEASIBLES LIST.
REVERSE H5 FROM GENERATOR.
QUIT IF PROOF IN HC, OR TRY RIGHTS.
FIND MAP OF RIGHT SIDES OF DEFEQ.
9-10 11W5 1W5=DEFMPS M013R050
J82 M013R100
70J36 M013R110
11W1 M013R120
J6 M013R130
M63 M013R140
60W6 1W6=FSBLS M013R150
109-100 M013R160
J100 M013R170
11W6 M013R180
J71 M013R190
J5 M013R200
11W6 M013R210
J71 M013R220
J5 M013R230
J36 M013R240
9-10 11W6 M013R250
M63 M013R260
11W1 M013R270
60W6 1W6=FSBLS M013R280
109-200 M013R290
J100 M013R300
11W6 M013R310
J71 M013R320
J5 M013R330
J36 M013R340
MAKE FREE VARIABLES DISJOINT.
FIND LEFT SUB SEG OF DEF.*

MATCH, MAKE DEF LIKE PROB IF CAN.

FIND RIGHT SIDE OF DEFINITION.
IF NONE,
ERASE SUBST. LIST, QUIT + FOR GEN.
TRY RIGHT SIDE OF (0) W/1WO.*

MAKE FREE VARIABLES DISJOINT.
FIND RIGHT SUB SEG OF DEF.

MATCH, MAKE DEF LIKE PROB IF CAN.

FIND LEFT SIDE OF DEFINITION.
IF NONE,
ERASE SUBST. LIST, QUIT + FOR GEN.
MAKE SUBPROB FROM SEGMENT (0)
WITH SUBSTITUTION LIST (1)
SUBSTITUTE INTO NEWTEX FROM SUBSTL.

ASSIGN DERIVATION, ADD FOUND LIST,
MEASURE UTILITY, ERASE IF NO GOOD.

TRY SUBSTITUTION, H5+ OUTPUT PROOF.

ERASE SUBSTITUTION LIST.

M14 FORWARD CHAINING METHOD FOR
PROBLEM (0), ADDS NEW SUBPROBS
TO UNTRIED LIST IF CAN.
H5+ MEANS OUTPUT (0) IS
SOLUTION.
H5- MEANS NO SOLUTION, NO OUTPUT
FIND MAIN CONNECTIVE OF PROB.
FIND APPROPRIATE THEOREM MAPS.

FIND MAP OF LEFT SIDES.

INPUT FAKE TEX.

GET FEASIBLE THEOREM LEFT SIDES.

TRY FEASIBLE THM LEFTS WITH
PROBLEM LEFT. GENERATE FSBLs.

ERASE LIST OF FEASIBLES.

FAKE TEX . . . 9-1 HOLDS MAX.
9-100 SUBPROCS, TRY LEFT SIDES.

MAKE FREE VARIABLES DISJOINT.

FIND LEFT SEGMENT OF THM TEX.

INPUT PROB LEFT
MATCH, OUTPUT LIST OF SUBSTITUTIONS
WILL MAKE THM LIKE PROB IF CAN.

FIND RIGHT SIDE OF PROB

FIND RIGHT SIDE OF THEOREM.

CREATE NEW TEX WITH COPIES.

THM LEFT, PROB RIGHT.

SUBSTITUTE INTO NEW TEX.

ERASE SUBSTL.

ASSIGN DERIVATION, ADD TO FOUND LIST
MEASURE UTILITY, ERASE IF NO, GOOD.

TRY SUBSTITUTION, HS? OUTPUT PROOF.

ERASE SUBSTITUTION LIST.

M15 BACKWARD CHAINING METHOD FOR
PROBLEM (0), ADDS NEW SUBPROBS
TO UNTRIED LIST IF CAN.
-136-

>5+ MEANS OUTPUT (O) IS A SOLUTION.
>5- MEANS NO SOLUTION, NO OUTPUT

FIND APPROPRIATE MAPS.
FIND MAP OF RIGHT SIDES.

GET FEASIBLE THEOREM RIGHT SIDES.
TRY FEASIBLE THM RIGHTS WITH PROBLEM RIGHT. GENERATE FSBL$.
ERASE LIST OF FEASIBLES.
FAKE TEX ... 9-1 HOLDS MEX. 9-0
9-100 SUBPROCESS, TRY RIGHT SIDES. 9-100

MAKE FREE VARIABLES DISJOINT.
FIND RIGHT SEGMENT OF THM TEX.

INPUT PROB RIGHT.
MATCH OUTPUT LIST OF SUBSTITUTIONS WILL MAKE THM LIKE PROB IF CAN.

FIND LEFT SIDE OF THEOREM.
FIND LEFT SIDE OF PROBLEM.
CREATE NEW TEX WITH COPIES.
PROB ON LEFT, THM ON RIGHT.
SUBSTITUTE INTO NEW TEX.
ERASE SUBSTL.

ASSIGN DERIVATION, ADD TO FOUND LIST MEASURE UTILITY, ERASE IF NO GOOD.

TRY SUBSTITUTION, H5+ OUTPUT PROOF.
ERASE SUBSTITUTION LIST.
M16 SUBLEVEL REPLACEMENT METHOD FOR M16 PROBLEM (0), ADD ALL NEW SUBPROBLEMS TO UNTRYED LIST. H5=MEANS NO SOL'N, NO OUTPUT. H5+MEANS (0) IS A SOL'N. (M16 TRIES ONE LEVEL AT A TIME)

FIND MAP OF DEF. LEFT SIDES. IF NONE, QUIT--.

FIND MAP OF DEF. RIGHT SIDES. IF NONE, QUIT--.

CLEAR LEVEL
FIND LOWEST LEVEL IN PROBLEM. IF NONE, QUIT--. IF YES,

BUMP 1W3, TEST IF GREATER THAN 1. IF NOT, QUIT--. IF YES,
COPY PROBLEM FOR REPLACEMENT.

SET 'NEW SUBPROBLEM FLAG' TO NO.

GENERATE SEGMENT LOCATIONS.
EXECUTE 'NEW SUBPROBLEM FLAG'. IF YES, GO FINISH IT UP.
BUMP 1W3 AND TEST IF GREATER THAN 1. IF NOT, GO CLEAN UP, QUIT--. IF GREATER, COPY 1W4 AND

ASSIGN COPY TO PROBLEM COPY.

LOOP FOR NEXT LEVEL.
ALL DONE.
ERASE LEFTOVER COPY, QUIT.
SET UP DERIVATION ASSOCIATIONS.

AND

FINISH BUILDING THE NEW SUBPROBLEM.
IF NO GOOD, SET UP TO LOOP.
IF GOOD, TRY SUBSTITUTION.
IF PROOF, QUIT++. IF NOT, LOOP.
9-20 SUBPROCESS--BUMP AND TEST 9-20
IF 1W3 GREATER THAN 1.
TEST IF 1W3 GREATER THAN N2.
IF NO, QUIT SUBPROCESS--.
IF YES,
SUBTRACT 1, QUIT SUBPROCESS++.
9-10C SUBPROCESS FOR SUBSEGMENT REPLACEMENT IN LOCATION (0).
CREATE LIST OF FEASIBLE DEFS.
SAVE LIST FOR LATER ERASURE.
GENERATE DEFS FOR LEFT SIDE MATCH.
IF MATCHED, GO CLEAN UP.
IF FAILED, ERASE OLD LIST,
CREATE NEW LIST OF FEASIBLE DEFS.
SAVE LIST FOR LATER ERASURE.
GENERATE DEFS FOR RIGHT SIDE MATCH.
ERASE LIST AND QUIT + FOR GEN.
9-20C SUBPROCESS, TRY REPLACEMENT BY MATCHING SEGMENT TO LEFT SIDES.
MAKE FREE VARIABLES DISJOINT.
FIND LEFT SEGMENT OF DEF.
IF NONE, QUIT + FOR GEN.
IF FOUND,
MATCH SEGMENT TO LEFT SIDE.
IF NO MATCH, QUIT + FOR GEN.
IF MATCHED, SAVE LIST.
FIND RIGHT SIDE OF DEF.
9-30C SUBPROCESS, TRY REPLACEMENT BY MATCHING SEGMENT TO RIGHT SIDES.
MAKE FREE VARIABLES DISJOINT.
FIND RIGHT SEGMENT OF DEF.
IF NONE, QUIT + FOR GEN.
IF FOUND,
MATCH SEGMENT TO RIGHT SIDE.
IF NO MATCH, QUIT + FOR GEN.
IF MATCHED, SAVE LIST.
FIND LEFT SIDE OF DEF.
IF NONE, CLEAN UP, QUIT ++.
IF FOUND, COPY IT++.
ERASE OLD SEGMENT, AND REPLACE OLD WITH COPY FROM DEF.
SUBSTITUTE IN PROB TEX PER 1W8.

SET 'NEW SUBPROBLEM FLAG' TO ON.
SET H5 TO QUIT.

ERASE 1W8; REVERSE H5 FOR GEN.

M17 SUBLEVEL REPLACEMENT METHOD FOR M17.
PROBLEM (0). ADD ALL NEW
SUBPROBLEMS TO UNTRIED LIST.
H5=MEANS NO SOL'N, NO OUTPJT.
H5+MEANS (0) IS A SOL'N.
(M17 TRIES ALL LEVELS A ONCE)

FIND MAP OF DEF. LEFT SIDES.
IF NONE, QUIT--.

FIND MAP OF DEF. RIGHT SIDES.
IF NONE, QUIT--.

CLEAR LEVEL.
FIND LOWEST LEVEL IN PROBLEM.
IF NONE, QUIT--. IF YES,

BUMP 1W3. TEST IF GREATER THAN 1.
IF NOT, QUIT--.
IF YES,
COPY PROBLEM FOR REPLACEMENT.

SET 'NEW SUBPROBLEM FLAG' TO NO.

GENERATE SEGMENT LOCATIONS.
BUMP 1W3 AND TEST IF GREATER THAN 1
IF NOT, GO TEST FLAG.
IF GREATER, COPY 1W4 AND

ASSIGN COPY TO PROBLEM COPY.

LOOP FOR NEXT LEVEL.
EXECUTE 'NEW SUBPROBLEM FLAG'.
IF YES, GO FINISH IT UP.
ALL DONE.
ERASE LEFTOVER COPY, QUIT.
SET UP DERIVATION ASSOCIATIONS, 9-1  11W4  M01TR370
AND
FINISH BUILDING THE NEW SUBPROBLEM.
IF NO GOOD, QUIT -.
IF GOOD, TRY SUBSTITUTION.
IF PROOF, QUIT. IF NOT, QUIT-
9-20 SUBPROCESS--BUMP AND TEST
IF 1W3 GREATER THAN 1.
TEST IF 1W3 GREATER THAN N2.
IF NO, QUIT SUBPROCESS--.
IF YES,  9-20
70J38
M01TR410
J38  0
M01TR440
M01TR470
M01TR460
M01TR490
M01TR520
M01TR550
M01TR580
M01TR590
M01TR600
M01TR610
M01TR620
M01TR630
M01TR640
M01TR650
M01TR660
M01TR670
M01TR680
M01TR690
M01TR700
M01TR710
M01TR720
M01TR730
M01TR740
M01TR750
M01TR760
M01TR770
M01TR780
M01TR790
M01TR800
M01TR810
M01TR820
M01TR830
M01TR840
M01TR850
M01TR860
M01TR870
M01TR880
M01TR890
SUBTRACT 1, QUIT SUBPROCESS+.
9-100 SUBPROCESS FOR SUBSEGMENT
REPLACEMENT IN LOCATION (0).
CREATE LIST OF FEASIBLE DEFS.
SAVE LIST FOR LATER ERASURE.
GENERATE DEFS FOR LEFT SIDE MATCH.
IF MATCHED, GO CLEAN UP.
IF FAILED, ERASE OLD LIST.
CREATE NEW LIST OF FEASIBLE DEFS.
SAVE LIST FOR LATER ERASURE.
GENERATE DEFS FOR RIGHT SIDE MATCH.
ERASE LIST AND QUIT + FOR GEN.
9-200 SUBPROCESS, TRY REPLACEMENT
BY MATCHING SEGMENT TO
LEFT SIDES.
MAKE FREE VARIABLES DISJOINT.
FIND LEFT SEGMENT OF DEF.
IF NONE, QUIT + FOR GEN.
IF FOUND,
MATCH SEGMENT TO LEFT SIDE.
IF NO MATCH, QUIT + FOR GEN.
IF MATCHED, SAVE LIST.
FIND RIGHT SIDE OF DEF.
9-300 SUBPROCESS, TRY REPLACEMENT
BY MATCHING SEGMENT TO
RIGHT SIDES.
MAKE FREE VARIABLES DISJOINT.
FIND RIGHT SEGMENT OF DEF.
IF NON, QUIT + FOR GEN.
IF FOUND,
MATCH SEGMENT TO RIGHT SIDE.
IF NO MATCH, QUIT + FOR GEN.
IF MATCHED, SAVE LIST.
FIND LEFT SIDE OF DEF.
   IF NONE, CLEAN UP; QUIT +.
   IF FOUND, COPY IT.
ERASE OLD SEGMENT, AND
REPLACE OLD WITH COPY FROM DEF.
SUBSTITUTE IN PROB TEx PER 1W8.
SET 'NEW SUBPROBLEM FLAG' TO ON.
SET H5 TO QUIT-.
ERASE 1W8, REVERSE H5 FOR GEN.

M19 FINISH BUILDING NEW SUBPROBLEM (3) FROM (2) VIA THM(1) BY
METHOD (0); MEASURE UTILITY.
H5-, NO OUTPUT, TEx ERASED.
H5+ MEANS OUTPUT (0) IS OK.
FILL
QUIT
DESCRIPTION.

MARK LOCAL FOR FOUND LIST.
EVALUATE UTILITY.
   IF N.G., REJECT IT.
   IF O.K., ADD TO UNTRIED LIST.
QUIT +., OUTPUT NEW PROBLEM.

TEST IF PRINTING REJECTS.
   IF NO, SKIP IT.
   IF YES, PRINT IT.
ERASE N. G. SUBPROBLEM.
QUIT --.

M40 TEST IF TOTAL EXPRESSIONS (0) AND (1) MATCH.
FIND MAIN SEGMENT.
FIND MAIN SEGMENT.

TEST IF SEGMENTS MATCH.
TEST IF OTHER MAIN SEGMENT EXISTS.

NO, QUIT W/H5+ FOR MATCH
YES, QUIT W/H5- FOR NO MATCH.

1

M41 TEST IF SEGMENTS (0) AND (1) MATCH.

TEST IF 1ST IS VARIABLE.
  IF NOT, EXAMINE SUBSEGMENTS.
  IF YES,

TEST IF SAME VARIABLE.
  IF YES, QUIT, H5+
  IF NO,

TEST IF 2ND IS FREE VARIABLE.
  IF NO, QUIT, H5-
  IF YES,

TEST IF 1ST IS FREE VARIABLE.
EXAMINE SUBSEGMENTS.
TEST IF NOT SEGMENT.
  IF NOT, QUIT, H5-
  IF YES,

TEST IF SAME CONNECTIVES.
  IF NOT, QUIT, H5-
  IF YES,

FIND LEFT SUBSEGMENT.
  IF NONE, CHECK OTHER SIDE.

FIND OTHER LEFT SUBSEGMENT.
  IF NONE, QUIT, H5-
  IF NOT, QUIT, H5-

FIND RIGHT SUBSEGMENT.
  IF NONE, CHECK OTHER SIDE.

FIND OTHER RIGHT SUBSEGMENTS.
  IF NONE, QUIT, H5-

QUIT, REVERSE H5.
NO FIRST SEGMENT ON 1WO.
FIND FIRST SEGMENT ON 1W.
FIND SECOND SEGMENT ON 1w1.
REVERSE H5 AND
QUIT, DISCARD (0).
M42 ADD PROBLEM (0) TO FOUND LIST.
IF CANNOT, SET H5 -.
GET NUMBER OF LEVELS
IF NONE, QUIT -.
GET SUBLIST
GET NUMBER OF DISTINCT VARIABLES
IF NONE, QUIT -.
GET SUBLIST
GET NUMBER OF VARIABLE PLACES
IF NONE, QUIT -.
GET SUBLIST

GENERATE SUBLIST FOR MATCH

INSERT AT END OF LIST

9-200 SUBPROCESS.
COMPARE EXPRESSIONS
9-100 SUBPROCESS, GET SUBLIST.
LOCATE SUBLIST.

CREATE NEW SUBLIST.
SAVE SUBLIST FOR OUTPUT.
INSERT NEW SUBLIST.

COPY DATA TERM.
MARK LOCAL.
INSERT BEFORE NEW SUBLIST.
GET SUBLIST AND QUIT.

M43 MEASURE UTILITY SUBPROBLEM (0).
SET H5+ IF GOOD, H5- IF N.G.
FIND MEX.
GO THRU 'NOTS'.
SAVE UNOTTED MEX.
TEST IF VARIABLE
IF H5-, QUIT (VARIABLE ONLY)

TEST IF MAIN CONNECTIVE 'OR'
IF NOT 'OR', LOOK ON FOUND LIST

LOCATE RIGHT SIDE

GET RIGHT SIDE

LOCATE LEFT SIDE

GET LEFT SIDE
TEST IF SIDES MATCH.

IF SAME QUIT W/H5-

ADD TO FOUND LIST IF CAN

M50 ADD TEX(0) TO TRUE EXPRESSIONS
LIST AND TRUE EXPRESSIONS MAP.
MAKE ALL VARIABLES FREE.

ADD TO LIST

ADD TO MAP

PRINT EXPRESSION AND QUIT.

M51 PRINT NEW SUBPROBLEM (0) AND
ADD TO UNTIRIED SUBPROBLEM LIST.

TALLY PREVIOUS SUBPROBLEM NUMBER.

ASSIGN PROBLEM NO.

FIND NO. OF LEVELS
IF NONE, QUIT --

LOCATE CORRESPONDING LIST.

GET LIST.
ADD NEW SUBPROBLEM.
9-1  J65  9-2  M051R190
   40H0  M051R200
   11W0  M051R210
   J91  M051R223
   J136  M051R226
   J64  M051R230
   J120  M051R240
   J136  M051R250
   J64  9-2  M051R260
   M75  M051R270
   J31  M051R280
   M051R290
   M051R300

CREATE LIST OF ONE SUBPROBLEM.
INSERT NEW LIST.
COPY LEVEL DATA TERM.
INSERT BEFORE NEW LIST.
PRINT NEW SUBPROBLEM.

1

M54 ADD TOTAL EXPRESSION (0) TO
MAP OF TRUE EXPRESSIONS (1).

ADD MAIN SEGMENT (1) TO MAP (0).

9-100 SUBPROCESS, ADD SEGMENT (1)
TO MAP (0).

TEST IF SIMPLE VARIABLE.
IF NO, CONTINUE DOWN MAP.
IF YES, ADD THMNAME.
TEST IF NAME LIST IN MAP HEAD.
IF NO, GO MAKE ONE.
IF YES,

INSERT NAME AND QUIT.

CREATE LIST OF ONE NAME.

PLACE IN MAP HEAD AND QUIT.
INPUT MAP HOLDER.
INPUT SEGMENT CONNECTIVE.
FIND SUBMAPS LIST.
IF FOUND, CONTINUE.
IF NONE,

CREATE 1ST LOCAL SUBMAP.

TEST IF 2ND SUBLIST NEEDED.
IF NO, SKIP IT.
IF YES,
CREATE 2ND LOCAL SUBMAP.

CREATE SUBMAP LIST.

-145-
ASSIGN AS SUBMAP LIST OF CONNECTIVE

FIND 1ST SUB SEGMENT.
  IF NONE, QUIT.

FIND 1ST SUB MAP.
  IF NONE, QUIT.

ADD SEGMENT (1) TO SUBMAP (0).

TEST IF MORE SEGMENTS.
  IF NO, QUIT.

FIND 2ND SUBSEGMENT.
  IF NONE, QUIT.

FIND 2ND SUBMAP.
  IF NONE, QUIT.

ADD SEGMENT (1) TO SUBMAP (0).

M60 FIND NEXT UNTRIED PROBLEM.
  H5 - MEANS NONE REMAINING.
LOCATE NEXT SUBLIST OF PROBLEMS.
  IF NONE, QUIT.
  IF SOME, GET SUBLIST AND
FIND FIRST PROBLEM.
  IF NONE, LOCATE NEXT LIST.
  IF FOUND, GET LOCATION OF LIST,
GET NAME OF LIST, AND
LOCATE FIRST PROBLEM.
  IF NONE, MACHINE ERROR--HALT.
  IF LOCATED, DELETE FROM LIST,
MARK OUTPUT REGIONAL, QUIT.+.

M62 CREATE A LIST OF TRUE
  EXPRESSIONS FROM MAP (1) FOR
  FEASIBLE MATCH WITH SEGMENT
  (0). OUTPUT MAY BE EMPTY.
SAVE COPY OF LIST IN MAP HEAD.

TEST IF SEGMENT IS SIMPLE VAR.
  IF YES, QUIT WITH OUTPUT.
  IF NO,
LOCATE 1ST SUBSEGMENT.
  IF NONE, OUTPUT 1W2, QUIT.
FIND LIST OF APPROPRIATE SUBMAPS.
IF NONE, OUTPUT 1W2, QUIT.
LOCATE 1ST SUBMAP.
IF NONE, OUTPUT 1W2, QUIT.
IF THERE, SAVE LOCATION.
SET UP HO AND
CREATE LIST FROM MAP (1) FOR (0).
TEST IF CONNECTIVE WAS NON-UNARY.
IF UNARY, FIX OUTPUT, QUIT.
IF NON-UNARY, SAVE LIST AND
LOCATE NEXT SUBSEGMENT.
IF NONE, FIX OUTPUT, QUIT.
LOCATE NEXT SUBMAP.
IF NONE, FIX OUTPUT, QUIT.
CREATE LIST FROM MAP (1) FOR (0),
'AND' RESULT WITH LIST 1W5, LOOP.

FIX OUTPUT --
'OR' W5 WITH 1W2,
LEAVE RESULT AS 1W2.
OUTPLT 1W2, CLEAR CONTEXT.

SUBPROCESS -- 'AND' (0) WITH 1W5.
GENERATE 1W5 FOR PROCESS MARKING.

GENERATE '(0)' TO UNMARK MARKED.
ERASE '(0)'

LOCATE NEXT OF 1W5.
IF NONE, QUIT SUBPROCESS.
TEST IF EXPRESSION MARKED.
IF NO, LOOP TO NEXT.
IF YES, UNMARK IT,
SAVE LOCATION AND DELETE THIS
EXPRESSION DUE NOT ON BOTH LISTS.
IF MORE, LOOP WITH NEXT.
SUBPROCESS - MARK PROCESSED.
SUBPROCESS - UNMARK IF MARKED.
M63 CREATE A LIST OF TRUE
EXPRESSIONS FROM MAP (Q) FOR
FEASIBLE MATCH WITH TEX (1).*
OUTPUT MAY BE AN EMPTY LIST.*

M70 PRINT EXPRESSION (O), WITH OR
WITHOUT A SUFFIX.*
ENTER NAME.*
TAB TO COLUMN (O).*
FIND MEX.*
IF NONE, SKIP IT.*
ENTER MEX.*
FIND SUFFIX.*
IF NONE, PRINT WITHOUT IT.*
BUMP COLUMN.*
ENTER SUFFIX AND PRINT.*

M71 PRINT PROOF SEQUENCE FROM (O).*
SKIP TWO LINES.*
ENTER 'PROOF FOUND' AND
PRINT.*
SKIP ONE LINE.*
FIND PROVING THEOREM.*
IF NONE, USE DUMMY CHARACTER.*
ENTER 'GIVEN'.*
PRINT FIRST LINE OF PROOF.*
INPUT TEX AND
'SUBSTITUTION'.*
PRINT NEXT EVEN LINE.*
FIND THEOREM USED IN DERIVATION.*
IF NONE, FINISH WITH Q.E.D.*
PRINT NEXT ODD LINE.*
FIND METHOD OF DERIVATION
FIND EXTERNAL NAME OF METHOD.
IF NONE, USE BLANKS.
FIND PROBLEM USED IN DERIVATION.
IF NONE, USE DUMMY CHARACTER.
LOOP TO PRINT NEXT EVEN LINE.
TAB TO COLUMN K41.
Enter 'Q.E.D.' AND PRINT.
PRINT LIMITS, CLEAR CONTEXT/QUIT.
PRINT -NO PROOF FOUND-
ENTER MESSAGE
PRINT MESSAGE, LIMITS
ENTER SEGMENT (0)
TEST IF VARIABLE
IF YES, ENTER VARIABLE.
TEST IF CONNECTIVE NOT
ENTER NOT.
LOCAT FIRST SEGMENT.
IF NONE, QUIT.
ENTER SEGMENT.
LOCATE NEXT SUBSEGMENT.
IF NONE, QUIT.
ENTER CONNECTIVE.
ENTER SEGMENT.
TEST IF VARIABLE

ENTER LEFT PAREN.
ENTER SUBEXPRESSION
ENTER RIGHT PAREN.

M74 ENTER TOTAL EXPRESSION (0),
 WITH OR WITHOUT SUFFIX.
ENTER MAIN EXPRESSION.
FIND SUFFIX.
 IF NONE, QUIT.
 IF ONE,
BJMP COLUMN AND ENTER SUFFIX.

M75 PRINT NEW SUBPROBLEM (0).
TAB TO COLUMN K47.
ENTER SUBPROBLEM NAME (NO.).
BUMP COLUMN.
ENTER SUBPROBLEM EXPRESSION.

TEST IF EXPRESSION WAS TOO BIG.
 IF YES, DON'T RESET.
 IF NO, RESET TO K48.
BUMP COLUMN.
FIND THEOREM.
 IF NONE, SKIP IT.
ENTER COMMA.

BUMP COLUMN.
FIND METHOD.
 IF NONE, PRINT NOW.
FIND EXTERNAL NAME.
 IF NONE, PRINT NOW.
ENTER TEXT AND PRINT.

M76 ENTER LIST OF DATA TERMS.

P4    M073R235
709-201 M073R238
P8    M073R240
70    M073R250
10K51  M073R260
10J52  M073R270
J157  M073R280
M73    M073R290
10N1   M073R300
J157  R
50/14  M79
M74    R
40H0   M074R000
J81    M074R010
70J8   M074R020
M73    M074R030
Q18    M074R040
10N1   M074R050
J161   M074R060
J157   M074R070
M75    R
J154   M075R000
10K47  M075R010
J160   M075R020
M74    M075R030
40H0   M075R040
M79    M075R050
10N3   M075R060
J161   M075R070
40H0   M075R080
PSV PROB M075R090
M74    M075R100
11W25  M075R110
10K48  M075R120
J116   M075R130
70     M075R140
9-2    M075R150
10N2   M075R160
J161   M075R170
40H0   M075R180
Q12    M075R190
709-1  M075R200
M79    M075R210
10N1   M075R220
J161   M075R230
J157   M075R240
10J157 M075R250
M76    M075R260
M76    M075R270
Q16    M075R280
10J155 M075R290
M76    M075R300
R
M76    M076R000
M77 PRINT LIMITS OF PROOF.

DOUBLE SPACE.

SET K12 TO ACTUAL EFFORT.

INPUT 'EFFORT'
PRINT LINE.

INPUT 'SUBPROBLEMS'
PRINT LINE.

INPUT 'SUBSTITUTIONS'
9-100 SUBPROCESS, PRINT LINE.
ENTER MESSAGE.

TAB TO COLUMN K44.

ENTER 'LIMIT'

BUMP COLUMN.
ENTER LIMIT.

TAB TO COLUMN K45.

ENTER 'ACTUAL'

BUMP COLUMN.
ENTER ACTUAL AND PRINT LINE.

M78 PRINT 'TO PROVE' PROBLEM (0).

INPUT TEXT.
ENTER MESSAGE.

SET UP TO PRINT ON NEW PAGE.

RESTORE SPACING AND PRINT (0).

M79 ENTER NAME OF (0).

FIND EXTERNAL NAME.

IF THERE
ENTER IT, DISCARD (0).

IF NOT THERE
9-1
40H0 PSV (0) Q8
M079R050
IF NOT THERE, ENTER INTERNAL.
IF THERE, ENTER NO.
AND ENTER PERIOD.

M80 PRINT PROOF LINE.
INPUT (G) IS METHOD OR 'GIVEN'
INPUT (1) IS TEXT
ENTER METHOD

ENTER NAME

ENTER EXPRESSION AND PRINT.

M81 PRINT REJECTED PROBLEM (C).
TAB TO COLUMN K47.
ENTER NAME.
BUMP COLUMN.
ENTER TEX.

TEST IF TEXT TOO LONG.
IF YES, SKIP RESET.
IF NO,
TAB TO COLUMN K48
BUMP COLUMN.

FIND THEOREM.
IF NONE, SKIP IT.
IF THERE, ENTER NAME.
ENTER COMMA, AND
BUMP COLUMN.
FIND METHOD.
IF NONE, SKIP IT.
IF THERE, FIND EXTERNAL NAME.
IF NONE, SKIP IT.
IF THERE, ENTER TEXT.
ENTER PERIOD, AND
BUMP COLUMN.

70J156
J157
50K53 J157
M079R060
M079R070
M079R080
R
M080R000
M080R010
M080R020
M080R030
M080R040
M080R050
M080R060
M080R070
M080R080
M080R090
M080R100
R
M081R000
M081R010
M081R020
M081R030
M081R040
M081R050
M081R060
M081R070
M081R080
M081R090
M081R100
M081R110
M081R120
M081R130
M081R140
M081R150
M081R160
M081R170
M081R180
M081R190
M081R200
M081R210
M081R220
M081R230
M081R240
M081R250
M081R260
M081R270
M081R280
M081R290
M081R300
M081R310
M081R320
M081R330
M081R340
ENTER MESSAGE AND PRINT.

MB2 PRINT 'REMEMBER PROVED THEOREM.' MB2

ENTER MESSAGE.

MB8 PRINT BAD LIST FORM EXPRESSION. MB8

ENTER 'BAD EXPRESSION'.

BUMP COLUMN.

GEN SYMBOLS FOR ENTRY, PRINT.

MB9 READ NEXT LOGIC EXPRESSION MB9
FROM NORMAL INPUT UNIT.
H5-- MEANS NONE THERE.

CLEAR AND
FILL BUFFER.
IF EOF, QUIT, H5--.

LOCATE 1ST OF NAME.
IF BLANK CARD, QUIT, H5--.

DETERMINE EXTENT.
IF REST OF CARD, RESET, GET NEXT

INPUT NAME.

TEST IF REGIONAL.
IF NOT, RESET, GET NEXT.

TEST IF NAME IS A CHARACTER SYMBOL.
IF YES, RESET, GET NEXT.

IF OK, GET EXTERNAL NAME.

RESET COLUMN TO 1ST OF NAME.

CREATE PROTOTYPE DATA TERM.
MARK LOCAL.
SET D.T. TO HOLD NAME.

M76 J155
M051R350
R
M082R000
M082R010
M082R020
M082R030
M082R040
M082R050
M082R060

M154
M155
M155
10T23
M76
J155
J154 J155

M154
10T24
M76
10N3
J161
10MT9
J100 J155

M086R000
M086R010
M086R020
M086R030
M086R040
M086R050
M086R060

M089R000
M089R010
M089R020
M089R030
M089R040
M089R050
M089R060
M089R070
M089R080
M089R090
M089R110
M089R120
M089R125
M089R130
M089R140
M089R150
M089R160
M089R170

M089R180
M089R190
M089R200
M089R210
M089R213
M089R215
M089R217
M089R220
M089R230
M089R240
M089R250
M089R260
M089R265
M089R270
M089R280
M089R290
M089R300
ASSIGN D.T. AS EXTERNAL NAME.
LOCATE 1ST OF EXPRESSION.
   IF NONE, RESET, GET NEXT.
FIND CHARACTER SYMBOL AT 1W25.
   IF NONE, EXPRESSION FINI.
   IF FOUND, GET ALTERNATE.
ADD AT END OF LIST EXPRESSION.
TALLY 1W25 AND LOOP FOR NEXT.
LOCATE FIRST OF SUFFIX.
   IF NONE, SKIP IT.
DETERMINE EXTENT.
   IF NONE, SKIP IT.
CREATE PROTOTYPE.
SET D.T. TO SUFFIX.
ASSIGN AS SUFFIX OF 1W0.
ADJUST FOR EXTRA IN H5.
SET H5+
DISCARD COLUMN D.T.
DISCARD EXTENT D.T.
RESET
   AFTER A
BAD EXPRESSION
   AND GET NEXT CARD.
M90 TEST IF PROBLEM LIMITS REACHED.
   M90 TEST NO. OF SUBPROBLEMS SET UP
TEST NO. OF SUBSTITUTIONS
COMPUTE EFFORT

TEST EFFORT.

M110 MAKE FREE VARIABLES OF TOTAL EXPRESSIONS (0) AND (1)
   DISJOINT, (SUBSTITUTES IN (1),)
   CREATE FREE VAR. LIST FOR (1)

CREATE FREE VAR. LIST FOR (0)

CREATE EMPTY SUBSTITUTION LIST.

CREATE EMPTY LIST OF MARKED PROCESSED

SET LOCATION ON SYSTEM FREE VARIABLE LIST

MARK ALL OF (0)-S FREE VARS.

ADD DUPLICATES TO SUBSTITUTION LIST

TEST IF ANY DUPLICATES.

UNMARK ALL MARKED VARIABLES

LOCATE NEXT DUPLICATE

LOCATE NEXT SYSTEM FREE VAR.

HALT DUE TO NOT ENOUGH FREE VAR.

TEST IF USED IN EITHER.

INSERT AS SUBSTITUTOR.
NUMARK ALL MARKED VARIABLES.

GET MAIN SEGMENT OF (1).

SUBSTITUTE.

MARK PROCESSED

ADD TO LIST 1w7 FOR UNMARKING
ADD THOSE MARKED TO SUBST. LIST.
MARK THOSE NOT MARKED.

UNMARK PROCESSED.

UNMARK PROCESSED ALL VARIABLES
ON LIST 1w7

M111 MATCH SEGMENTS (0) AND (1).
H5+ MEANS OUTPUT (0) IS LIST OF PAIRS--1ST IS FREE VAR.,
2ND IS SUBSTITUTE.
H5- MEANS NO MATCH--NO OUTPUT.
9-100 MATCH SUBPROCESS
(EXPECTS FREE VARIABLES DISJOIN IS (0) A VARIABLE.
IS (0) A FREE VARIABLE.

IS THERE ALREADY A SUBSTITUTOR FOR (0).?
SET SUBSTITUTE AS (0) AND MATCH.

TEST IF (1) IS A VARIABLE
(1) IS VARIABLE, TEST (1)=(0)

NO SUBSTITUTOR, QUIT W/H5+

IF NOT, MAKE EXPRESSION INTERNAL
ASSIGN (1) AS SUBSTITUTOR FOR (0)

( (0) IS VARIABLE, NOT FREE, )
IS (1) A VARIABLE.

IS (1) A FREE VARIABLE.

IS THERE ALREADY A SUBSTITUTOR
FOR (1).
SET SUBSTITUTOR AS (1) AND MATCH.

TEST IF (0) IS A VARIABLE

(0) IS VARIABLE, TEST (0)=(1)

IF NOT, MAKE EXPRESSION INTERNAL
ASSIGN (0) AS SUBSTITUTOR FOR (1)
( BOTH ARE VARIABLES, NOT FREE )

ARE VARIABLES IDENTICAL.
( (0) IS EXPRESSION, )
IS (1) A VARIABLE

IS (1) A FREE VARIABLE.
( BOTH ARE EXPRESSIONS, )

ARE CONNECTIVES IDENTICAL

LOCATE NEXT SUBSEGMENT ON (0)

LOCATE NEXT SUBSEGMENT ON (1)
MATCH SEGMENTS (0) AND (1) FOR SUBSTITUTION. SETS H5, IF + OUTPUT (0) IS EXPANDED SUBSTITUTION LIST.

MATCH SUBSEGMENTS

LOCATE NEXT SUBSEGMENT ON (1)

M112 EXPAND SUBSTITUTION LIST (0).
REPLACE EXPRESSIONS WITH COMPLETELY SUBSTITUTED LOCALLY NAMED COPIES.
LOCATE NEXT SUBSTATION

1

M112  40HC  M112  R000
       J51   M112  R010
         J60   M112  R020
         J60   M112  R030
       J60   M112  R040
       J60   M112  R050
       J60   M112  R060
       J60   M112  R070
       J60   M112  R080
       J60   M112  R090
       J60   M112  R100
       J60   M112  R110
       J60   M112  R120
       J60   M112  R130
       J60   M112  R140
       J60   M112  R150
       J60   M112  R160
       J60   M112  R170
       J60   M112  R180
       J60   M112  R190
       J60   M112  R200
       J60   M112  R210
       J60   M112  R220
       J60   M112  R230
       J60   M112  R240
       J60   M112  R250
       J60   M112  R260
       J60   M112  R270
       J60   M112  R280
       J60   M112  R290
       J60   M112  R300
       J60   M112  R310
       J60   M112  R320
       J60   M112  R330
       J60   M112  R340
       J60   M112  R350
       J60   M112  R360
       J60   M112  R370

DELINEATE SEGMENT AT THIS STATION

SEGMENT DELINEATION SUBPROCESS.

IS INPUT A VARIABLE.

IS INPUT A FREE VARIABLE.

DELINEATE FREE VARIABLE.

CREATE SUBSTITUTED LOCAL COPY

GENERATE FREE VARIABLE LOCATIONS.

DELINEATE FREE VARIABLE

FIND CORRESPONDING SUBSTITUTE

DELINEATE SUBSTITUTE

OUTPUT FREE VARIABLE

1

M113  M111  M113  R000
       703   M113  R010
       40HC  M113  R020
         J4   M113  R030
TEST IF SEGMENT (0) MATCHES SEGMENT (1).

M114 M111
7CC J71

M115 SUBSTITUTE IN SEGMENT (0)
FROM SUBSTITUTION LIST (1).
SUBSTITUTES ONLY FOR VARIABLES.
SAMPLE SUBST. LIST.
IF EMPTY, QUIT.

TEST IF FREE VARIABLE.

IF YES,
GENERATE LOCATIONS OF FREE VARIABLES.
TEST IF BOUND VARIABLE.

IF YES,
GENERATE LOCATIONS OF BOUND VARIABLES.

TEST IF A VARIABLE

IF AN EXPRESSION, COPY IT.
STORE IN LOCATION

CREATE LIST OF FREE VARIABLES IN TEX (0), SETS H5, NO OUTPUT IF --
GET MAIN SEGMENT.

GENERATE LOCATIONS OF FREE VARS.

TEST IF ANY FREE VARS.

ADD TO OUTPUT IF NOT ALREADY ON.

M115 FIND LIST OF BOUND VARIABLES IN TEX (0).
H5- MEANS NO OUTPUT.
<table>
<thead>
<tr>
<th>P4</th>
<th>GO THRU NOTS OF SEGMENT (O), LEAVE 1ST UNNOTED SEGMENT. H5- MEANS NO OUTPUT. QUIT, H5+ MEANS NORMAL EXIT. FIND SUBSEGMENT OF NOT. IF NONE QUIT -, ELSE LOOP.</th>
</tr>
</thead>
<tbody>
<tr>
<td>P6</td>
<td>TEST IF (O) IS NOT UNARY.</td>
</tr>
<tr>
<td>P7</td>
<td>TEST IF (O) IS CONNECTIVE</td>
</tr>
<tr>
<td>P8</td>
<td>TEST IF (O) IS VARIABLE</td>
</tr>
<tr>
<td>P9</td>
<td>TEST IF (1) IS FREE VARIABLE</td>
</tr>
<tr>
<td>P12</td>
<td>FIND MEX OF TEX (O).</td>
</tr>
<tr>
<td>P13</td>
<td>FIND LEFT SIDE OF TEX(O).</td>
</tr>
<tr>
<td>P14</td>
<td>FIND RIGHT SIDE OF TEX(O).</td>
</tr>
<tr>
<td>P15</td>
<td>TEST IF (O) IS IN INTERNAL (TREE) FORM.</td>
</tr>
<tr>
<td>P16</td>
<td>FIND MAIN CONNECTIVE OF TEX (O).</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P2</th>
<th>TEST IF (O) IS A BOUND VARIABLE.</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3</td>
<td>CLEAR DESCRIPTIONS OF TOTAL EXPRESSION (O)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P4</th>
<th>12HC</th>
</tr>
</thead>
<tbody>
<tr>
<td>P6</td>
<td>11K6</td>
</tr>
<tr>
<td>P7</td>
<td>700</td>
</tr>
<tr>
<td>P8</td>
<td>700</td>
</tr>
<tr>
<td>P9</td>
<td>700</td>
</tr>
<tr>
<td>P12</td>
<td>700</td>
</tr>
<tr>
<td>P13</td>
<td>700</td>
</tr>
<tr>
<td>P14</td>
<td>700</td>
</tr>
<tr>
<td>P15</td>
<td>700</td>
</tr>
<tr>
<td>P16</td>
<td>700</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P2</th>
<th>G9</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3</td>
<td>40H0</td>
</tr>
<tr>
<td></td>
<td>10Q2</td>
</tr>
<tr>
<td></td>
<td>J14</td>
</tr>
<tr>
<td></td>
<td>40HC</td>
</tr>
<tr>
<td></td>
<td>J14</td>
</tr>
<tr>
<td></td>
<td>10Q4</td>
</tr>
<tr>
<td></td>
<td>J14</td>
</tr>
<tr>
<td></td>
<td>700</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P4</th>
<th>10Q3</th>
</tr>
</thead>
<tbody>
<tr>
<td>P6</td>
<td>Q14</td>
</tr>
<tr>
<td>P7</td>
<td>Q14</td>
</tr>
<tr>
<td>P8</td>
<td>Q5</td>
</tr>
<tr>
<td>P9</td>
<td>Q6</td>
</tr>
<tr>
<td>P12</td>
<td>J81</td>
</tr>
<tr>
<td>P13</td>
<td>J81</td>
</tr>
<tr>
<td>P14</td>
<td>J81</td>
</tr>
<tr>
<td>P15</td>
<td>Q15</td>
</tr>
<tr>
<td>P16</td>
<td>J81</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>P2</th>
<th>700</th>
</tr>
</thead>
<tbody>
<tr>
<td>P3</td>
<td>700</td>
</tr>
<tr>
<td>P4</td>
<td>700</td>
</tr>
<tr>
<td>P6</td>
<td>700</td>
</tr>
<tr>
<td>P7</td>
<td>700</td>
</tr>
<tr>
<td>P8</td>
<td>700</td>
</tr>
<tr>
<td>P9</td>
<td>700</td>
</tr>
<tr>
<td>P12</td>
<td>700</td>
</tr>
<tr>
<td>P13</td>
<td>700</td>
</tr>
<tr>
<td>P14</td>
<td>700</td>
</tr>
<tr>
<td>P15</td>
<td>700</td>
</tr>
<tr>
<td>P16</td>
<td>700</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>M117</th>
<th>R030</th>
</tr>
</thead>
<tbody>
<tr>
<td>P002</td>
<td>R000</td>
</tr>
<tr>
<td>P022</td>
<td>R010</td>
</tr>
<tr>
<td>P033</td>
<td>R000</td>
</tr>
<tr>
<td>P038</td>
<td>R010</td>
</tr>
<tr>
<td>P039</td>
<td>R020</td>
</tr>
<tr>
<td>P03R</td>
<td>R030</td>
</tr>
<tr>
<td>P03R</td>
<td>R040</td>
</tr>
<tr>
<td>P03R</td>
<td>R050</td>
</tr>
<tr>
<td>P04R</td>
<td>R000</td>
</tr>
<tr>
<td>P04R</td>
<td>R010</td>
</tr>
<tr>
<td>P04R</td>
<td>R020</td>
</tr>
<tr>
<td>P04R</td>
<td>R030</td>
</tr>
<tr>
<td>P04R</td>
<td>R040</td>
</tr>
<tr>
<td>P04R</td>
<td>R050</td>
</tr>
<tr>
<td>P05R</td>
<td>R000</td>
</tr>
<tr>
<td>P05R</td>
<td>R010</td>
</tr>
<tr>
<td>P05R</td>
<td>R020</td>
</tr>
<tr>
<td>P05R</td>
<td>R030</td>
</tr>
<tr>
<td>P06R</td>
<td>R010</td>
</tr>
<tr>
<td>P06R</td>
<td>R020</td>
</tr>
<tr>
<td>P07R</td>
<td>R000</td>
</tr>
<tr>
<td>P07R</td>
<td>R010</td>
</tr>
<tr>
<td>P08R</td>
<td>R000</td>
</tr>
<tr>
<td>P08R</td>
<td>R010</td>
</tr>
<tr>
<td>P09R</td>
<td>R000</td>
</tr>
<tr>
<td>P09R</td>
<td>R010</td>
</tr>
<tr>
<td>P12R</td>
<td>R000</td>
</tr>
<tr>
<td>P12R</td>
<td>R010</td>
</tr>
<tr>
<td>P13R</td>
<td>R000</td>
</tr>
<tr>
<td>P13R</td>
<td>R010</td>
</tr>
<tr>
<td>P14R</td>
<td>R000</td>
</tr>
<tr>
<td>P14R</td>
<td>R010</td>
</tr>
<tr>
<td>P15R</td>
<td>R000</td>
</tr>
<tr>
<td>P15R</td>
<td>R010</td>
</tr>
<tr>
<td>P16R</td>
<td>R000</td>
</tr>
<tr>
<td>P16R</td>
<td>R010</td>
</tr>
<tr>
<td>P16R</td>
<td>R020</td>
</tr>
<tr>
<td>P16R</td>
<td>R030</td>
</tr>
</tbody>
</table>
P17 CREATE COPY OF SEGMENT (0)
   IF NOT A SIMPLE VARIABLE.
COPY AND MARK LOCAL.

P18 TEST IF (0) IS A
   CHARACTER SYMBOL.

GET APPROPRIATE INTERNAL
   CHARACTER SYMBOL (0) FOR
   EXTERNAL CHARACTER SYMBOL (0).
   IF REPLACED, QUIT.

P20 MAKE FAKE TEX WITH LEFT SIDE
   OF TEX (0).

P21 MAKE FAKE TEX WITH RIGHT SIDE
   OF TEX (0).

P22 CREATE NEW SUBPROBLEM WITH
   SEGMENT (0) ON THE LEFT,
   SEGMENT (1) ON THE RIGHT,
   AND IMPLIES AS CONNECTIVE.
CREATE MEX.
INSERT CONNECTIVE.

CREATE TEX, CLEAN UP, QUIT.

ERASE MADE EXPRESSION (0)
DELETE MAIN SEGMENT

P24 MAKE TEX FROM MEX (0).

DESCRIBE AS IN TREE FORM.

P25 COPY TEX (0) FOR SUBSTITUTION.

CLEAR EXTERNAL NAME
   AND CLEAR DESCRIPTIONS.

P26 GENERATE SEGMENT LOCATIONS AT
   LEVEL (2) OF PROBLEM (1) FOR
   PROCESS (0).
TEST IF MEX IS A VARIABLE.
    IF YES, QUIT.
    IF NO, REVERSE,

CREATE COUNTER WITH VALUE 1, AND
SAVE BOTH LEVEL AND COUNTER.

GENERATE SUBSEGMENT
LOCATIONS FOR 9-100.
ERASE COUNTER AND QUIT.
POP HC,
POP HO, AND QUIT.
SUBPROCESS 9-100.

TEST IF THIS IS THE LEVEL.
    IF YES, FIRE J18.
    IF NO, GENERATE
    SUBSEGMENTS FOR 9-100.
SUBGENERATOR 9-200.

CREATE NEW SUBLEVEL COUNTER.
BUMP COUNTER.

LOCATE NEXT SEGMENT PLACE.
    IF NONE, QUIT.
    IF FOUND, PRESERVE LOCATION.

TEST IF SEGMENT IS A VARIABLE.
    IF NO, FIRE J18.
    IF YES, LOOP TO LOCATE.

IF J18 QUIT++, LOOP TO LOCATE,
IF J18 QUIT-, ERASE COUNTER AND QUIT.
P27 REPLACE BOUND BY FREE IN (O).

CREATE LIST OF BOUND OF 1WO.
LOCATE FIRST BOUND.
IF NO BOUND, QUIT.
CREATE LIST OF FREE OF 1WO.
GEN FREE VAR TO BE MARKED.

GEN SYS FREE VAR TO REPLACE.
H5+ MEANS NOT ENOUGH FREE VAR.

FIND MAIN SEGMENT.

IF NONE, SKIP IT.
REPLACE IN MEX FROM 1W1.

GEN FREE VAR TO BE UNMARKED.

ERASE CREATED FREE LIST.

ERASE BOUND LIST.
9-10G SUBPROCESS, INSERT (0)
AFTER SYMBOL IN 1W2
IF (O) IS UNMARKED.
THEN ADVANCE TO NEXT
AFTER CELL HOLDING
INSERTED SYMBOL.

QUIT; H5- MEANS QUIT GENERATOR.
9-200 SUBPROCESS, UNMARK PROCESSED.

GENERATE LOCATIONS OF FREE
VARIABLES WITHIN SEGMENT (1)
FOR PROCESS (0).

TEST IF INPUT SEGMENT IS VARIABLE.
IF SO, QUIT.
LOCATE NEXT SUBSEGMENT.

TEST IF SUBSEGMENT IS FREE VAR.
IF SO, GENERATE LOCATION.

TEST IF SUBSEGMENT IS VARIABLE.
IF NOT, GENERATE SUBSEGMENT.
1 P29 GENERATE LOCATIONS OF BOUND VARIABLES WITHIN SEGMENT (1) FOR PROCESS (0).
   IF YES, QUIT H5+.
   IF NO, LOCATE NEXT SEGMENT
   IF NONE, QUIT, H5+.
   TEST IF FREE VARIABLE.
   IF YES, GET NEXT.
   IF NO, TEST IF BOUND VARIABLE.
   IF NO, GENERATE ON SEGMENT.
   IF YES, FEED LOCATION TO PROCESS.
   IF H5-, SUBPROCESS SAID QUIT.
   INPUT SEGMENT.
   INPUT PROCESS.
   GENERATE LOCATIONS OF BND. VAR. 1 P30 CREATE LIST OF FREE VARIABLES IN TEXT (0).
   FIND MAIN SEGMENT.
   IF NONE, CLEAN UP, QUIT.
   GEN. LOCATIONS OF FREE VARIABLES.
   SUBPROCESS, ADD FREE VARIABLE 2HC TO LIST 1WC IF NOT ON.
   QUIT, H5+ FOR GEN.

1 P31 CREATE LIST OF BOUND VARIABLES. IN TEXT (0).
   FIND MAIN SEGMENT.
   IF NONE, CLEAN UP, QUIT.
   GENERATE LOCATIONS OF BOUND VAR.
   SUBPROCESS, ADD BOUND VARIABLE 2HC TO LIST 1WC IF NOT ON.
   QUIT, H5+ FOR GEN.

1 P50 CONVERT LOGIC EXPRESSION (0) TO INTERNAL (TREE) FORM IF IN EXTERNAL (LIST) FORM. ENTIRE
   40HC 15
   70 78
EXPRESSION MUST BE ENCLOSED
IN PARENTHESES, NO OUTPUT.
H5- MEANS FAILURE.

CREATE NEW MAIN SEGMENT.
IF FAILED, QUIT.
SAVE NEW MEX.
SAVE OLD HEAD.
DISCARD OLD LIST.

INSERT MEX UNDER OLD HEAD.

DESCRIBE AS IN INTERNAL FORM.

1

P51 REPLACE ALL DELIMITED EXTERNAL
CONNECTIVES IN EXPRESSION (O)
IF (O) IS IN EXTERNAL
LIST FORM.

J41
60 W0
P51
11 W0
P52
70 J31
60 W1
51 W0
J75
J71
11 W0
11 W1
J64
11 W0
10 Q15
10 Q15
J11
J31

R
P050 R030
P050 R040
P050 R050
P050 R060
P050 R070
P050 R075
P050 R080
P050 R090
P050 R100
P050 R110
P050 R120
P050 R130
P050 R140
P050 R150
P050 R200
P050 R210
P050 R220
P051 R000
P051 R005
P051 R010
P051 R015
P051 R020
P051 R025
P051 R030
P051 R040
P051 R050
P051 R060
P051 R070
P051 R075
P051 R080
P051 R090
P051 R100
P051 R110
P051 R120
P051 R130
P051 R140
P051 R150
P051 R200
P051 R210
P051 R220

LOCATE FIRST DELIMITER IN LIST.

LOCATE CONNECTIVE (EXTERNAL FORM).

LOCATE SECOND DELIMITER IN LIST.
IF NOT ALL THERE, QUIT.

TEST IF 2ND IS SAME AS 1ST.
IF NOT, TRY ON REMAINDER.
IF YES, DELETE 2ND.

FIND INTERNAL FORM.

REPLACE EXTERNAL.

DELETE FIRST DELIMITER.
RESET AND DO IT AGAIN.
ALL DONE, CLEAN UP AND QUIT.

1

P52 CREATE MAIN SEGMENT FROM
LIST (O). H5- MEANS NO OUTPUT
DUE TO BAD EXPRESSION.
9-100 SUBPROCESS CREATE NEXT SGMNT.
H5- MEANS NO OUTPUT.

J41
20 W1
9-100 J31
9-100 04 J0
11 W1

1W1=CURLOC, CP052 R020
1W1=CURLOC, CP052 R020
1W1=CURLOC, CP052 R020
1W1=CURLOC, CP052 R020
1W1=CURLOC, CP052 R020
LOCATE FIRST OF EXPRESSION
IF NONE, QUIT.

TEST IF OPENING PAREN.
IF YES, BUILD SEGMENT.

TEST IF NOT.
IF YES, BUILD SEGMENT.

TEST IF VARIABLE.
IF NO QUIT.
OUTPUT VARIABLE.

TAKE ERROR ACTION, THEN
ERASE USELESS SEGMENT.

OUTPUT SEGMENT.

BUILD SEGMENT
CREATE 1ST SUBSEGMENT.
IF NONE, CLEAN UP, QUIT.
INSERT 1ST SUBSEGMENT.

LOCATE NEXT SYMBOL.

TEST IF CONNECTIVE.
IF NOT, CLEAN UP, QUIT.

INSERT CONNECTIVE.
CREATE 2ND SUBSEGMENT.
IF NONE, CLEAN UP, QUIT.
INSERT 2ND SUBSEGMENT.

LOCATE NEXT SYMBOL.

TEST IF CLOSING PAREN.
IF NO, CLEANUP, QUIT.
BUILD NOTTED SEGMENT.

NOT THE HEAD.
CREATE SUBSEGMENT.
IF NONE, CLEAN UP AND QUIT.
INSERT
9-200 SUBPROCESS, SET UP EMPTY SEG.

01
12W1
709-101
J60
20W1
709-101
J2
12W1
101
J2
709-101
J2
709-101
J2
12W1
709-101
J2
12W1
9-101
16/14
9-103
11W1
9-102
11W1
9-104
11W1
9-110
9-200
1W0=NEWSEG
P052R060
P052R070
P052R080
P052R090
P052R100
P052R100
P052R100
P052R120
P052R130
P052R140
P052R150
P052R160
P052R170
P052R180
P052R190
P052R200
P052R205
P052R210
P052R220
P052R230
P052R240
P052R250
P052R260
P052R270
P052R280
P052R290
P052R300
P052R310
P052R320
P052R330
P052R340
P052R350
P052R360
P052R370
P052R380
P052R390
P052R400
P052R410
P052R420
P052R430
P052R440
P052R450
P052R460
P052R470
P052R480
P052R490
P052R500
P052R510
P052R520
P052R530
P052R540
P052R550
9-300 SUBPROCESS, INSERT SEGMENT.

1

P55 LOCATE SUBLIST FOLLOWING
DATA TERM (0) ON LIST (1)
H5+ MEANS OUTPUT (0) IS
CELL HOLDING SUBLIST.
H5- MEANS OUTPUT (0) IS
CELL AFTER WHICH TO INSERT.

TEST IF PAST.
IF YES, QUIT, H5-.

TEST IF EQUAL.
IF YES, QUIT, H5+.
IF NO, MOVE DOWN THE LIST.

SET UP CELL TO INSERT AFTER.

SET UP CELL HOLDING SUBLIST.

Q1 FIND CONNECTIVE OF SEGMENT (0).

TEST IF IT IS A CONNECTIVE.

Q2 FIND NO. OF LEVELS OF TEX (0).

CREATE LEVEL DATA TERM = ZERO.

CREATE COUNTER.
SAVE COUNTER LEVEL, TEX.

FIND MEX.
IF NONE, CLEANUP, QUIT-.
IF THERE, COUNT LEVELS.

ERASE COUNTER.
INPUT LEVEL;
IF H5-, ERASE LEVEL, QUIT-.
IF H5+, ASSIGN LEVEL 1W1 AS VALUE
OF Q2 OF TEX 1W2.
QUIT +, OUTPUT (0) IS LEVEL.
9-2 J11 J32 Q002R220
9-10C SUBPROCESS, COUNT SUBLEVELS.
9-100 11WC J12C Q002R230
CREATE COUNTER EQUAL TO THIS LEVEL.
J125 Q002R240
PRESERVE PREVIOUS COUNTER.
J50 Q002R250
40HC Q002R260
PB Q002R270
TEST IF SIMPLE VARIABLE.
7C Q002R280
IF YES, UPDATE LEVEL.
9-102 J109-100 Q002R290
J100 9-101 Q002R300
THEN QUIT + OR-
UPDATE LEVEL.
51W1 Q002R310
11WC Q002R320
J115 Q002R330
709-103 Q002R340
11WC Q002R350
11W1 Q002R360
J121 Q002R370
30HC Q002R380
TEST IF COUNTER GREATER THAN LEVEL.
9-103 J4 Q002R390
IF NO, QUIT +.
ERASE COUNTER OF THIS LEVEL.
9-101 11WC Q002R400
RESET PREVIOUS COUNTER, QUIT.
30WC J9 Q002R410
1 R Q002R420
Q002R430
G3 FIND NO. OF DISTINCT VARIABLES
Q002R440
IN TOTAL EXPRESSION (0).
40HC Q002R450
10Q3
J10
Q003R000
709-10 J6 J8
Q003R010
Q003R020
Q003R030
Q003R040
Q003R050
FIND AS VALUE IN DESC. LIST.
60WC 1W0=TEX Q003R060
1W1=FREE Q003R070
CREATE FREE LIST,
60W1 Q003R080
SAVE IT,
1W2=OUTPUT Q003R090
COUNT IT,
J126 Q003R100
MARK COUNT LOCAL,
J136 Q003R110
AND SAVE FOR OUTPUT.
60W2 Q003R120
ERASE FREE LIST.
51W1 Q003R130
CREATE BOUND LIST,
J71 Q003R140
SAVE IT,
11WC Q003R150
COUNT IT,
P31 Q003R160
60W1 1W1=BOUND Q003R170
J126 Q003R180
ADD IT TO OUTPUT DATA TERM.
40HC Q003R190
ERASE BOUND LIST.
11W2 Q003R200
ERASE BOUND COUNT.
11W2 Q003R210
10G3 Q003R220
J71 Q003R230
J9 Q003R240
11WC Q003R250
11W2 Q003R260
ASSIGN AS VALUE OF Q3.
CLEAN UP AND QUIT.
Q4 FIND NO. OF VARIABLE PLACES IN TEX (O).
FIND AS VALUE OF DESC. LIST.
  IF NONE, GO COUNT PLACES.
  IF THERE, CLEAN UP, QUIT, H5+.
SET UP THREE COPIES OF TEX NAME.
CREATE D.T. WITH VALUE = 0.
GENERATE FREE LOCATIONS FOR TALLY.
GENERATE BOUND LOCATIONS FOR TALLY.
SAVE OUTPUT D.T.
ASSIGN AS VALUE OF Q4 OF TEX.
DISCARD (O), TALLY (1).

<table>
<thead>
<tr>
<th>Q4</th>
<th>11W2</th>
<th>J11</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q5</td>
<td>10Q5</td>
<td>J10</td>
</tr>
<tr>
<td>Q6</td>
<td>10Q6</td>
<td>J10</td>
</tr>
<tr>
<td>Q7</td>
<td>10Q7</td>
<td>J10</td>
</tr>
<tr>
<td>Q8</td>
<td>10Q8</td>
<td>J10</td>
</tr>
<tr>
<td>Q9</td>
<td>10Q9</td>
<td>J10</td>
</tr>
<tr>
<td>Q10</td>
<td>10Q10</td>
<td>J10</td>
</tr>
<tr>
<td>Q11</td>
<td>10Q11</td>
<td>J10</td>
</tr>
<tr>
<td>Q12</td>
<td>10Q12</td>
<td>J10</td>
</tr>
<tr>
<td>Q13</td>
<td>10Q13</td>
<td>J10</td>
</tr>
<tr>
<td>Q14</td>
<td>10Q14</td>
<td>J10</td>
</tr>
<tr>
<td>Q15</td>
<td>1CQ15</td>
<td>J10</td>
</tr>
<tr>
<td>Q16</td>
<td>1CT10</td>
<td>J6</td>
</tr>
</tbody>
</table>

1WO=OUTPUTQ004R17C
Q17 FIND LEVEL OF SUBSEGMENT REPLACEMENT IN TEX (O).
   FIND CURRENT LEVEL.
   IF NONE,  
   FIND NUMBER OF LEVELS,
   IF NONE, QUIT -.
   COPY,
   SAVE ONE FOR OUTPUT,
   AND ASSIGN AS CURRENT LEVEL.

Q18 FIND SUFFIX OF EXPRESSION (O).
   Q18 10Q18 J10  

Q19 FIND CHARACTER SYMBOL FOR 'O'.
   Q19 10L9 J6 J10
<table>
<thead>
<tr>
<th>DATA HEADER</th>
<th>A</th>
<th>1</th>
<th>5</th>
<th>D</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>FREE VARIABLE A</td>
<td>0</td>
<td>0</td>
<td>A000D000</td>
<td>A000D010</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>A000D020</td>
<td>A000D030</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>A000D040</td>
<td>A000D050</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>A000D060</td>
<td>A000D070</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>A000D080</td>
<td>B000D000</td>
<td></td>
</tr>
<tr>
<td>FREE VARIABLE B</td>
<td>0</td>
<td>0</td>
<td>B000D010</td>
<td>B000D020</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>B000D030</td>
<td>B000D040</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>B000D050</td>
<td>B000D060</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>B000D070</td>
<td>B000D080</td>
<td></td>
</tr>
<tr>
<td>FREE VARIABLE C</td>
<td>0</td>
<td>0</td>
<td>C000D000</td>
<td>C000D010</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>C000D020</td>
<td>C000D030</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>C000D040</td>
<td>C000D050</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>C000D060</td>
<td>C000D070</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>C000D080</td>
<td>D000D000</td>
<td></td>
</tr>
<tr>
<td>FREE VARIABLE D</td>
<td>0</td>
<td>0</td>
<td>D000D010</td>
<td>D000D020</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>D000D030</td>
<td>D000D040</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>D000D050</td>
<td>D000D060</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>D000D070</td>
<td>D000D080</td>
<td></td>
</tr>
<tr>
<td>FREE VARIABLE E</td>
<td>0</td>
<td>0</td>
<td>E000D000</td>
<td>E000D010</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>E000D020</td>
<td>E000D030</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>E000D040</td>
<td>E000D050</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>E000D060</td>
<td>E000D070</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>E000D080</td>
<td>F000D000</td>
<td></td>
</tr>
<tr>
<td>FREE VARIABLE F</td>
<td>0</td>
<td>0</td>
<td>F000D010</td>
<td>F000D020</td>
<td></td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>F000D030</td>
<td>F000D040</td>
<td></td>
</tr>
</tbody>
</table>
FREE VARIABLE G

G       0
       0
       0
       0
       0
       0
       0
       0

H0 LUBRICATION.

H0       *1
       *2
       *3
       *4
       0
       0
       0
       0

IMPLIES

I       0
       0
       0
       0
       0
       0

K0 SYMBOL FOR CHARACTER K.

K       0
       0
       0
       0
       0

HOLDS 'OR'

K1       V0
       0

HOLDS 'NOT'

K2       -0
       0

HOLDS 'AND'

K3       *0
       0

HOLDS 'PROVEN EQUIVALENCE'

K4       =0
       0

HOLDS 'DEFINITIONAL EQUIVALENCE'

K5       =1
       0

HOLDS 'IMPLIES'

K6       10
       0

K7 HOLDS CONNECTIVE DELIMITER.

K7       *
       0

K10 PREVIOUS PROBLEM NUMBER.

K10       01
       0

SUBSTITUTION COUNT

K11       +1
       0

EFFORT BASE (AND TOTAL).

K12       +1
       0

LIMIT ON NO. OF SUBPROBLEMS

K20       +1
       100

LIMIT ON NO. OF SUBSTITUTIONS

K21       +1
       100

LIMIT ON EFFORT

K22       +1
       100
       0000

K30 F

K31 DON'T PRINT REJECTS IF HOLDS NO

K31       NO
       0

K41 VALUE = METHOD COLUMN.

K41       01
       10

K42 VALUE = NAME COLUMN.

K42       01
       40

K43 VALUE = EXPRESSION COLUMN.

K43       01
       50

K44 VALUE = 'LIMIT' COLUMN.

K44       01
       20

K45 VALUE = 'ACTUAL' COLUMN.

K45       01
       40

K46 VALUE = 'REJECTED' COLUMN.

K46       01
       20
K47 VALUE = NAME OF NEW SUBPROBLEM
K48 VALUE = THM, METHOD COLUMN.
K51 DATA TERM '.(' K52 DATA TERM ')'
K53 DATA TERM ' ' K54 DATA TERM ' ' L0 SYMBOL FOR CHARACTER L.
L1 TRUE THEOREMS AXIOMS DEFINITIONS
L2 LIST OF FREE VARIABLES
L3 PROBLEM LIST FOR MULTI PROB EXEC
L4 TRUE EXPRESSIONS MAPS
L5 LIST DESCRIBED BY L4
L6 LIST OF METHODS FOR ORIG PROBS
L7 LIST OF METHODS FOR PROBLEMS.
L8 DESCRIPTION LIST TABLE OF
DELIMITABLE EXTERNAL CONNECTIVES
L9 DESCRIPTION LIST TABLE OF
CHARACTER SYMBOLS FOR
READING TEXT.
UNTIRIED PROBLEMS LIST
L10 L11 M
0 0 C Q7 +21M
0 0 C Q7 +21N
0 0 C Q7 0

1 INTEGER CONSTANTS.
N1 +01 1 N2 +01 2 N3 +01 3 N4 +01 4 N5 +01 5
N6 +01 6 N7 +01 7 N8 +01 8 N9 +01 9 N10 +01 0

0 SYMBOL FOR CHARACTER O.
0 0 C Q7 0

VARIABLE P
P 0 0 Q5 Q5 Q9 Q9 Q7 +21P

VARIABLE Q
Q 0 0 Q5

L009D150 L009D160 L009D170 L009D180 L009D190 L009D200 L009D210 L009D220 L009D230 L009D240 L009D250 L009D260 L010D000 L011D000 M00D000 M00D010 M00D020 M00D030 M00D040 N00C000 C Q7 0 N00C010 C Q7 0 N00C020 C Q7 0 N00C030 C Q7 0 N00C040 C Q7 0 N00D000 C Q7 0 N00D010 C Q7 0 N00D020 C Q7 0 N00D030 C Q7 0 N00D040 P00D000 P00D010 P00D020 P00D030 P00D033 P00D037 P00D040 P00D050 P00D060 Q00D000 Q00D010 Q00D020
VARIABLE R

R
  0
  0
Q5
Q5
Q9
Q9
Q7
Q7
0
+21Q
Q000D030
Q000D033
Q000D037
Q000D040
Q000D050
Q000D060
R000D000
R000D010
R000D020
R000D030
R000D033
R000D037
R000D040
R000D050
R000D060

VARIABLE S

S
  0
  0
Q5
Q5
Q9
Q9
Q7
Q7
0
+21R
S000D000
S000D010
S000D020
S000D030
S000D033
S000D037
S000D040
S000D050
S000D060

VARIABLE T

T
  0
  0
Q5
Q5
Q9
Q9
Q7
Q7
0
+21S
T000D000
T000D010
T000D020
T000D030
T000D033
T000D037
T000D040
T000D050
T000D060

T1 'GIVEN'

T1
  0
  0
  0
21GIVEN
T001D000
T001D010

T2 'PROOF FOUND.'

T2
  0
  0
9-1
9-2
9-3
0
21PROO
T022D000
T022D010
T022D020
T022D030

T3 'SUBSTITUTION'

T3
  0
  0
9-1
9-2
9-3
0
21ND
T03D000
T03D010
T03D020
T03D030

T4 'Q.E.D.'

T4
  0
  0
21SUBST
T03D040

21TUTI
T03D050

21ON
T03D060

T04D000
T5 LIST OF ONE BLANK D.T.

T6 'NO PROOF FOUND'

T7 'EFFORT'

T8 'SUBPROBLEMS'

T9 'SUBSTITUTIONS'

T10 DESC. LIST TABLE OF NAMES.

T12 'DETACHMENT'
| T13 'REPLACEMENT' | 9-2 21MENT | T012040 |
|                  | 9-1 21REPLA | T013000 |
|                  | 9-2 21CEMEN | T013010 |
|                  | 9-3 21T      | T013020 |
|                  | 9-4 0        | T013030 |
| T14 'FORWARD CHAINING' | 9-1 21FORWA | T013040 |
|                  | 9-2 21RD CH  | T013050 |
|                  | 9-3 21AININ  | T013060 |
|                  | 9-4 21G      | T013070 |
|                  | 9-1 0        | T014000 |
|                  | 9-2 0        | T014010 |
|                  | 9-3 0        | T014020 |
|                  | 9-4 0        | T014030 |
| T15 'BACKWARD CHAINING' | 9-1 21BACKW | T014040 |
|                  | 9-2 21ARD C  | T014050 |
|                  | 9-3 21HAINI  | T014060 |
|                  | 9-4 21NG     | T014070 |
|                  | 9-1 0        | T015000 |
|                  | 9-2 0        | T015010 |
|                  | 9-3 0        | T015020 |
|                  | 9-4 0        | T015030 |
| T16 'SUBLEVEL REPLACEMENT' | 9-1 21SUBLE | T015040 |
|                  | 9-2 21VEL R  | T015050 |
|                  | 9-3 21EPLAC  | T015060 |
|                  | 9-4 21EMENT  | T015070 |
|                  | 9-1 0        | T016000 |
|                  | 9-2 0        | T016010 |
|                  | 9-3 0        | T016020 |
|                  | 9-4 0        | T016030 |
| T19 'REJECTED PROBLEM' | 9-1 21REJEC | T016040 |
|                  | 9-2 21TED P  | T016050 |
|                  | 9-3 21ROBLE  | T016060 |
|                  | 9-4 21M      | T016070 |
|                  | 9-1 0        | T019000 |
|                  | 9-2 0        | T019010 |
|                  | 9-3 0        | T019020 |
|                  | 9-4 0        | T019030 |
| T20 'ACTUAL'     | 9-1 21ACTUA | T019040 |
|                  | 9-2 21L      | T019050 |
|                  | 9-1 0        | T020000 |
|                  | 9-2 0        | T020010 |
|                  | 9-1 21L      | T020020 |
|                  | 9-2 21L      | T020030 |
| T21 'LIMIT'      | 9-1 0        | T021000 |
|                  | 9-2 0        | T021010 |
T22 'TO PROVE'

<table>
<thead>
<tr>
<th></th>
<th>T22</th>
<th>T22</th>
<th>T22</th>
<th>T22</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LIMIT</td>
<td>LIMIT</td>
<td>LIMIT</td>
<td>LIMIT</td>
</tr>
<tr>
<td></td>
<td>T021D020</td>
<td>T022D000</td>
<td>T022D010</td>
<td>T022D020</td>
</tr>
<tr>
<td>9-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-2</td>
<td>21TO PR</td>
<td>21TO PR</td>
<td>21TO PR</td>
<td>21TO PR</td>
</tr>
<tr>
<td>9-3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-4</td>
<td>21OVE</td>
<td>21OVE</td>
<td>21OVE</td>
<td>21OVE</td>
</tr>
<tr>
<td>9-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

T23 'REMEMBER PROVED THEOREM'

<table>
<thead>
<tr>
<th></th>
<th>T23</th>
<th>T23</th>
<th>T23</th>
<th>T23</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MEM</td>
<td>MEM</td>
<td>MEM</td>
<td>MEM</td>
</tr>
<tr>
<td></td>
<td>T023D000</td>
<td>T023D010</td>
<td>T023D020</td>
<td>T023D030</td>
</tr>
<tr>
<td>9-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-2</td>
<td>21RE</td>
<td>21RE</td>
<td>21RE</td>
<td>21RE</td>
</tr>
<tr>
<td>9-3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-4</td>
<td>21MEMBE</td>
<td>21MEMBE</td>
<td>21MEMBE</td>
<td>21MEMBE</td>
</tr>
<tr>
<td>9-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

T24 'BAD EXPRESSION'

<table>
<thead>
<tr>
<th></th>
<th>T24</th>
<th>T24</th>
<th>T24</th>
<th>T24</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>EXP</td>
<td>EXP</td>
<td>EXP</td>
<td>EXP</td>
</tr>
<tr>
<td></td>
<td>T024D000</td>
<td>T024D010</td>
<td>T024D020</td>
<td>T024D030</td>
</tr>
<tr>
<td>9-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-2</td>
<td>21BAD E</td>
<td>21BAD E</td>
<td>21BAD E</td>
<td>21BAD E</td>
</tr>
<tr>
<td>9-3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-4</td>
<td>21XPR</td>
<td>21XPR</td>
<td>21XPR</td>
<td>21XPR</td>
</tr>
<tr>
<td>9-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

UC SYMBOL FOR CHARACTER U.

<table>
<thead>
<tr>
<th></th>
<th>U</th>
<th>U</th>
<th>U</th>
<th>U</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>T000D000</td>
<td>T000D010</td>
<td>T000D020</td>
<td>T000D030</td>
</tr>
<tr>
<td>9-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-4</td>
<td>+21U</td>
<td>+21U</td>
<td>+21U</td>
<td>+21U</td>
</tr>
<tr>
<td>9-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

OR

<table>
<thead>
<tr>
<th></th>
<th>V</th>
<th>V</th>
<th>V</th>
<th>V</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>T000D000</td>
<td>T000D010</td>
<td>T000D020</td>
<td>T000D030</td>
</tr>
<tr>
<td>9-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-4</td>
<td>+21V</td>
<td>+21V</td>
<td>+21V</td>
<td>+21V</td>
</tr>
<tr>
<td>9-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

XC SYMBOL FOR CHARACTER X.

<table>
<thead>
<tr>
<th></th>
<th>X</th>
<th>X</th>
<th>X</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>T000D000</td>
<td>T000D010</td>
<td>T000D020</td>
<td>T000D030</td>
</tr>
<tr>
<td>9-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-4</td>
<td>+21X</td>
<td>+21X</td>
<td>+21X</td>
<td>+21X</td>
</tr>
<tr>
<td>9-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

YC SYMBOL FOR CHARACTER Y.

<table>
<thead>
<tr>
<th></th>
<th>Y</th>
<th>Y</th>
<th>Y</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>T000D000</td>
<td>T000D010</td>
<td>T000D020</td>
<td>T000D030</td>
</tr>
<tr>
<td>9-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-4</td>
<td>+21Y</td>
<td>+21Y</td>
<td>+21Y</td>
<td>+21Y</td>
</tr>
<tr>
<td>9-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

ZC SYMBOL FOR CHARACTER Z.

<table>
<thead>
<tr>
<th></th>
<th>Z</th>
<th>Z</th>
<th>Z</th>
<th>Z</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>C</td>
<td>C</td>
<td>C</td>
<td>C</td>
</tr>
<tr>
<td></td>
<td>T000D000</td>
<td>T000D010</td>
<td>T000D020</td>
<td>T000D030</td>
</tr>
<tr>
<td>9-1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-3</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-4</td>
<td>+21Z</td>
<td>+21Z</td>
<td>+21Z</td>
<td>+21Z</td>
</tr>
<tr>
<td>9-5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9-6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
NOT
- +21Z 0 20000030
   0 20000040
   0 20000000
   0 20000010
   Q14 -00000020
   J3 -00000030
   Q7 -00000040

AND
* +21- 0 *00000000
   0 *00000010
   Q14 *00000020
   J4 *00000030
   Q7 *00000040

PROVEN EQUIVALENCE
= +21* 0 *00000050
   0 *00000060
   Q14 *00000000
   J4 *00000010
   Q7 *00000020

=1 DEFINITIONAL EQUIVALENCE
=1 +21= 0 =00000000
   0 =00000010
   Q14 =00000020
   J4 =00000030
   Q7 =00000040

+0 SYMBOL FOR PLUS SIGN.
+ +21+ 0 +00000000
   0 +00000010
   Q7 +00000020

/0 SYMBOL FOR SLASH.
/ +21/ 0 /00000000
   0 /00000010
   Q7 /00000020
   /00000030

/1 SYMBOL FOR DIGIT 1.
/1 +21/ 0 /00000000
   0 /00000010
   Q7 /00000020
   /00000030

/2 SYMBOL FOR DIGIT 2.
/2 +211 0 /00000000
   0 /00000010
   Q7 /00000020
   /00000030

/3 SYMBOL FOR DIGIT 3.
/3 +212 0 /00000000
   0 /00000010
<table>
<thead>
<tr>
<th>Symbol for Digit</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>0</td>
</tr>
<tr>
<td>22</td>
<td>0</td>
</tr>
<tr>
<td>23</td>
<td>0</td>
</tr>
<tr>
<td>24</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>26</td>
<td>0</td>
</tr>
<tr>
<td>27</td>
<td>0</td>
</tr>
<tr>
<td>28</td>
<td>0</td>
</tr>
<tr>
<td>29</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>31</td>
<td>0</td>
</tr>
<tr>
<td>32</td>
<td>0</td>
</tr>
<tr>
<td>33</td>
<td>0</td>
</tr>
<tr>
<td>34</td>
<td>0</td>
</tr>
<tr>
<td>35</td>
<td>0</td>
</tr>
<tr>
<td>36</td>
<td>0</td>
</tr>
<tr>
<td>37</td>
<td>0</td>
</tr>
<tr>
<td>38</td>
<td>0</td>
</tr>
<tr>
<td>39</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>41</td>
<td>0</td>
</tr>
<tr>
<td>42</td>
<td>0</td>
</tr>
<tr>
<td>43</td>
<td>0</td>
</tr>
<tr>
<td>44</td>
<td>0</td>
</tr>
<tr>
<td>45</td>
<td>0</td>
</tr>
<tr>
<td>46</td>
<td>0</td>
</tr>
<tr>
<td>47</td>
<td>0</td>
</tr>
<tr>
<td>48</td>
<td>0</td>
</tr>
<tr>
<td>49</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>51</td>
<td>0</td>
</tr>
<tr>
<td>52</td>
<td>0</td>
</tr>
<tr>
<td>53</td>
<td>0</td>
</tr>
<tr>
<td>54</td>
<td>0</td>
</tr>
<tr>
<td>55</td>
<td>0</td>
</tr>
<tr>
<td>56</td>
<td>0</td>
</tr>
<tr>
<td>57</td>
<td>0</td>
</tr>
<tr>
<td>58</td>
<td>0</td>
</tr>
<tr>
<td>59</td>
<td>0</td>
</tr>
<tr>
<td>60</td>
<td>0</td>
</tr>
<tr>
<td>61</td>
<td>0</td>
</tr>
<tr>
<td>62</td>
<td>0</td>
</tr>
<tr>
<td>63</td>
<td>0</td>
</tr>
<tr>
<td>64</td>
<td>0</td>
</tr>
<tr>
<td>65</td>
<td>0</td>
</tr>
<tr>
<td>66</td>
<td>0</td>
</tr>
<tr>
<td>67</td>
<td>0</td>
</tr>
<tr>
<td>68</td>
<td>0</td>
</tr>
<tr>
<td>69</td>
<td>0</td>
</tr>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>71</td>
<td>0</td>
</tr>
<tr>
<td>72</td>
<td>0</td>
</tr>
<tr>
<td>73</td>
<td>0</td>
</tr>
<tr>
<td>74</td>
<td>0</td>
</tr>
<tr>
<td>75</td>
<td>0</td>
</tr>
<tr>
<td>76</td>
<td>0</td>
</tr>
<tr>
<td>77</td>
<td>0</td>
</tr>
<tr>
<td>78</td>
<td>0</td>
</tr>
<tr>
<td>79</td>
<td>0</td>
</tr>
<tr>
<td>80</td>
<td>0</td>
</tr>
<tr>
<td>81</td>
<td>0</td>
</tr>
<tr>
<td>82</td>
<td>0</td>
</tr>
<tr>
<td>83</td>
<td>0</td>
</tr>
<tr>
<td>84</td>
<td>0</td>
</tr>
<tr>
<td>85</td>
<td>0</td>
</tr>
<tr>
<td>86</td>
<td>0</td>
</tr>
<tr>
<td>87</td>
<td>0</td>
</tr>
<tr>
<td>88</td>
<td>0</td>
</tr>
<tr>
<td>89</td>
<td>0</td>
</tr>
<tr>
<td>90</td>
<td>0</td>
</tr>
<tr>
<td>91</td>
<td>0</td>
</tr>
<tr>
<td>92</td>
<td>0</td>
</tr>
<tr>
<td>93</td>
<td>0</td>
</tr>
<tr>
<td>94</td>
<td>0</td>
</tr>
<tr>
<td>95</td>
<td>0</td>
</tr>
<tr>
<td>96</td>
<td>0</td>
</tr>
<tr>
<td>97</td>
<td>0</td>
</tr>
<tr>
<td>98</td>
<td>0</td>
</tr>
<tr>
<td>99</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>0</td>
</tr>
<tr>
<td>101</td>
<td>0</td>
</tr>
<tr>
<td>102</td>
<td>0</td>
</tr>
<tr>
<td>103</td>
<td>0</td>
</tr>
<tr>
<td>104</td>
<td>0</td>
</tr>
<tr>
<td>105</td>
<td>0</td>
</tr>
<tr>
<td>106</td>
<td>0</td>
</tr>
<tr>
<td>107</td>
<td>0</td>
</tr>
<tr>
<td>108</td>
<td>0</td>
</tr>
<tr>
<td>109</td>
<td>0</td>
</tr>
<tr>
<td>110</td>
<td>0</td>
</tr>
<tr>
<td>111</td>
<td>0</td>
</tr>
<tr>
<td>112</td>
<td>0</td>
</tr>
<tr>
<td>113</td>
<td>0</td>
</tr>
<tr>
<td>114</td>
<td>0</td>
</tr>
<tr>
<td>115</td>
<td>0</td>
</tr>
<tr>
<td>116</td>
<td>0</td>
</tr>
<tr>
<td>117</td>
<td>0</td>
</tr>
<tr>
<td>118</td>
<td>0</td>
</tr>
<tr>
<td>119</td>
<td>0</td>
</tr>
<tr>
<td>120</td>
<td>0</td>
</tr>
<tr>
<td>121</td>
<td>0</td>
</tr>
<tr>
<td>122</td>
<td>0</td>
</tr>
<tr>
<td>123</td>
<td>0</td>
</tr>
<tr>
<td>124</td>
<td>0</td>
</tr>
<tr>
<td>125</td>
<td>0</td>
</tr>
<tr>
<td>126</td>
<td>0</td>
</tr>
<tr>
<td>127</td>
<td>0</td>
</tr>
<tr>
<td>128</td>
<td>0</td>
</tr>
<tr>
<td>129</td>
<td>0</td>
</tr>
<tr>
<td>130</td>
<td>0</td>
</tr>
<tr>
<td>131</td>
<td>0</td>
</tr>
<tr>
<td>132</td>
<td>0</td>
</tr>
<tr>
<td>133</td>
<td>0</td>
</tr>
<tr>
<td>134</td>
<td>0</td>
</tr>
<tr>
<td>135</td>
<td>0</td>
</tr>
<tr>
<td>136</td>
<td>0</td>
</tr>
<tr>
<td>137</td>
<td>0</td>
</tr>
<tr>
<td>138</td>
<td>0</td>
</tr>
<tr>
<td>139</td>
<td>0</td>
</tr>
<tr>
<td>140</td>
<td>0</td>
</tr>
<tr>
<td>141</td>
<td>0</td>
</tr>
<tr>
<td>142</td>
<td>0</td>
</tr>
<tr>
<td>143</td>
<td>0</td>
</tr>
<tr>
<td>144</td>
<td>0</td>
</tr>
<tr>
<td>145</td>
<td>0</td>
</tr>
<tr>
<td>146</td>
<td>0</td>
</tr>
<tr>
<td>147</td>
<td>0</td>
</tr>
<tr>
<td>148</td>
<td>0</td>
</tr>
<tr>
<td>149</td>
<td>0</td>
</tr>
<tr>
<td>150</td>
<td>0</td>
</tr>
<tr>
<td>151</td>
<td>0</td>
</tr>
<tr>
<td>152</td>
<td>0</td>
</tr>
<tr>
<td>153</td>
<td>0</td>
</tr>
<tr>
<td>154</td>
<td>0</td>
</tr>
<tr>
<td>155</td>
<td>0</td>
</tr>
<tr>
<td>156</td>
<td>0</td>
</tr>
<tr>
<td>157</td>
<td>0</td>
</tr>
<tr>
<td>158</td>
<td>0</td>
</tr>
<tr>
<td>159</td>
<td>0</td>
</tr>
<tr>
<td>160</td>
<td>0</td>
</tr>
<tr>
<td>161</td>
<td>0</td>
</tr>
<tr>
<td>162</td>
<td>0</td>
</tr>
<tr>
<td>163</td>
<td>0</td>
</tr>
<tr>
<td>164</td>
<td>0</td>
</tr>
<tr>
<td>165</td>
<td>0</td>
</tr>
<tr>
<td>166</td>
<td>0</td>
</tr>
<tr>
<td>167</td>
<td>0</td>
</tr>
<tr>
<td>168</td>
<td>0</td>
</tr>
<tr>
<td>169</td>
<td>0</td>
</tr>
<tr>
<td>170</td>
<td>0</td>
</tr>
<tr>
<td>171</td>
<td>0</td>
</tr>
<tr>
<td>172</td>
<td>0</td>
</tr>
<tr>
<td>173</td>
<td>0</td>
</tr>
<tr>
<td>174</td>
<td>0</td>
</tr>
<tr>
<td>175</td>
<td>0</td>
</tr>
<tr>
<td>176</td>
<td>0</td>
</tr>
<tr>
<td>177</td>
<td>0</td>
</tr>
<tr>
<td>178</td>
<td>0</td>
</tr>
<tr>
<td>179</td>
<td>0</td>
</tr>
<tr>
<td>180</td>
<td>0</td>
</tr>
</tbody>
</table>
/14 DUMMY CHARACTER SYMBOL

/16 DUMMY EXPRESSION -- 'DEFINITIONS'.

EXTERNAL NAME
CONNECTIVE 'I'.

DUMMY VARIABLE 'DEFIN'.

EXTERNAL NAME
DUMMY VARIABLE 'TIONS'.

EXTERNAL NAME.
(0 SYMBOL FOR LEFT PAREN.

(0 SYMBOL FOR QUOTE MARK.

(0 SYMBOL FOR RIGHT PAREN.

(0 SYMBOL FOR COMMA.
.0 SYMBOL FOR PERIOD

$0 SYMBOL FOR DOLLAR SIGN.
EXECUTIVE HEADER

1

X1 11W26
10X23
J73
J76
50X21
10J147
J100
10X22
10J148
J100

R

X001R000
X001R010
X001R020
X001R030
X001R040
X001R050
X001R060
X001R070
X001R080
X001R090
X001R100
X001R110
X001R120
X001R130
X001R140
X001R150
X001R160
X001R170
X001R180
X001R190
X001R200
X001R210
X001R215
X001R220
X001R230
X001R240
X001R250
X001R260
X001R270
X001R280
X001R290
X001R300
X001R310
X001R320
X001R330
X001R340
X009R000
X009R010
X010R000
X010R010
X010R020
X010R030
X010R040
X011R000
X011R010

SET UP TRAPS.

MARK TO TRACE.

MARK TO PROPAGATE TRACE.

GET NEXT TRUE TEX FROM

INPUT UNIT IF ANY LEFT.

CONVERT TO TREE FORM.

ADD TO SET OF TRUE EXPRESSION.

TAKE ACTION; TRY FOR ANOTHER

SKIP TWO LINES.

GET NEXT PROBLEM TEX.

IF NO MORE, GO TRY PROOFS.

CONVERT TO INTERNAL.

PRINT EXPRESSION.

ADD TO LIST OF PROBLEMS.

BAD INPUT ACTION.

BAD INPUT ACTION.

X10 INVOKE FULL TRACE.

X11 REVOKE CURRENT TRACE MODE

IF ANOTHER EXISTS.
X14 SAVE FOR RESTART ON INTERRUPT
    USING UNIT 3.
SAVE AND SET H5+.
    IF H5+, HALT.
    IF H5-, CONTINUE.

X19 MONITOR POINT FORCER.
SAVE FOR RESTART
REFERENCES


