

MEMORANDUM
RM-3731-CC
JUNE 1963

**THE LOGIC THEORY MACHINE:
A MODEL HEURISTIC PROGRAM**

Einar Stefferud

PREPARED FOR:
CARNEGIE CORPORATION

The **RAND** *Corporation*
SANTA MONICA • CALIFORNIA

MEMORANDUM

RM-3731-CC

JUNE 1963

**THE LOGIC THEORY MACHINE:
A MODEL HEURISTIC PROGRAM**

Einar Stefferud

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.

The **RAND** Corporation
SANTA MONICA • CALIFORNIA

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 unjustifiably 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.[†]

[†]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

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

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

THE 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:

-	intuitive meaning:	NOT
V	intuitive meaning:	OR
I	intuitive meaning:	IMPLIES
*	intuitive meaning:	AND
=	intuitive meaning:	EQUIVALENT TO
.=.	intuitive meaning:	EQUIVALENT TO BY DEFINITION

- 2) Axioms (expressions given as true):

- *1.2 [AVA]IA
- *1.3 AI[BVA]
- *1.4 [AVB]I[BVA]
- *1.5 [AV[BVC]]I[BV[AVC]]
- *1.6 [AIB]I[[CVA]I[CVB]]

- 3) Definitions (expressions defining connectives):

- *1.01 [AIB].=.[AVB]DEF.
- *3.01 [A*B].=.-[-AV-B]DEF.
- *4.01 [A=B].=.[[AIB]*[BIA]]DEF.

[†]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
 PI[PVP] Substitution
*1.2 [AVA]IA Given
 [PVP]IP Substitution
*2.08 PIP Chaining
 Q.E.D.

Chaining uses the two expressions PI[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
Description list	{	9-0	0	
		Q15		Attr. "tree form"
Main Expression	{	Q15		
		Q7		Attr. "external name"
Left Expression	{	9-2	0	Value is data term containing text
		21*1.5	I0	Main connective
Right Expression	{	9-1	9-3	Left subexpression
		9-4	0	Right subexpression
Right of left	{	9-3	V0	Connective OR
		A0		Variable A
Right of Right	{	9-4	9-5	0
		V0		
		9-5	V0	Variable B
		9-6	0	
		9-6	V0	
			B0	Variable C
			C0	0
			A0	
			C0	0 Variable C

Variables and connectives look like this:

Connective	I0	9-0	0
IMPLIES	9-0	0	
Attr. "type"		Q14	
Value "non-unary"		J4	
		Q7	
		9-1	0
	9-1	21I	
Connective OR	V0	9-0	0
	9-0	0	
		Q14	
		J4	
		Q7	
		9-1	0
	9-1	21V	
Free Variable "A"	A0	9-0	0
	9-0	0	
Attr. "Variable"		Q5	
		Q5	
Attr. "Free Var."		Q6	
		Q6	
		Q7	
		9-1	0
	9-1	21A	
Free Variable "B"	B0	9-0	0
	9-0	0	
		Q5	
		Q5	
		Q6	
		Q6	
		Q7	
		9-1	0
	9-1	21B	
Free Variable "C"	C0	9-0	0
	9-0	Q5	
		Q5	
		Q6	
		Q6	
		Q7	
		9-1	0
	9-1	21C	

The rules of inference are embodied in routines called methods:

M11	Detachment
M12	Substitution
M13	Replacement
M14	Forward Chaining
M15	Backward Chaining
M16	Subexpression Replacement I
M17	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 subproblem 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 subproblem AIB."

The main heuristic[†] 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"[‡] 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.

[†] By heuristic, we mean, "Any principle or device that contributes to the reduction in the average search to solution." (9)

[‡] 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

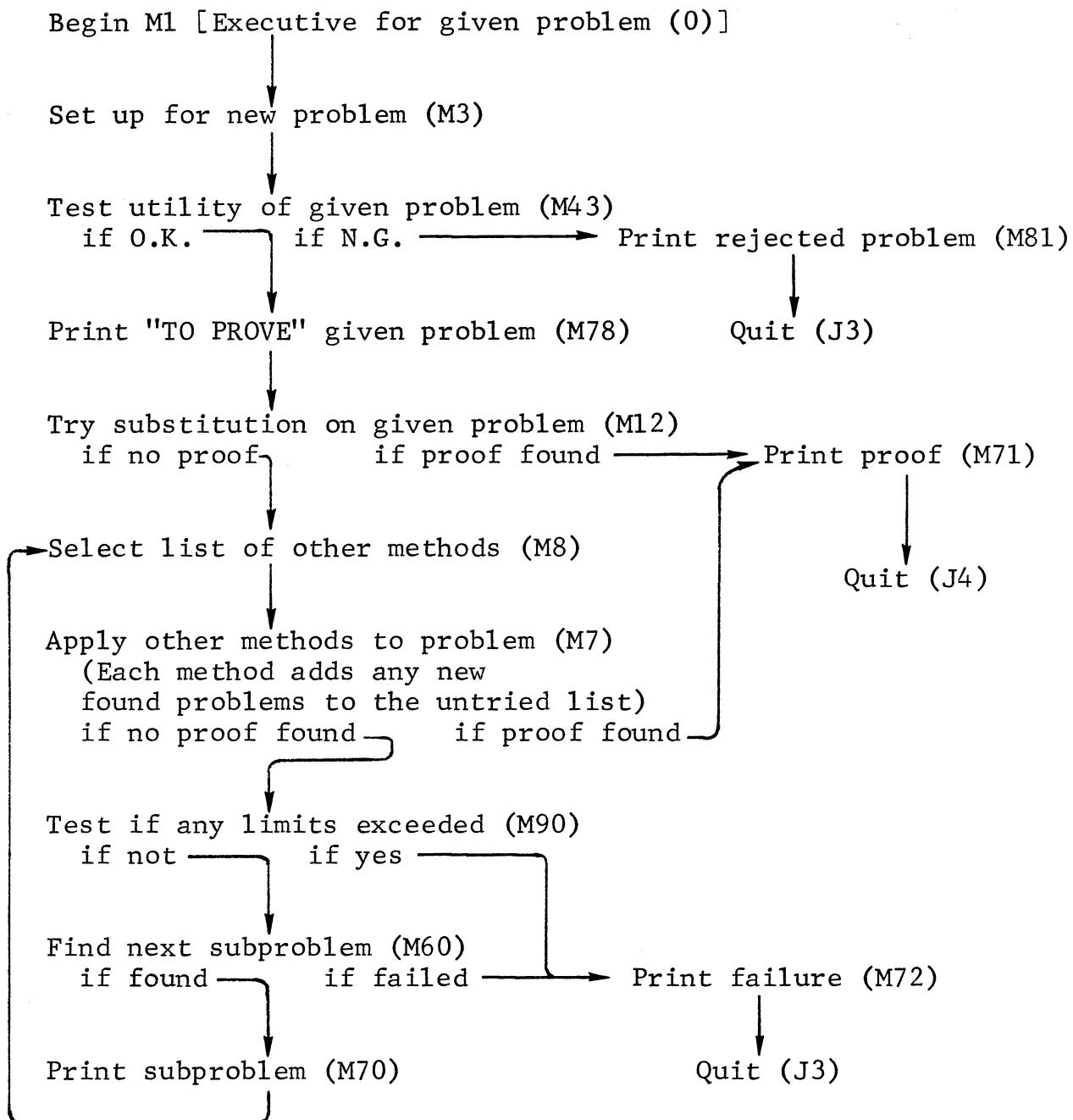


Fig. 1--The Single-Problem Executive

TO PROVE:

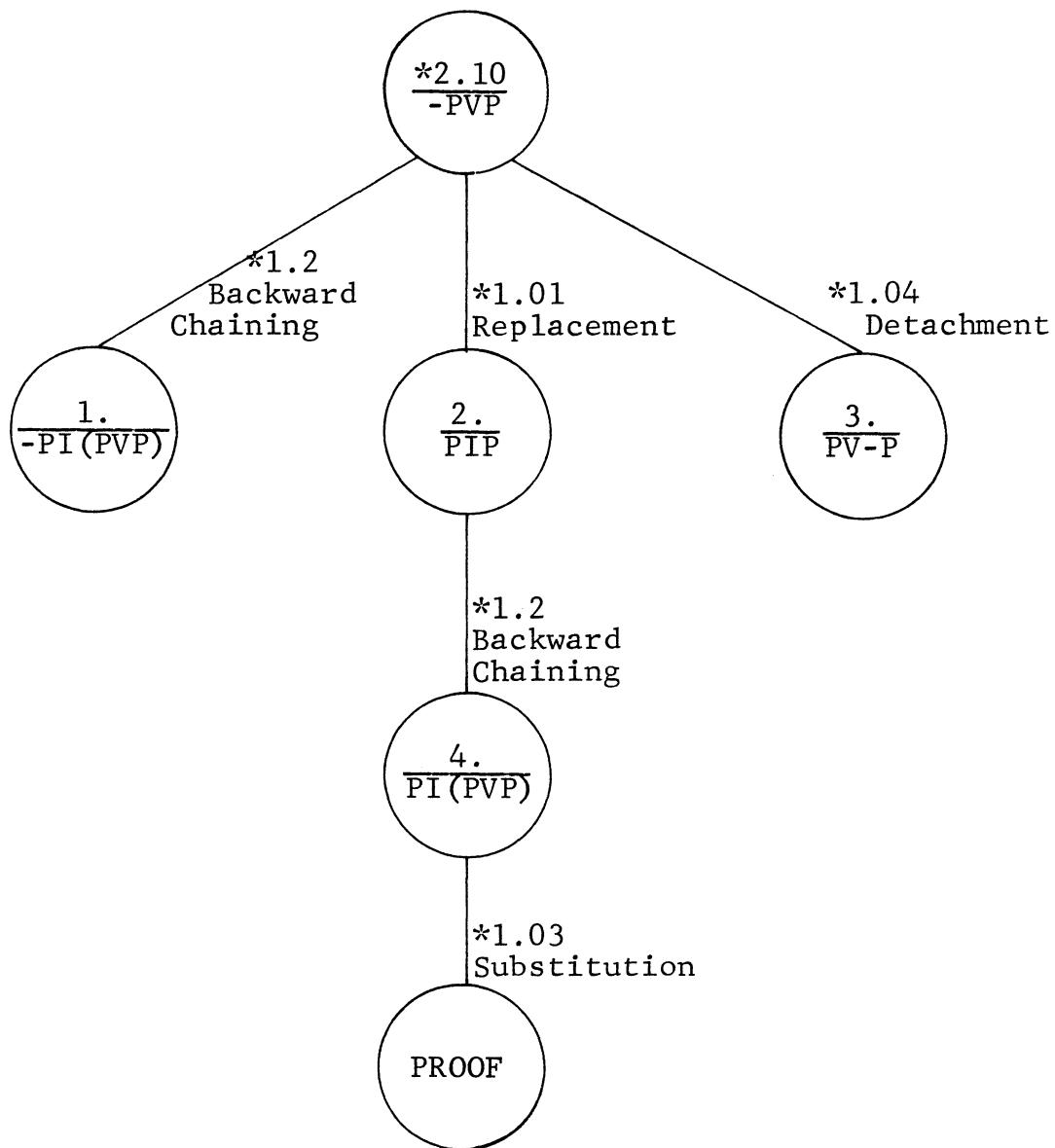


Fig. 2--Sample Subproblem Tree

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.

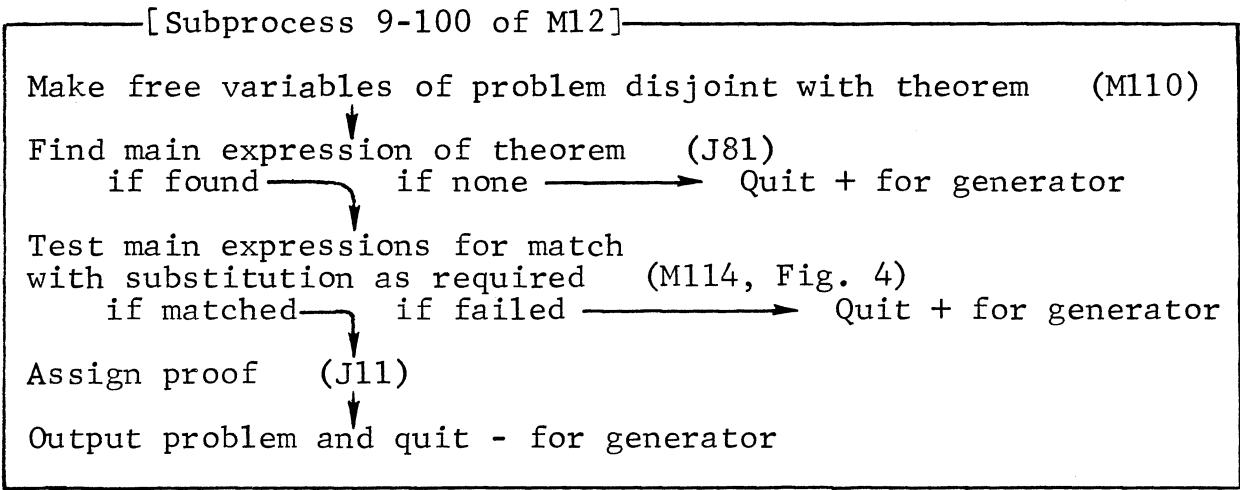
It may seem confusing, but substitution method M12 does not actually do any substitution. It only acts as an executive for matching a given problem with a sequence of appropriate theorems until a match is found or the theorem supply is exhausted. Determination of required substitutions is made by the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

Test for match routine, M114, is also an executive. It "puts off the work" to match process M111 (Fig. 5). M114's only reason for existence is to discard the output of M111 if a match is successful. Remember, TEST routines should leave no outputs in H0.

M111 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)



↓
Erase feasibles list (J71)
↓
Clear context (J33)
↓
Reverse H5 from generator and quit M12 (J5)

Fig. 3--The Substitution Method

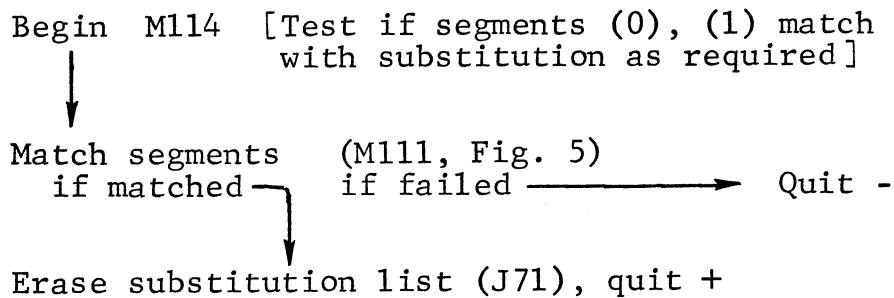


Fig. 4--Test for Match with Substitution

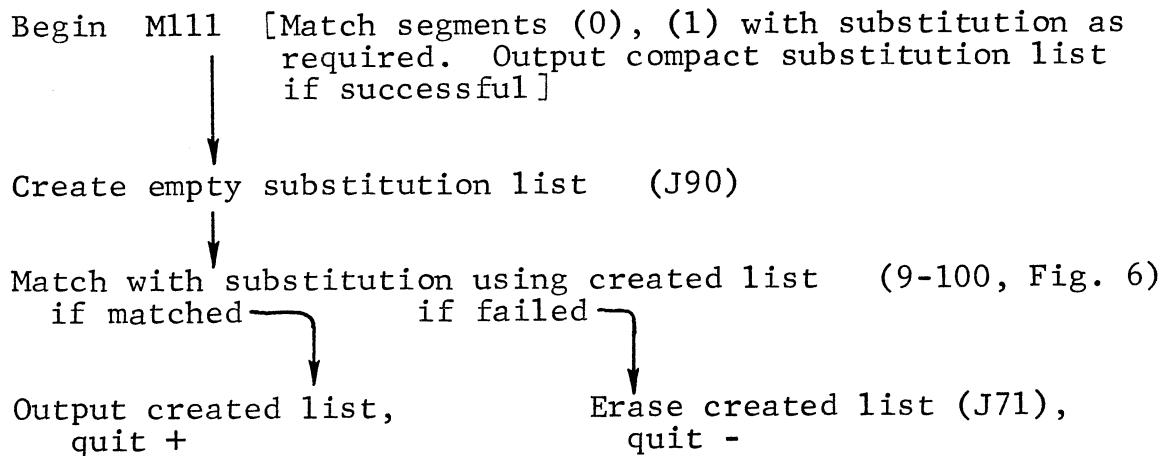


Fig. 5--The Match Process

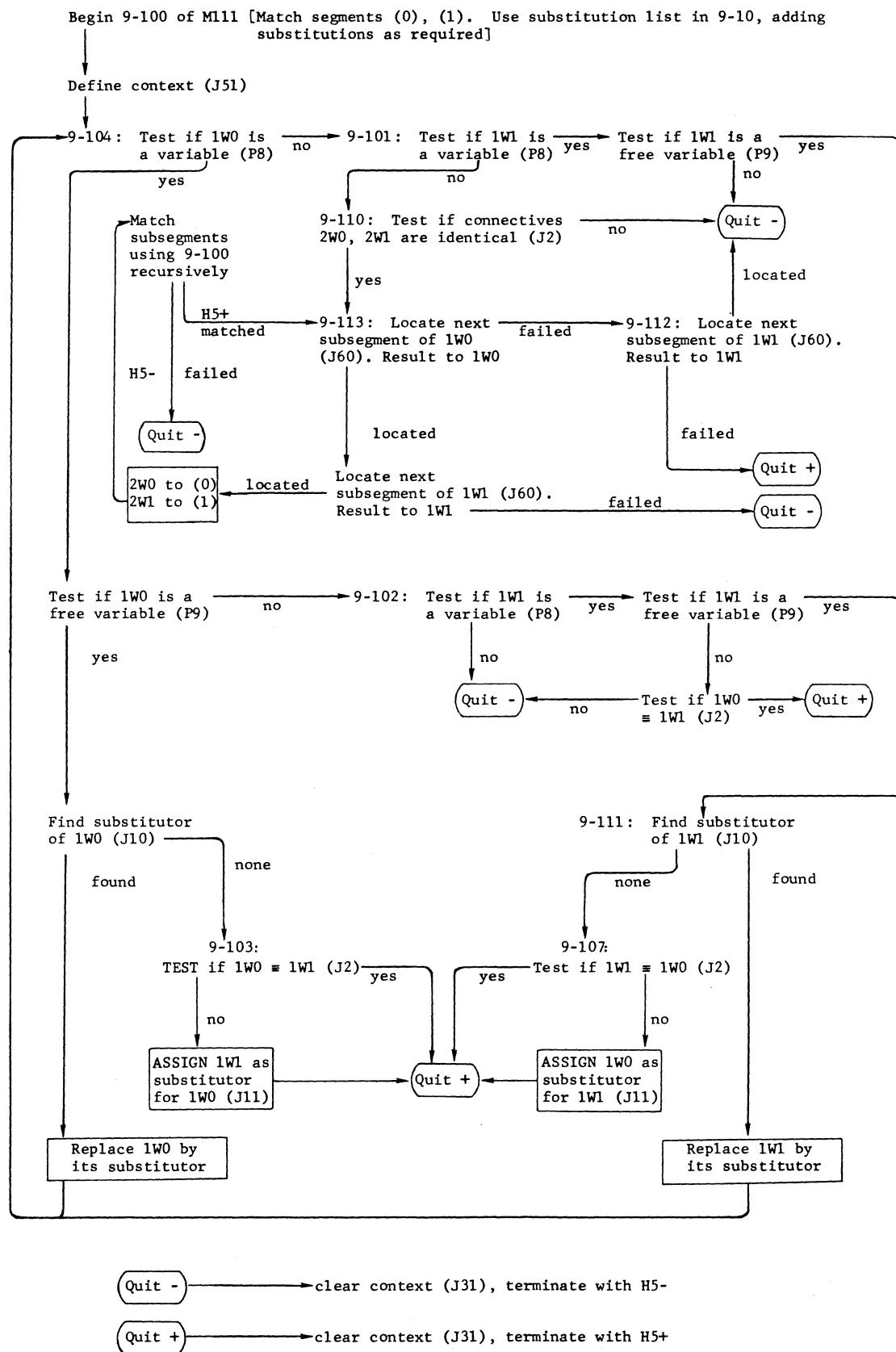


Fig. 6--The Match Subprocess

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 M111 is called a compact substitution list.

Figure 6 shows how M111 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.

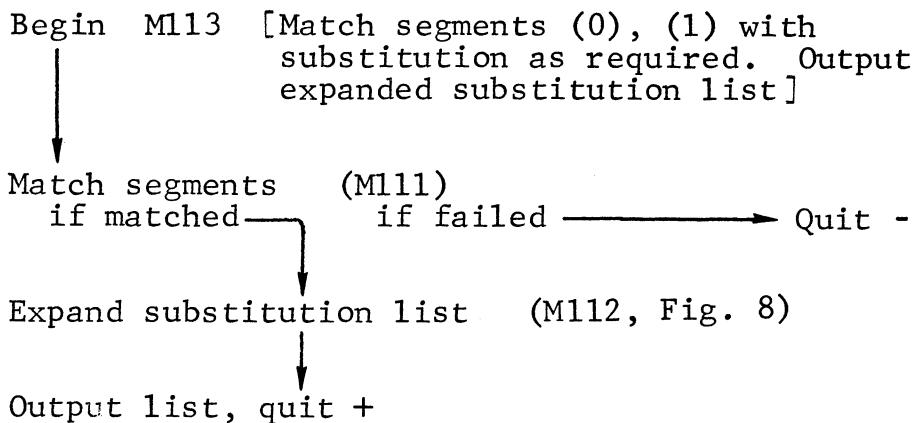


Fig. 7--Match with Substitution and Output
Expanded Substitution List

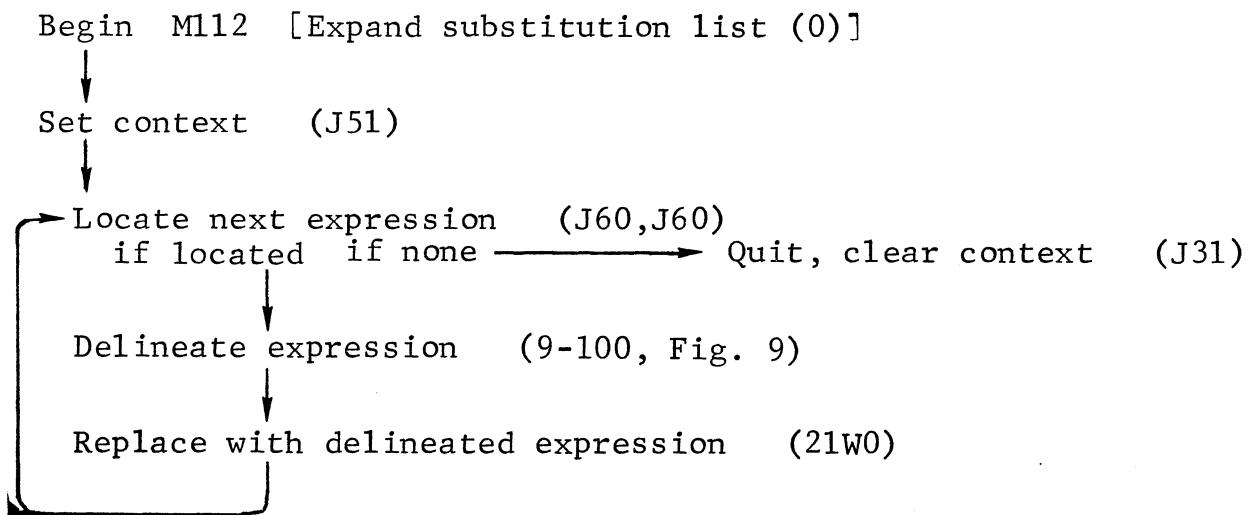


Fig. 8--Expand Substitution List

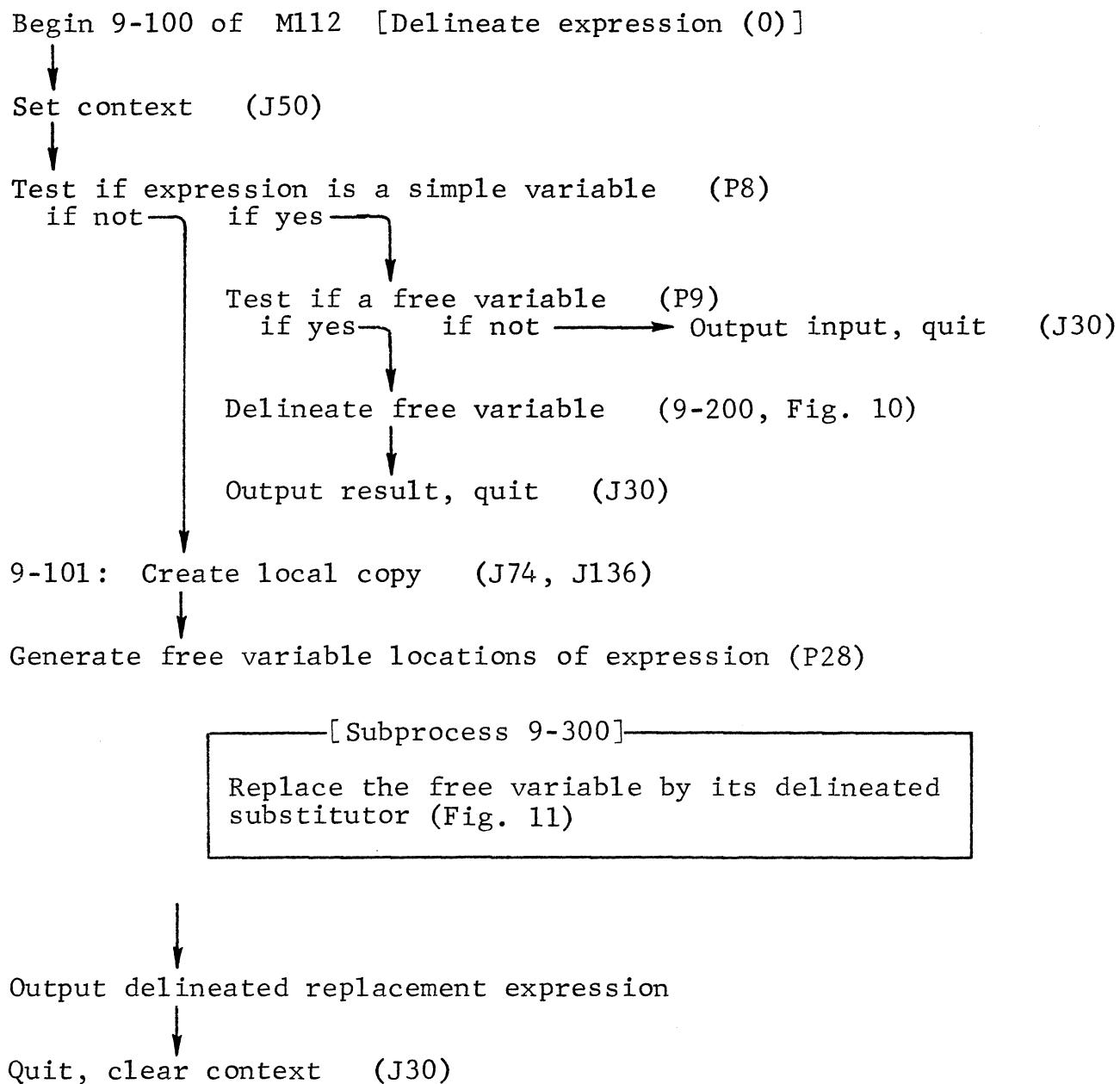


Fig. 9--Delineate Expression

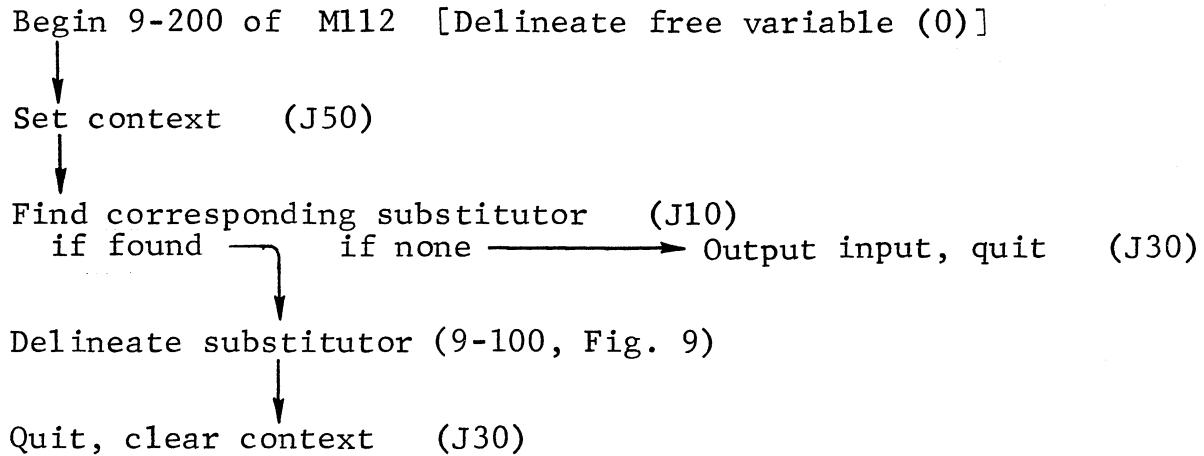


Fig. 10--Delineate Free Variable

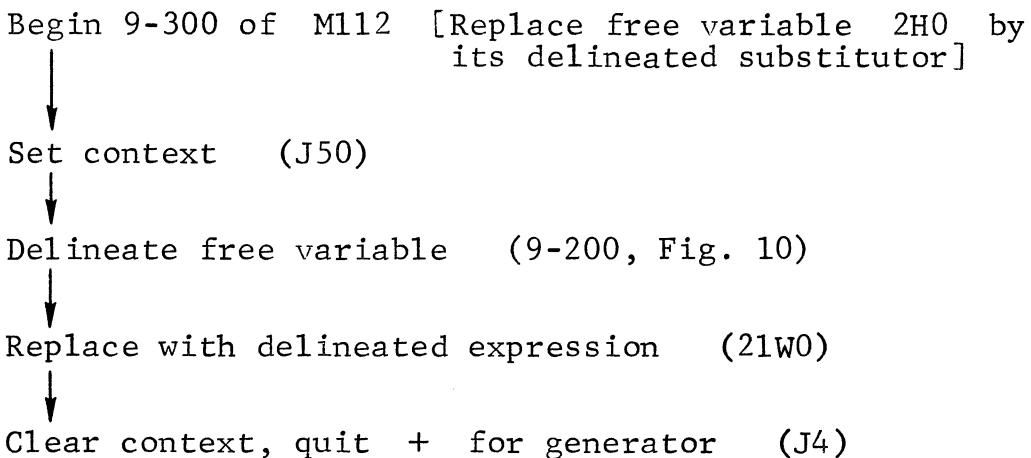


Fig. 11--Replace Free Variable by Delineated Substitutor

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 -PVP and axiom [AVB]I[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 \therefore).

Problem PIP and definition [AIB] \therefore [-AVB]
yield subproblem -PVP.

Problem -PVP and definition [AIB] \therefore [-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.

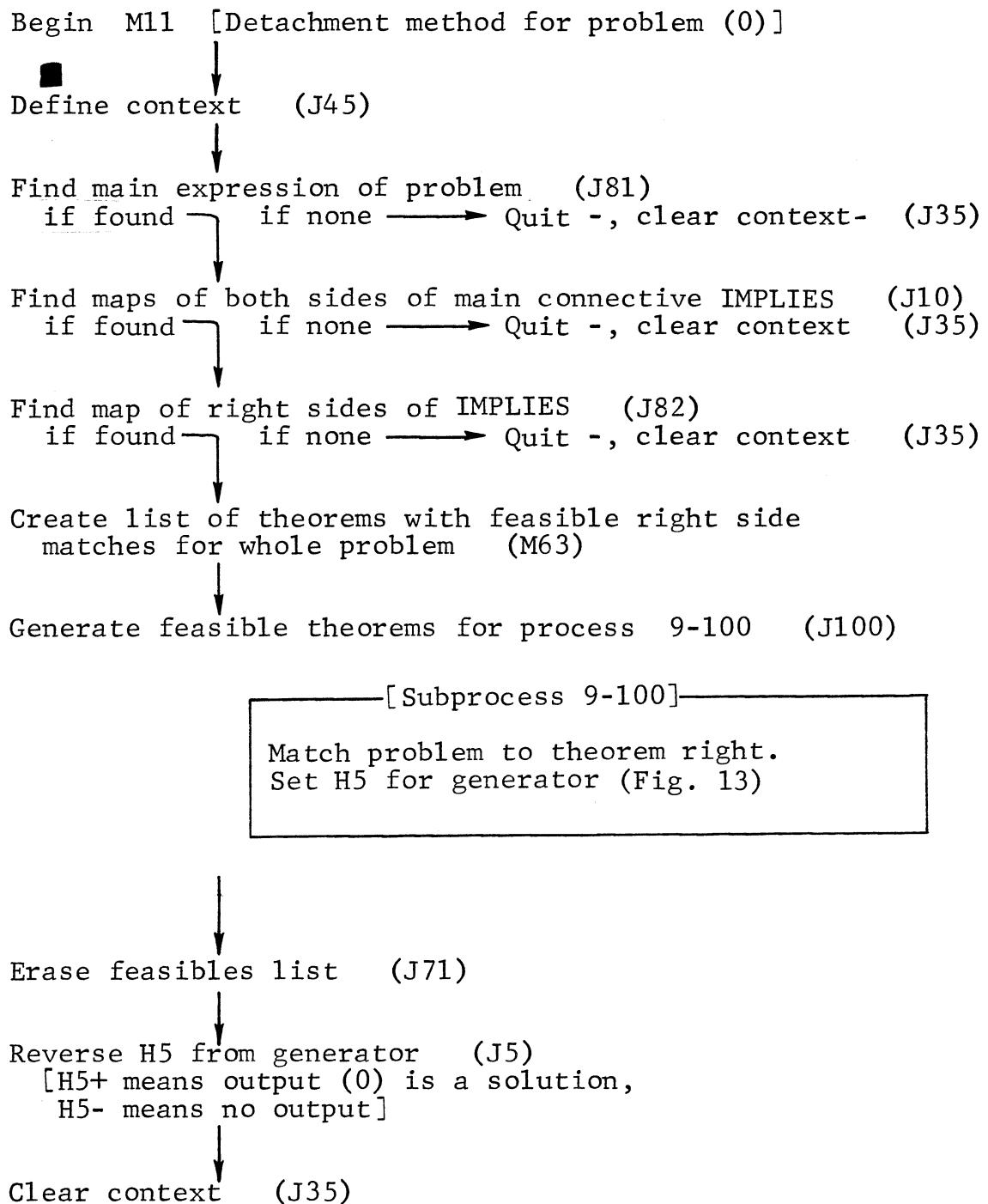


Fig. 12--Detachment Method

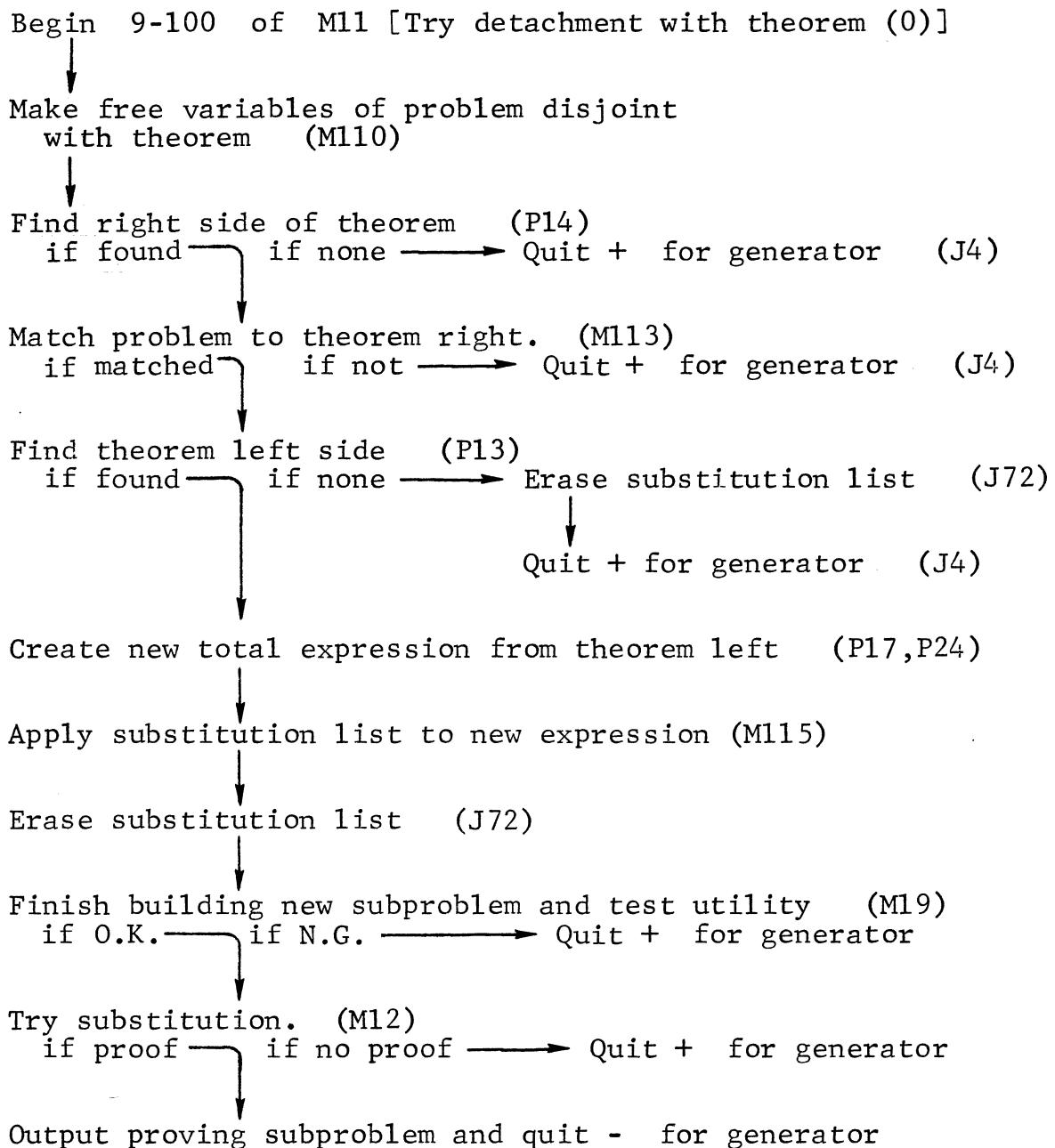


Fig. 13--Detachment Method Subprocess

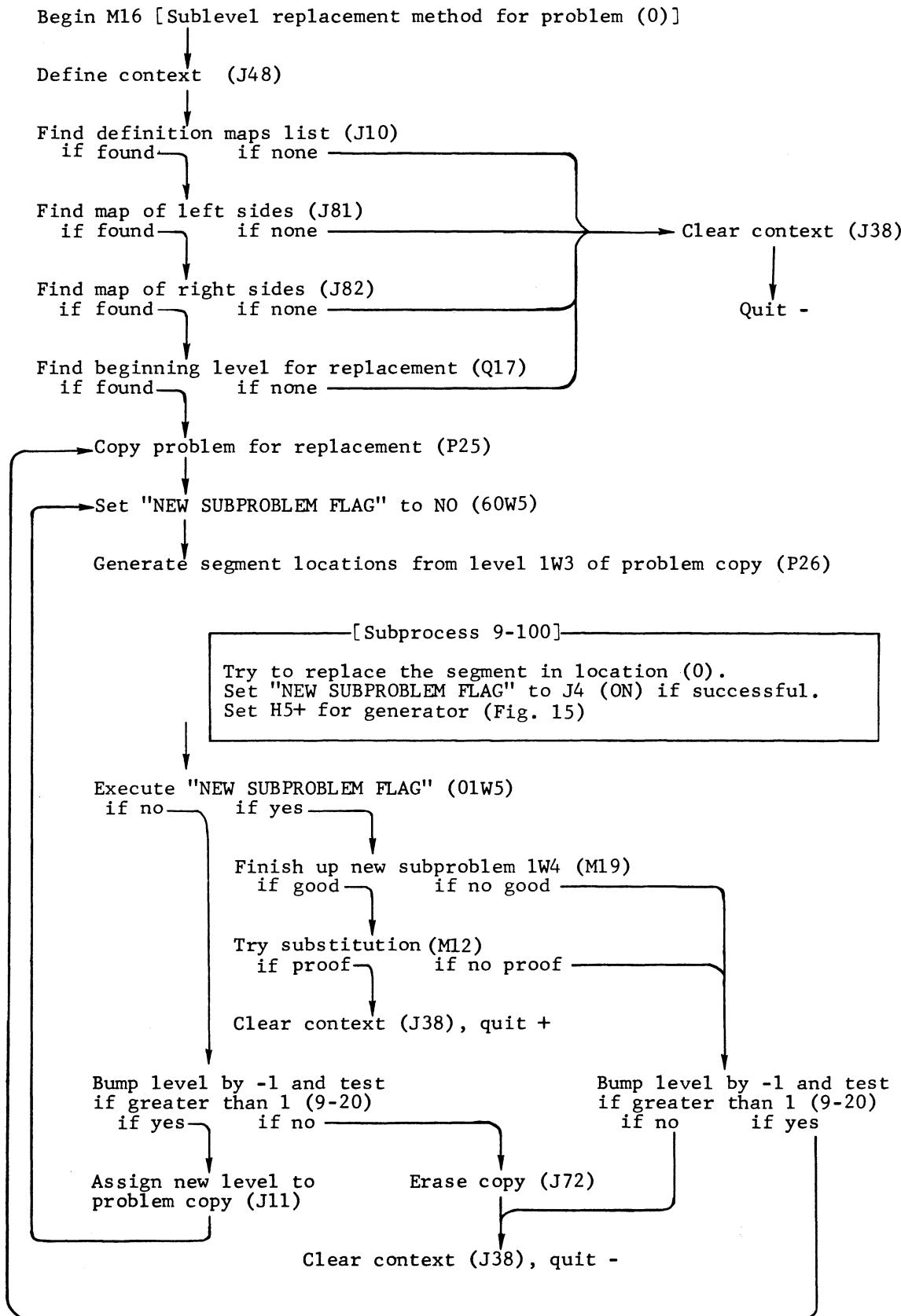


Fig. 14--Sublevel Replacement Method I

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)

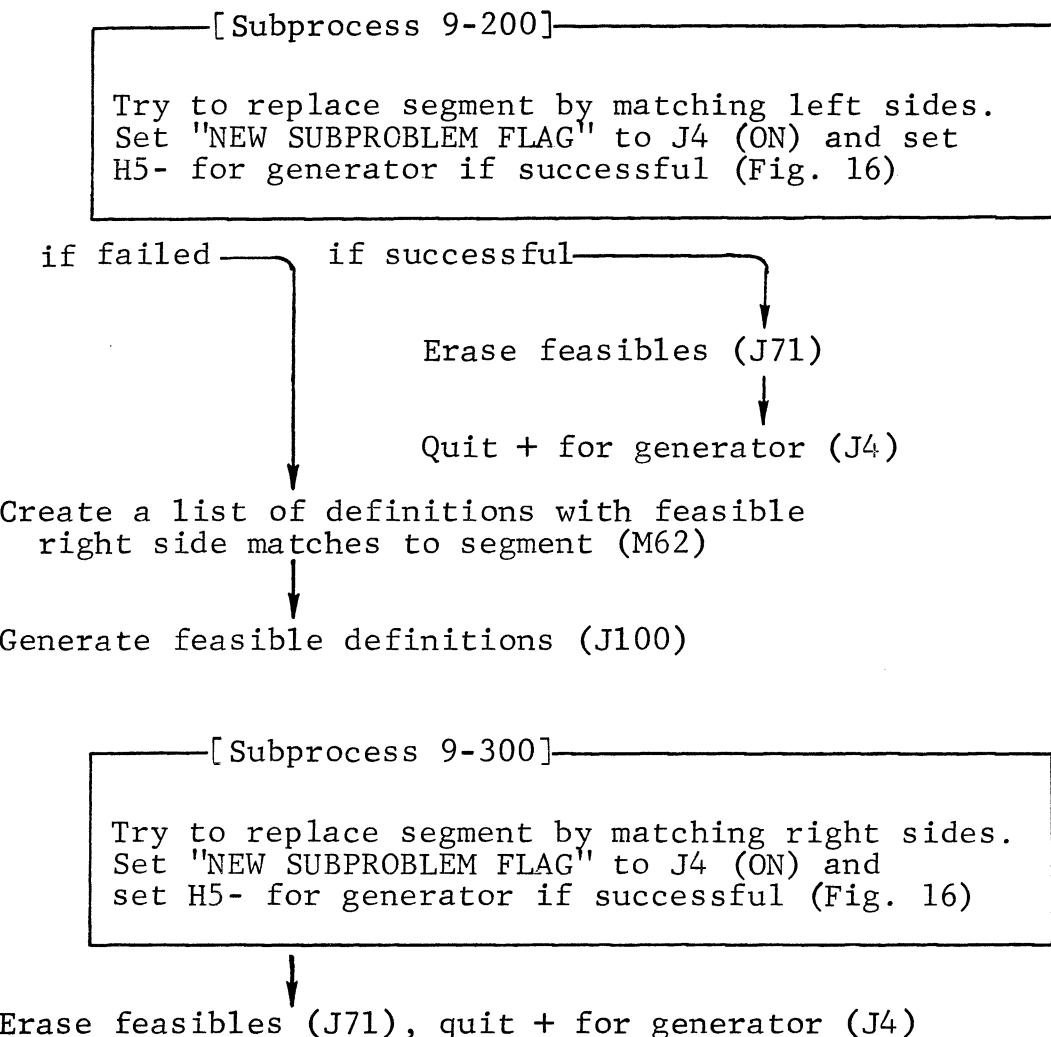


Fig. 15--Try to Replace Located Segment

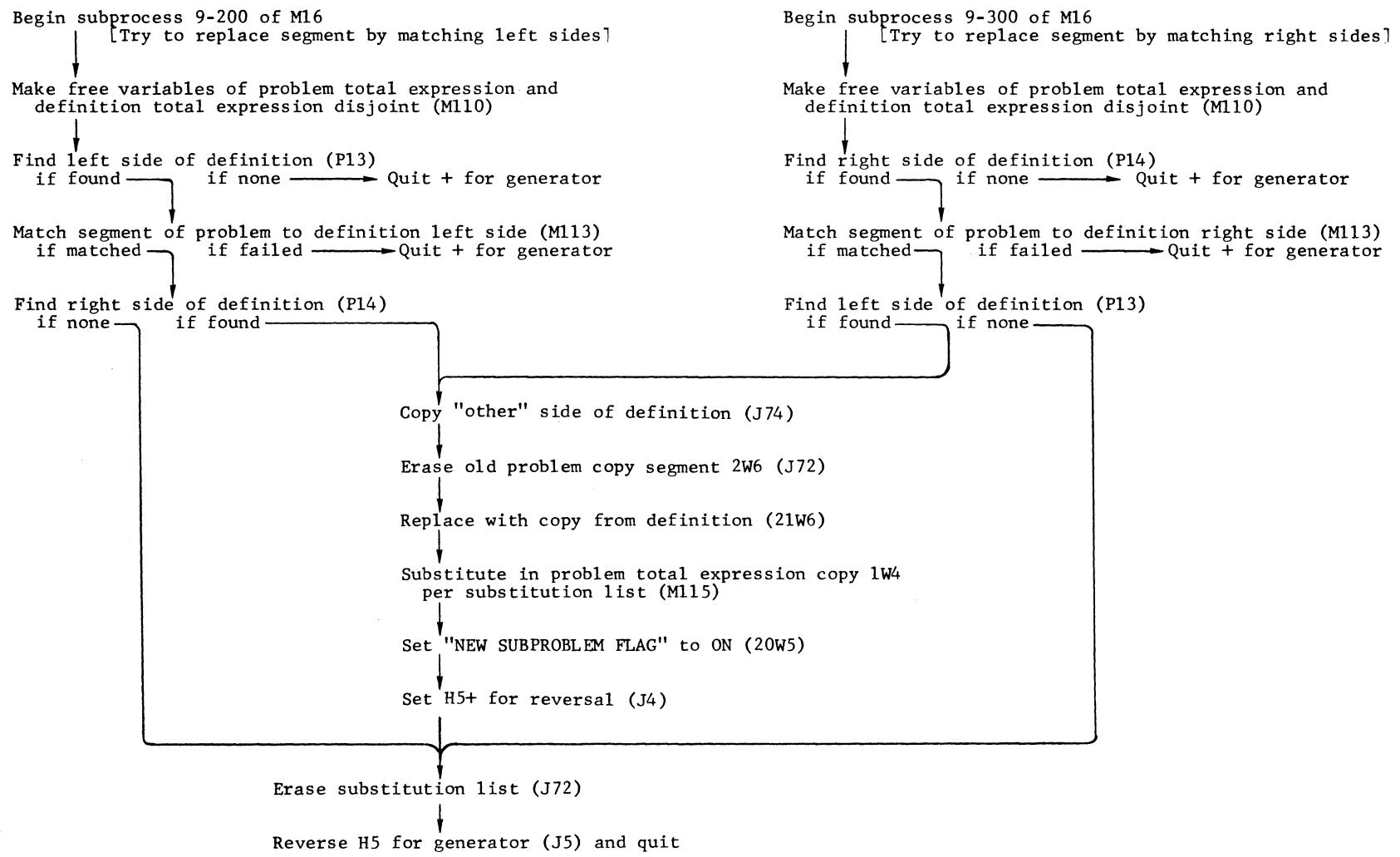


Fig. 16--Try to Replace Segment by Matching Left (Right) Sides

Problem PI-PI-P and definition AIB.=.[-AVB] yield subproblem [-PV-P]I-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 M113.

Problem PV---P and theorem AV-A yield new subproblem -PI---P.

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

Problem PI[PVQ] and theorem [AVB]I[BVA] yield new subproblem PI[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.

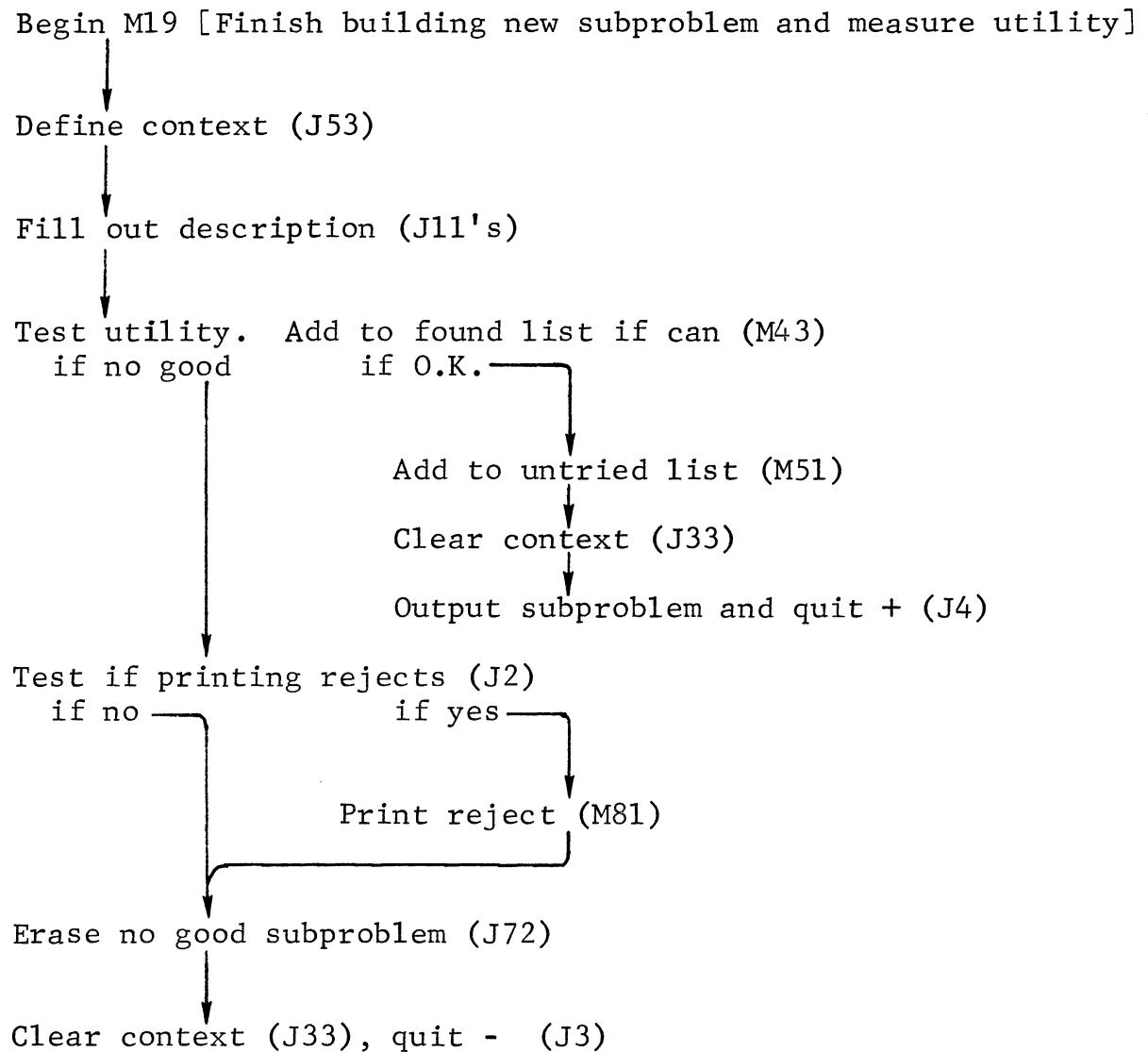


Fig. 17--Finish Building New Subproblem

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 , $\neg P$), problems with matching sub-segments of main connective OR ($P \vee P$, $[PIQ]V[PIQ]$), 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.

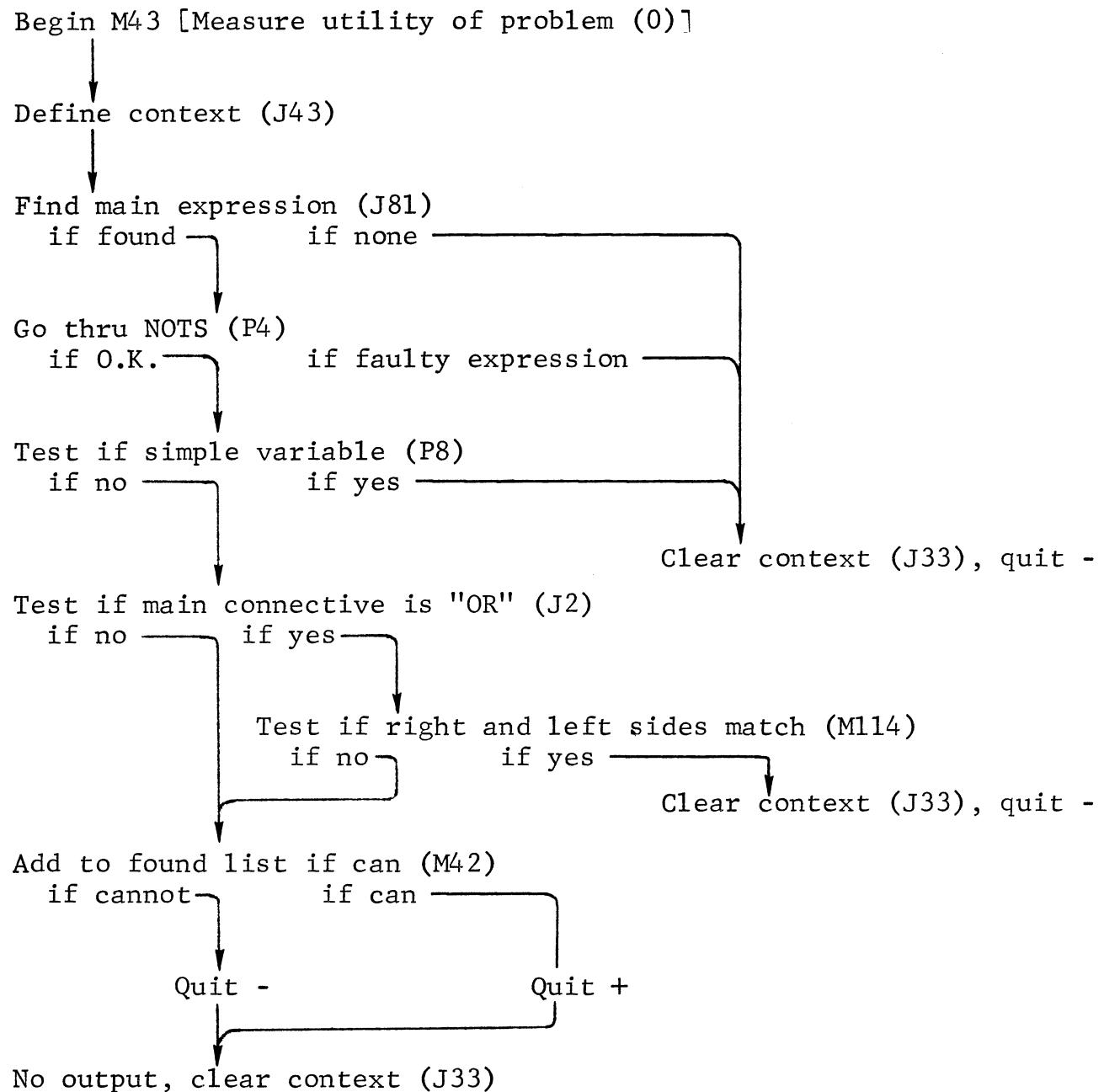


Fig. 18--Measure Utility

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

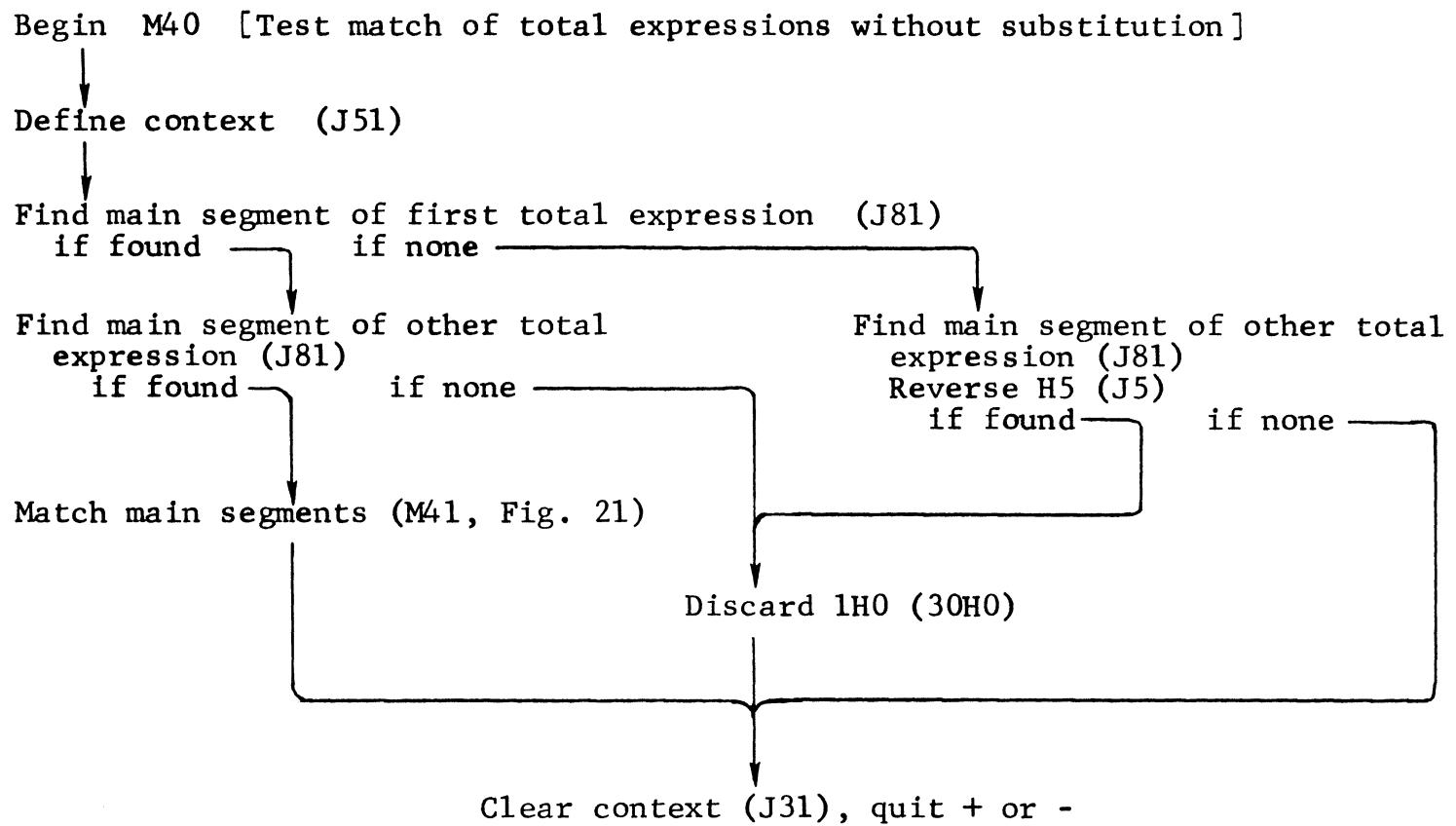


Fig. 20--Test Match of Total Expressions Without Substitution

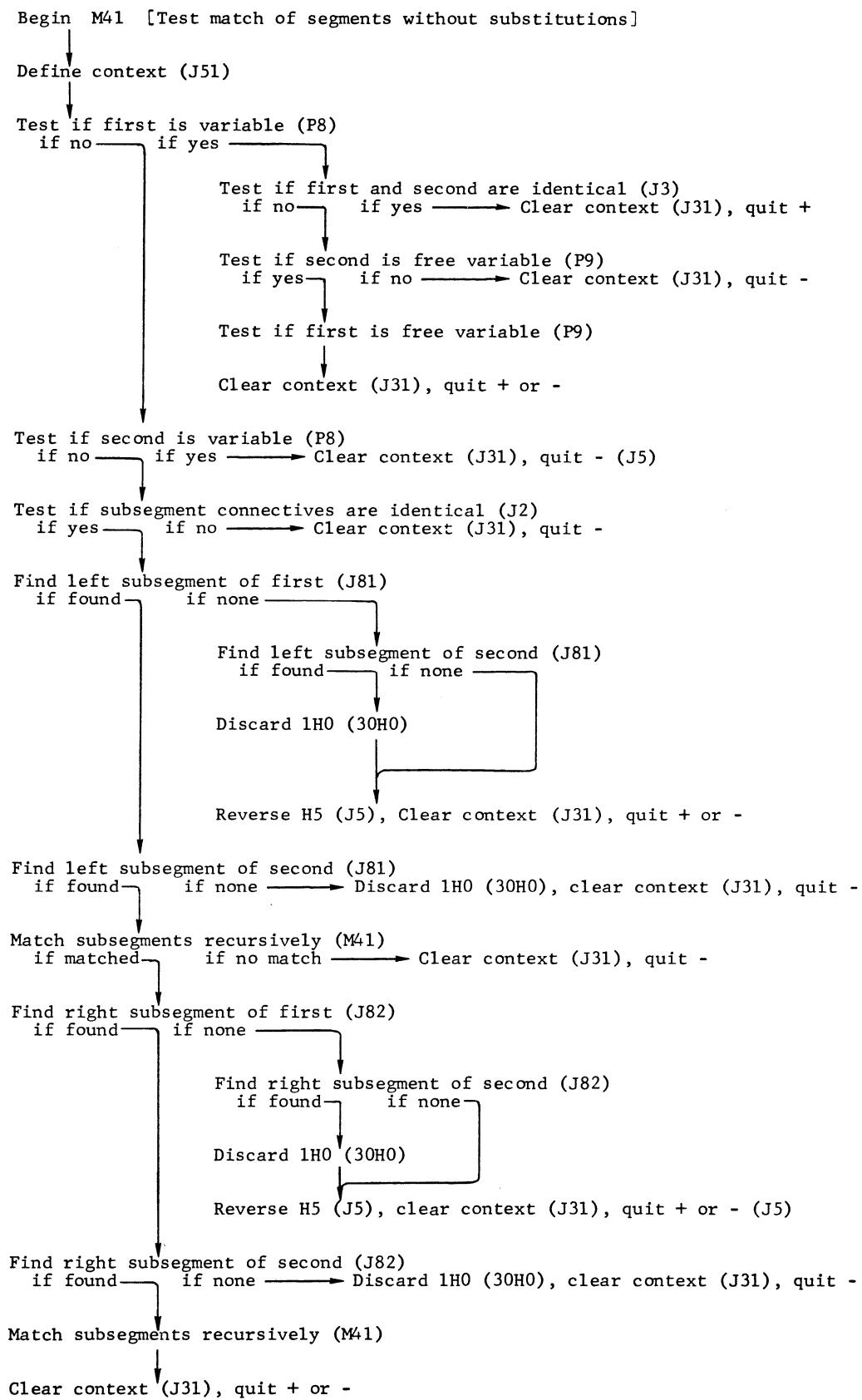


Fig. 21--Test Match of Segments Without Substitution

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 sub-segment 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]IA
*1.3 AI[BVA]
*1.4 [AVB]I[BVA] ;

but not for Axioms

*1.5 [AV[BVC]]I[BV[AVC]]
*1.6 [AIB]I[[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.

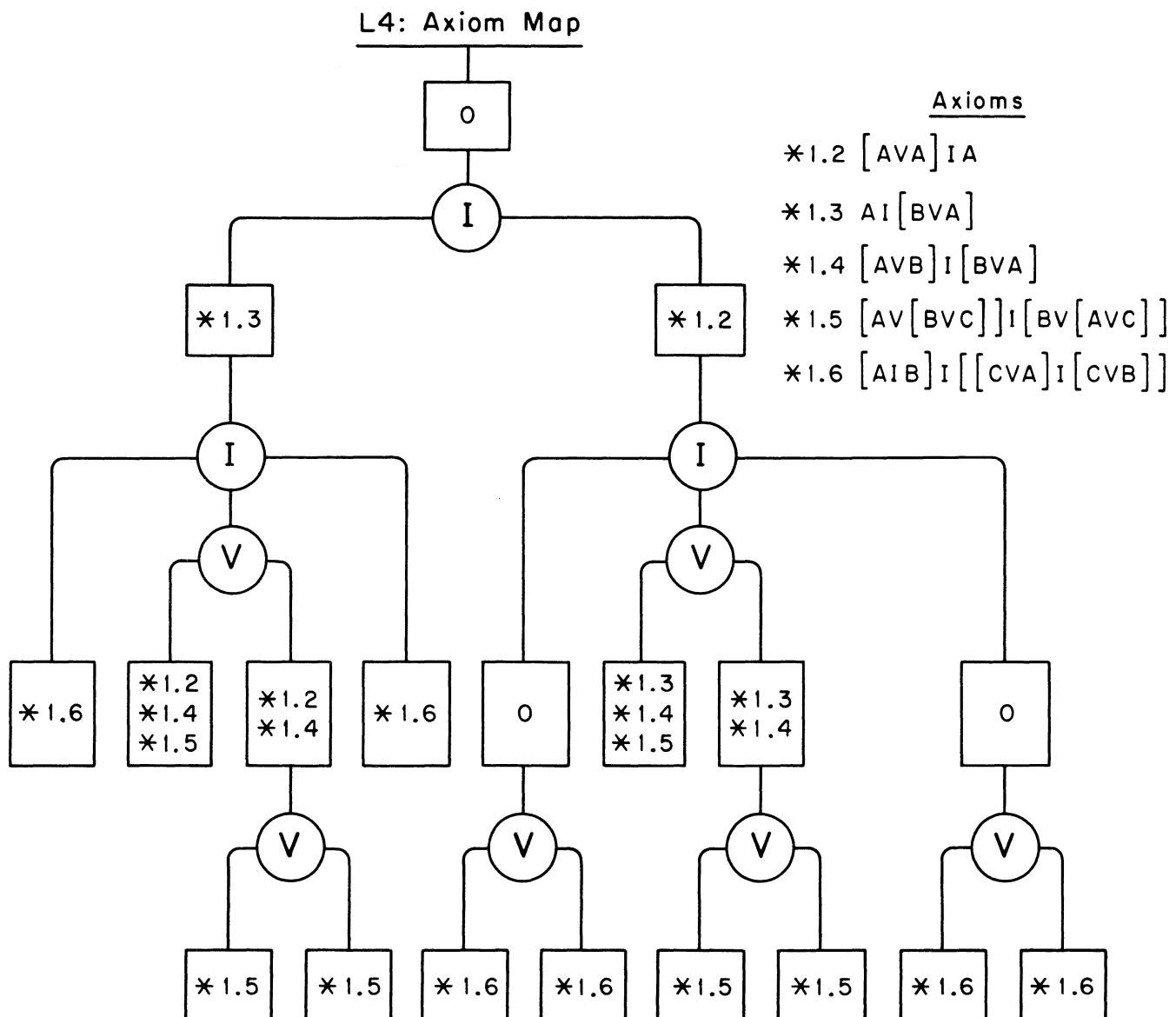


Fig. 22--Axiom Map

TRUE EXPRESSIONS MAP HOLDING AXIOMS	L4	0
*1.2, *1.3, *1.4, *1.5, *1.6.		I0
	9-1	0
MAPS LIST FOR MAIN CONNECTIVE I.	9-1	0
	9-2	
	9-3	0
SUBMAP, LEFT SIDES OF I.	9-2	9-4
		I0
		9-5
		V0
		9-6
SUBMAP, RIGHT SIDES OF I.	9-3	0
	9-7	
	I0	
	9-8	
	V0	
	9-9	0
AXIOM, VARIABLE ON LEFT OF I.	9-4	0
	*13	0
MAPS LIST FOR I ON LEFT OF I.	9-5	0
	9-10	
	9-11	0
MAPS LIST FOR V ON LEFT OF I.	9-6	0
	9-12	
	9-13	0
AXIOM, VARIABLE ON RIGHT OF I.	9-7	0
	*12	0
MAPS LIST FOR I ON RIGHT OF I.	9-8	0
	9-14	
	9-15	0
MAPS LIST FOR V ON RIGHT OF I.	9-9	0
	9-16	
	9-17	0
SUBMAP, LEFT OF I ON LEFT OF I.	9-10	9-18
SUBMAP, RIGHT OF I ON LEFT OF I.	9-11	0
SUBMAP, LEFT OF V ON LEFT OF I.	9-12	9-20
		V0
		9-21
SUBMAP, RIGHT OF V ON LEFT OF I.	9-13	0
SUBMAP, LEFT OF I ON RIGHT OF I.	9-14	9-22
		0
		V0
		9-23
SUBMAP, RIGHT OF I ON RIGHT OF I.	9-15	0
		V0
		9-24
SUBMAP, LEFT OF V ON RIGHT OF I.	9-16	0
SUBMAP, RIGHT OF V ON RIGHT OF I.	9-17	9-25
		0
		V0
		9-26
		9-27
AXIOM, VAR. LEFT OF I, LEFT OF I.	9-18	0
	*16	0
AXIOM, VAR. RIGHT OF I, LEFT OF I.	9-19	0
	*16	0

Fig. 23--Axiom Map in Keypunched Form

AXIOMS, VAR. LEFT OF V, LEFT OF I.	9-20	0
		*14
		*12 0
MAPS LIST, V LEFT OF V, LEFT OF I.	9-21	0
		9-28
		9-29 0
AXIOMS, VAR. RIGHT OF V, LEFT OF I.	9-22	0
		*15
		*14
		*12 0
MAPS LIST, V LEFT OF V, RIGHT OF I.	9-23	0
		9-30
		9-31 0
MAPS LIST, V RT. OF I, RT. OF I.	9-24	0
		9-32
		9-33
AXIOMS, VAR. LT. OF V, RT. OF I.	9-25	0
		*15
		*14
		*13 0
AXIOMS, VAR. RT. OF V, RT. OF I.	9-26	0
		*14
		*13 0
MAPS LIST, V RT. OF V, RT. OF I.	9-27	0
		9-34
		9-35 0
MAP, LT. OF V, LT. OF V, LT. OF I.	9-28	9-36 0
MAP, RT. OF V, LT. OF V, LT. OF I.	9-29	9-37 0
MAP, LT. OF V, LT. OF V, RT. OF I.	9-30	9-38 0
MAP, RT. OF V, LT. OF V, RT. OF I.	9-31	9-39 0
MAP, LT. OF V, RT. OF I, RT. OF I.	9-32	9-40 0
MAP, RT. OF V, RT. OF I, RT. OF I.	9-33	9-41 0
MAP, LT. OF V, RT. OF V, RT. OF I.	9-34	9-42 0
MAP, RT. OF V, RT. OF V, RT. OF I.	9-35	9-43 0
AXIOM, VAR. L OF V, L OF V, L OF I.	9-36	0
		*15 0
AXIOM, VAR. R OF V, L OF V, L OF I.	9-37	0
		*15 0
AXIOM, VAR. L OF V, L OF V, R OF I.	9-38	0
		*16 0
AXIOM, VAR. R OF V, L OF V, R OF I.	9-39	0
		*16 0
AXIOM, VAR. L OF V, R OF I, R OF I.	9-40	0
		*16 0
AXIOM, VAR. R OF V, R OF I, R OF I.	9-41	0
		*16 0
AXIOM, VAR. L OF V, R OF V, R OF I.	9-42	0
		*15 0
AXIOM, VAR. L OF V, R OF V, R OF I.	9-43	0
		*15 0

Fig. 23--(Continued)

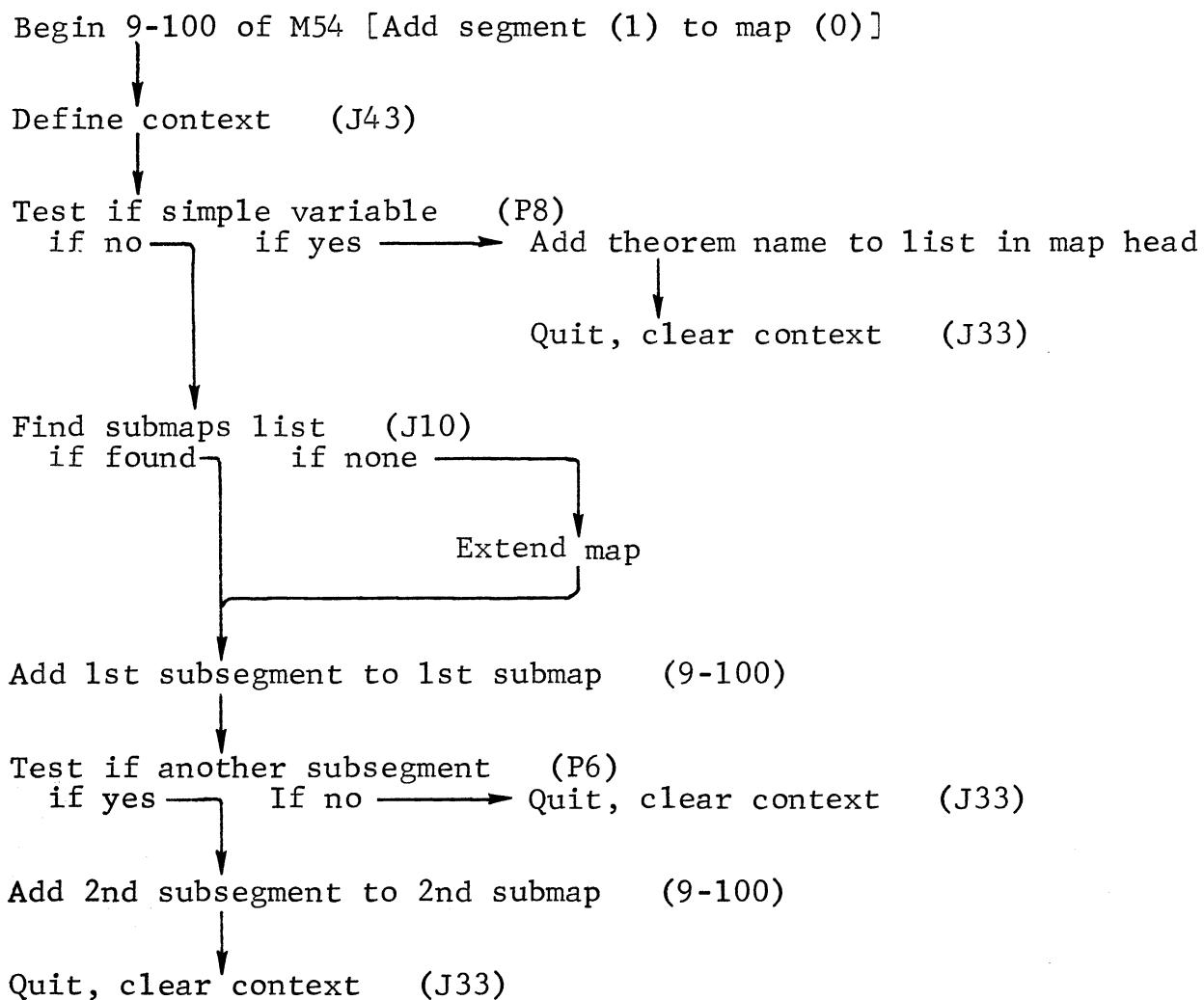
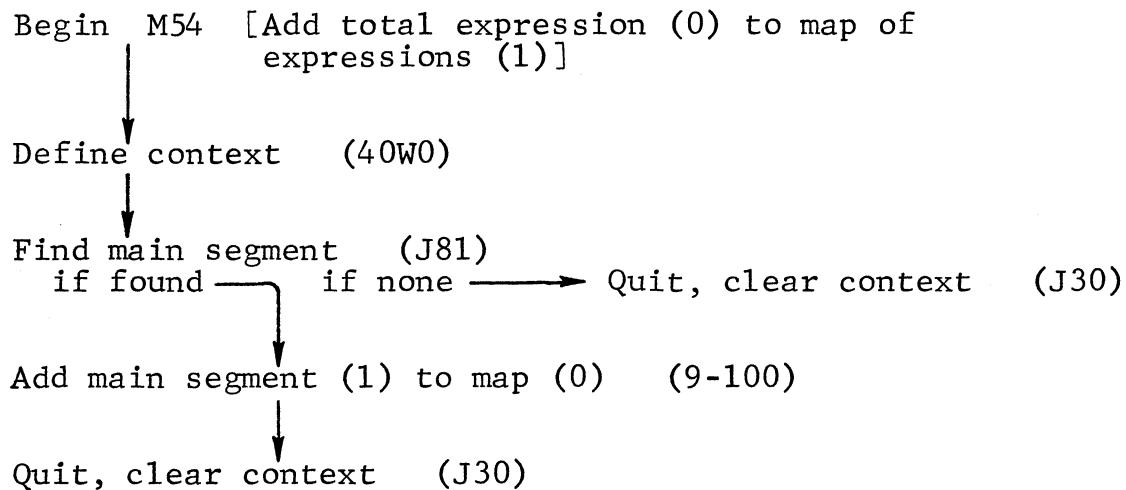


Fig. 24--Add Expression to Map

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 $[PVQ]I[PV[PVQ]]$, 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

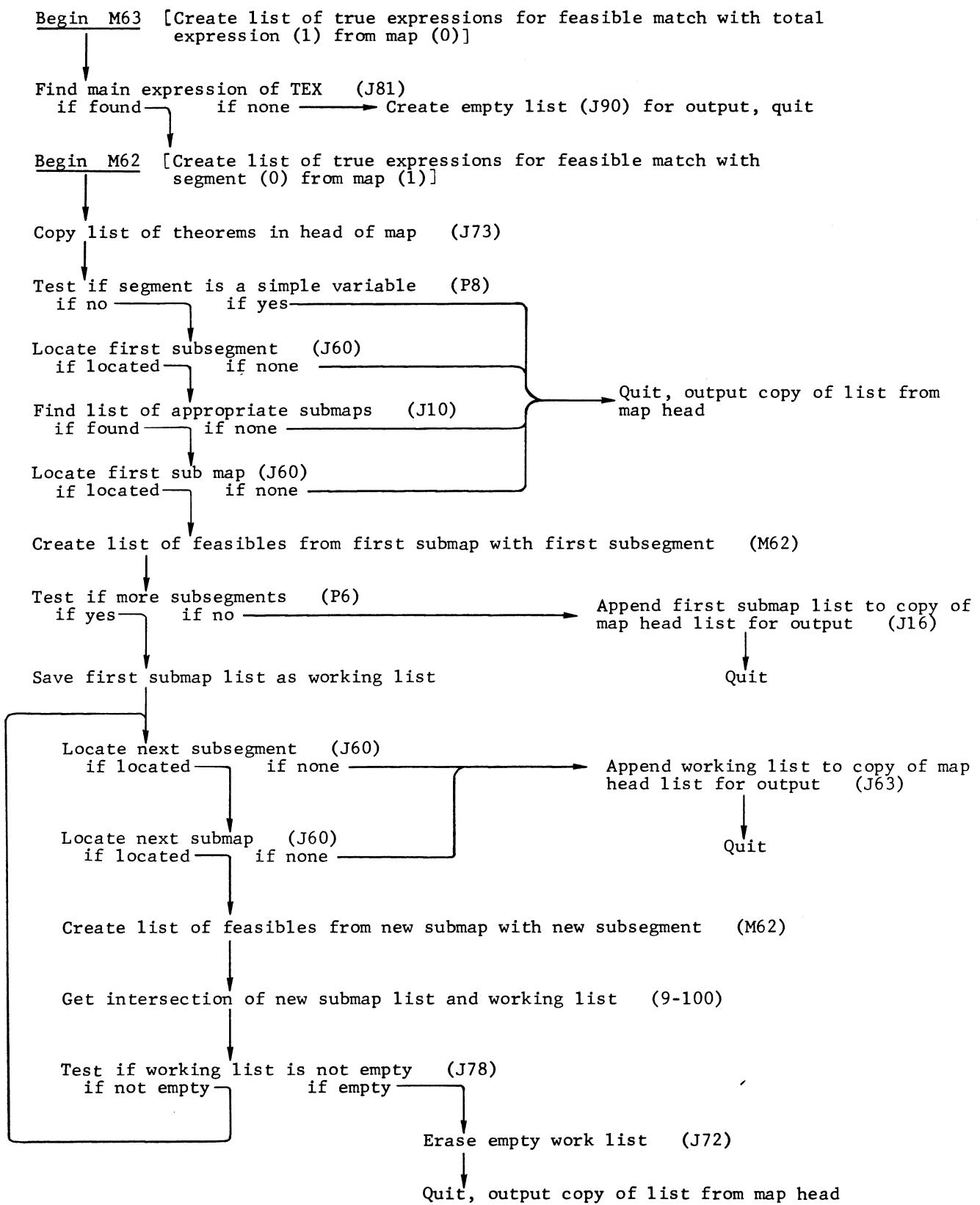


Fig. 25--Create List of Feasible Expressions

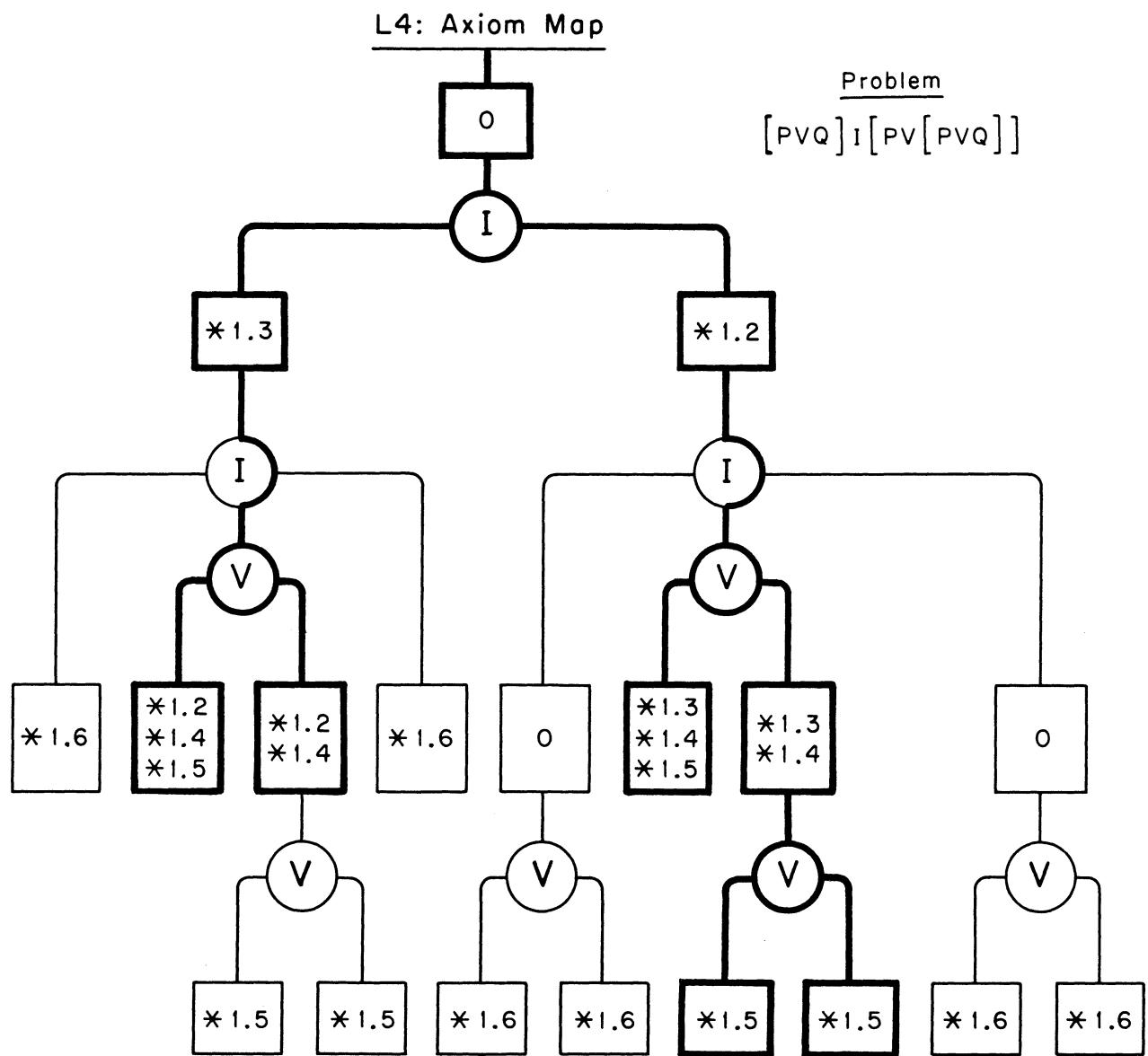


Fig. 26--Axiom Map with Overlaid Problem

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 (M11-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.

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

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.

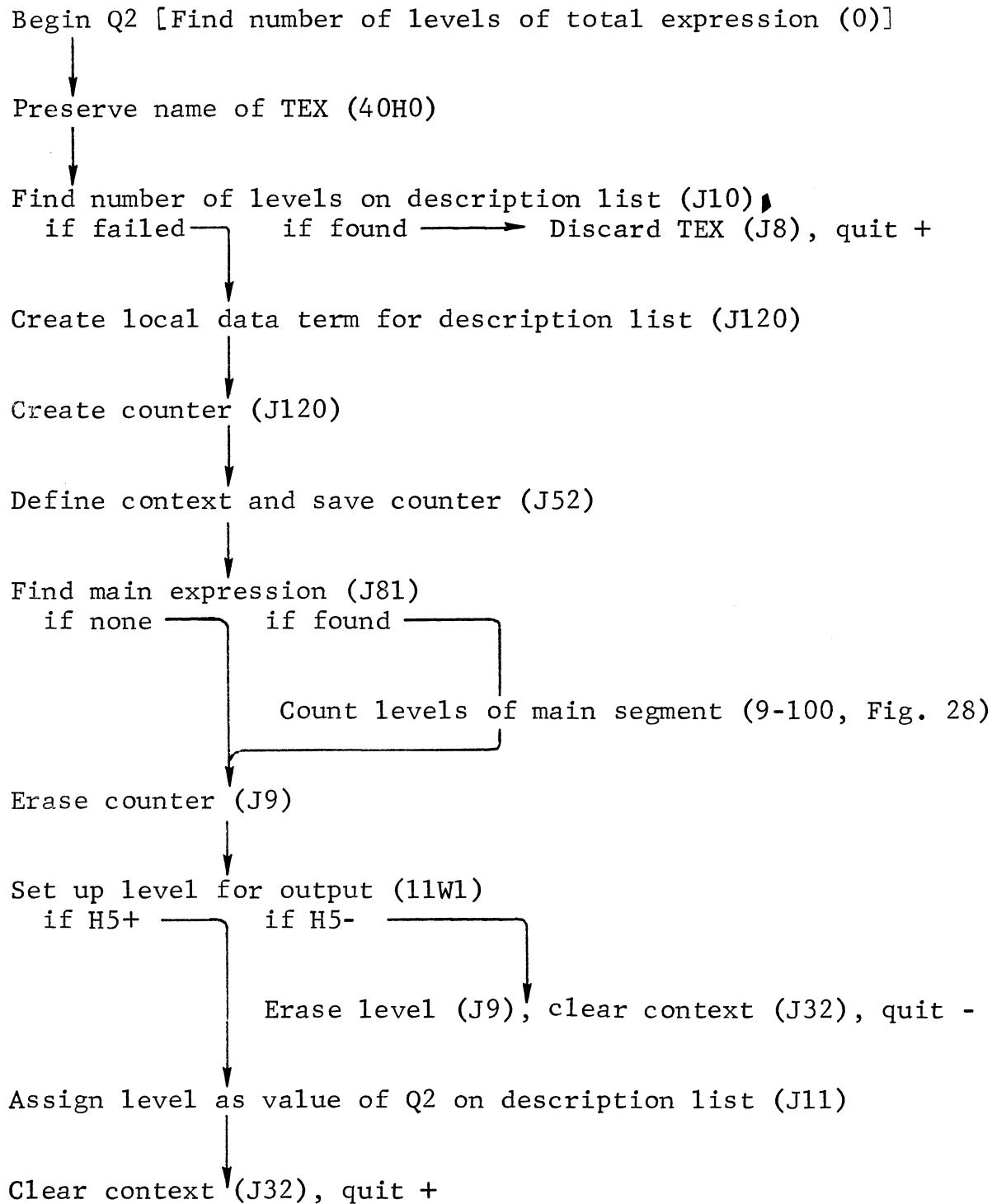


Fig. 27--Find Number of Levels

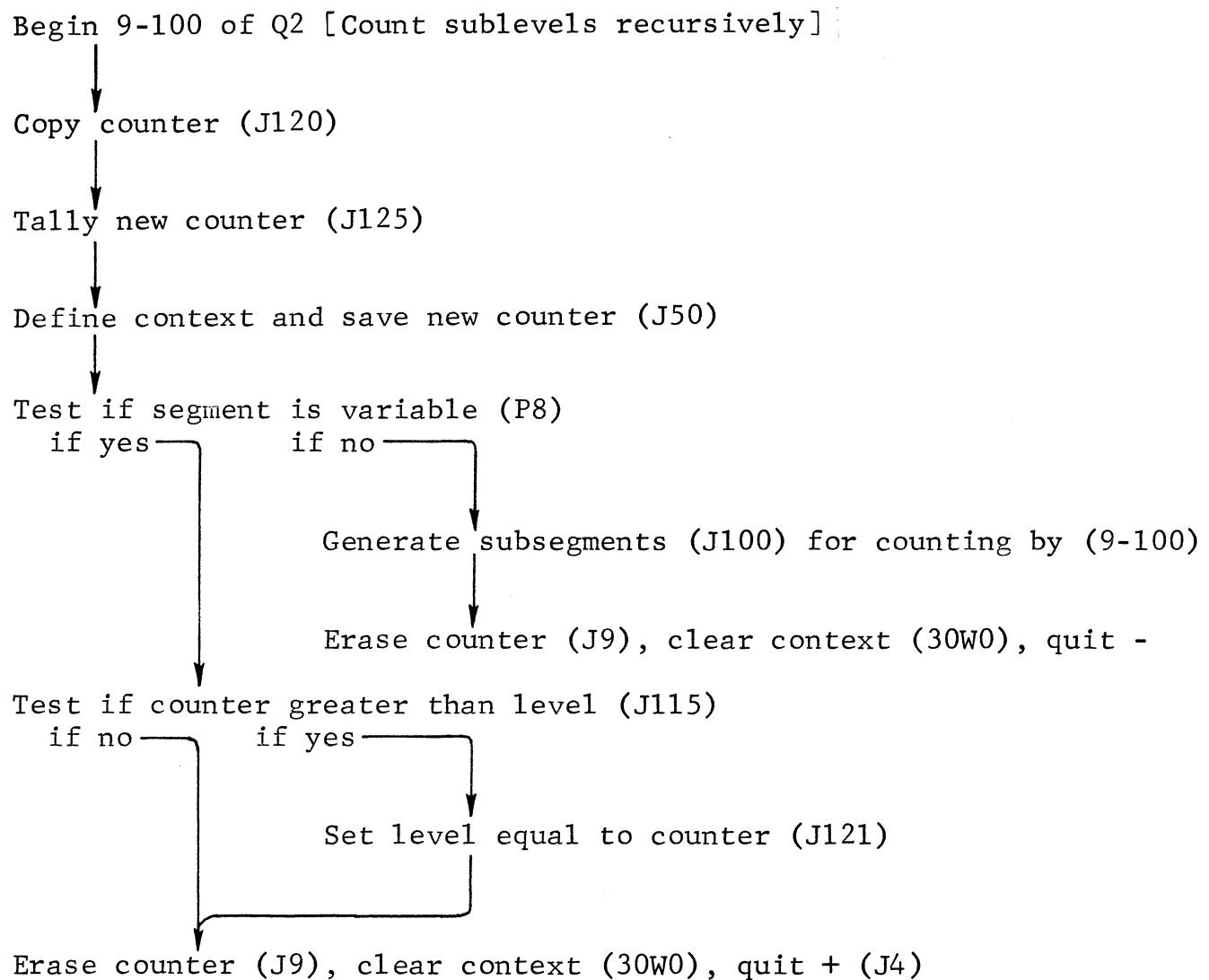


Fig. 28--Count Sublevels Recursively

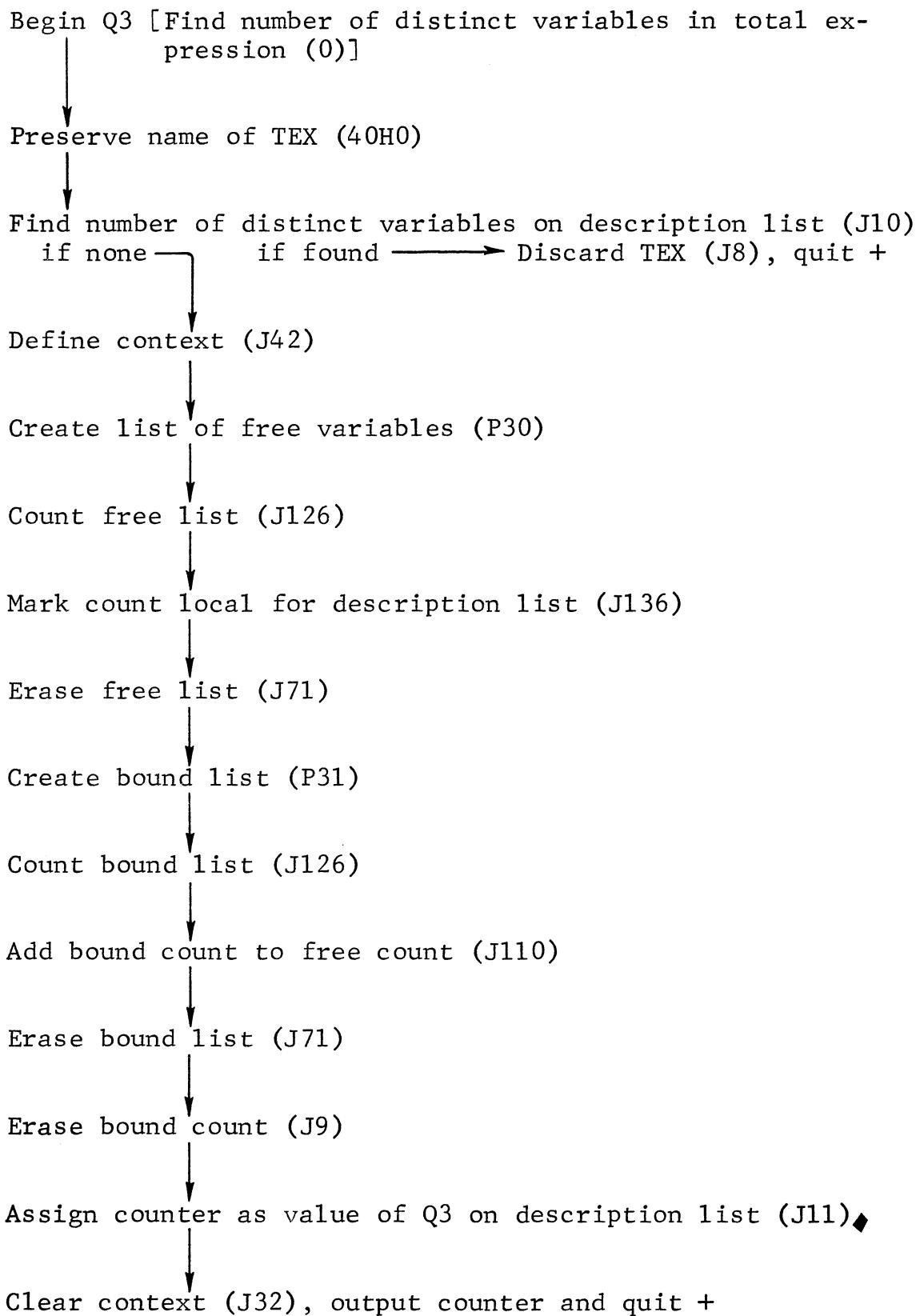


Fig. 29--Find Number of Distinct Variables

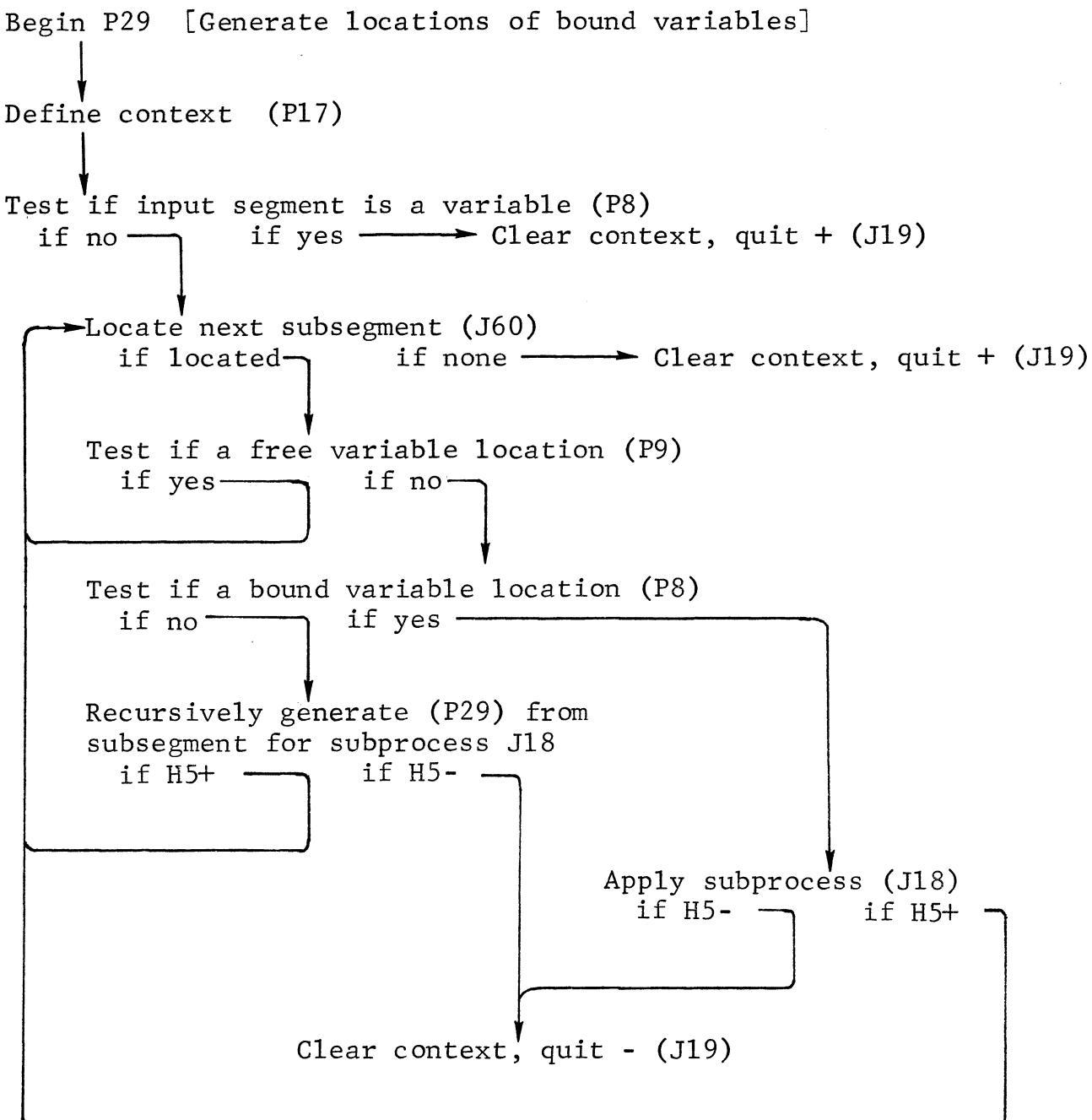


Fig. 30--Generate Locations of Bound Variables

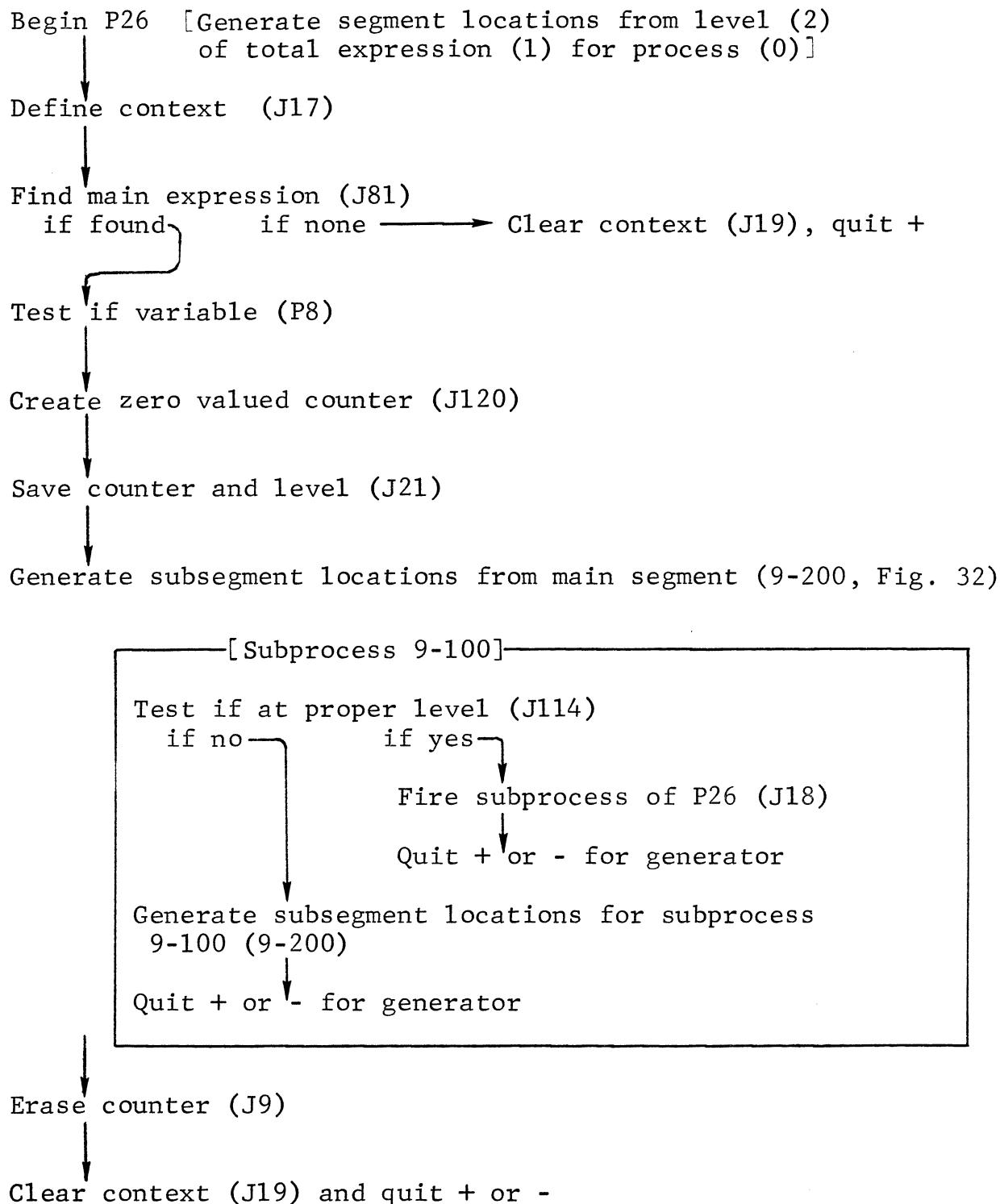


Fig. 31--Generate Segment Locations from Given Level of Expression

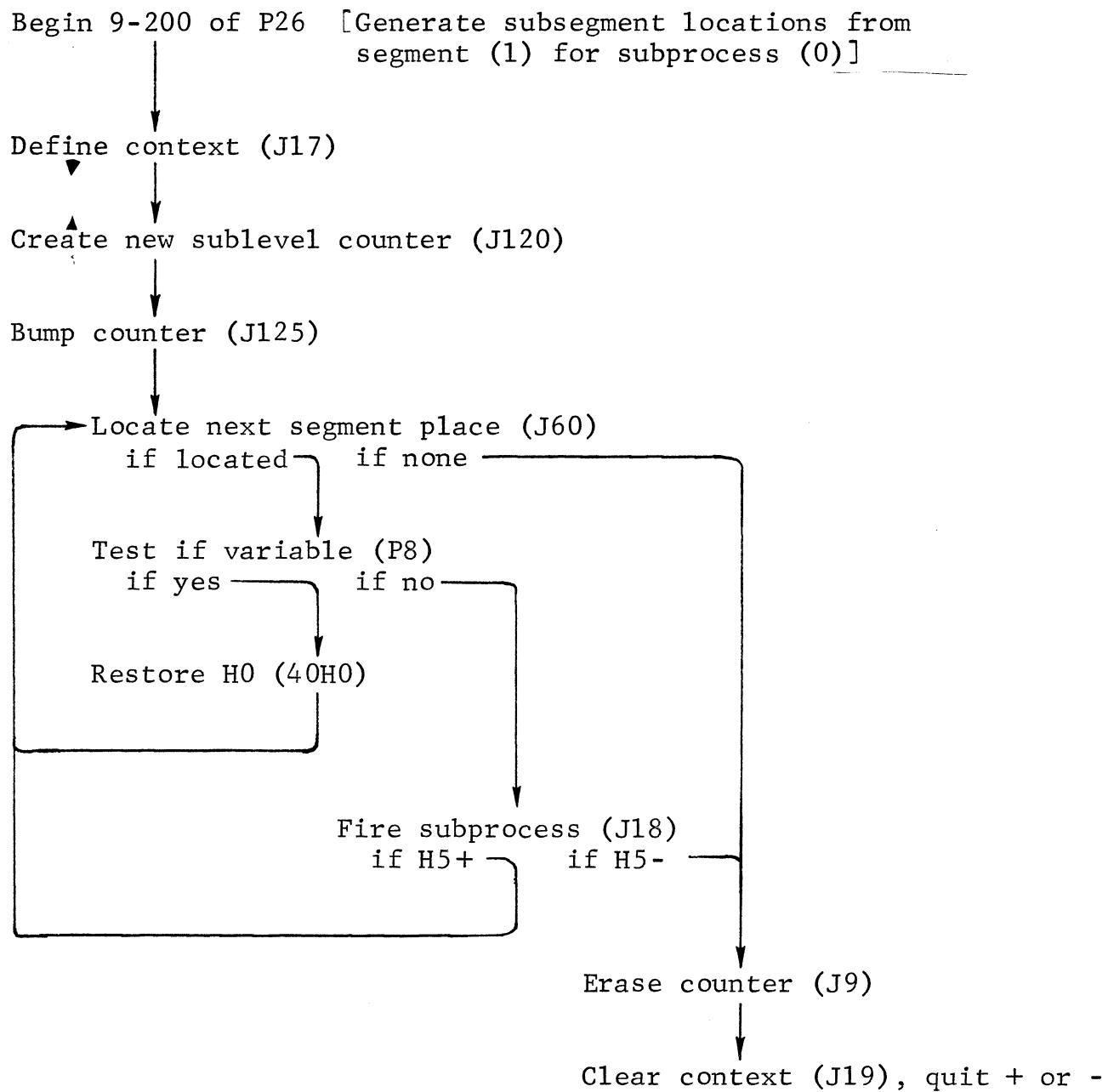


Fig. 32--Generate Subsegment Locations from Segment

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.

Head	{ *208	9-1	0
List Form Expression	{	(0	
		PO	
		IO	
		PO	
)0	0
Description List	{ 9-1	0	
		Q7	
		9-2	0
External Name	{ 9-2	21*2.08	

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 \therefore . 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

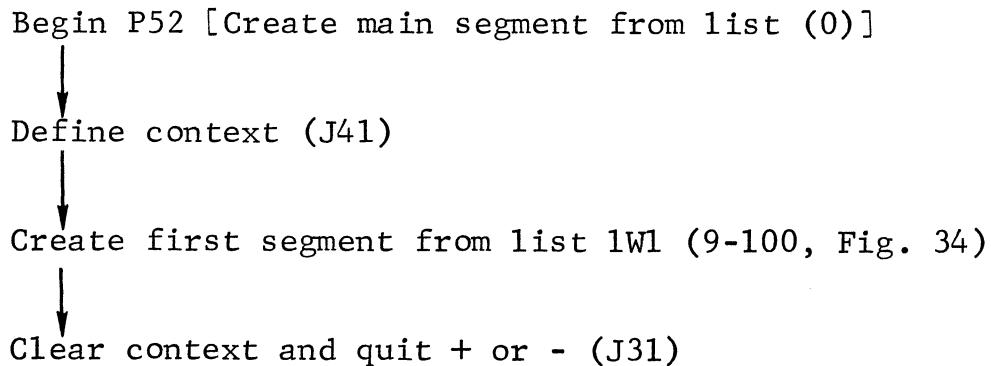


Fig. 33--Create Main Segment from List Expression

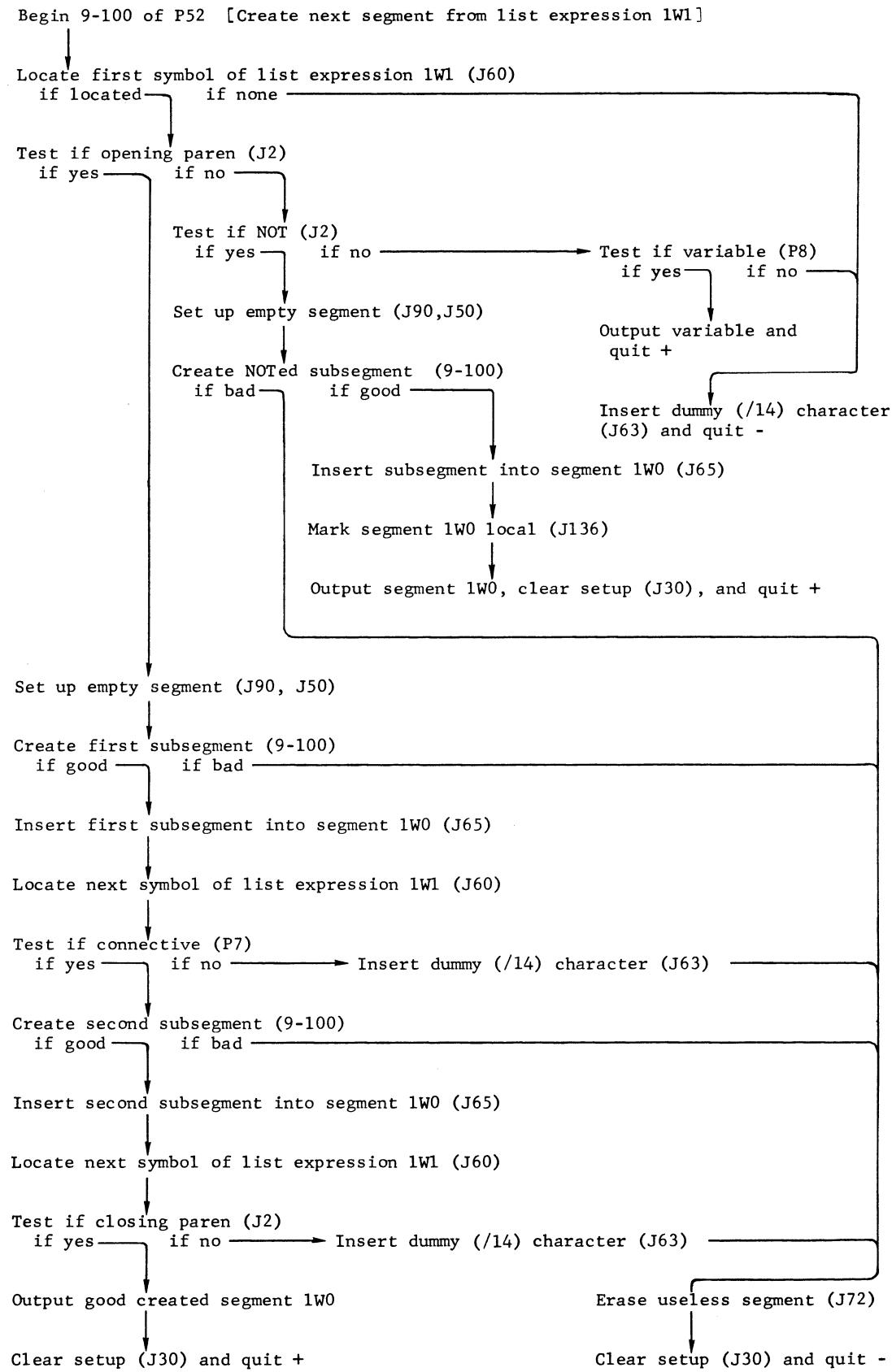


Fig. 34--Create Next Segment from List Expression

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 1W19. (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

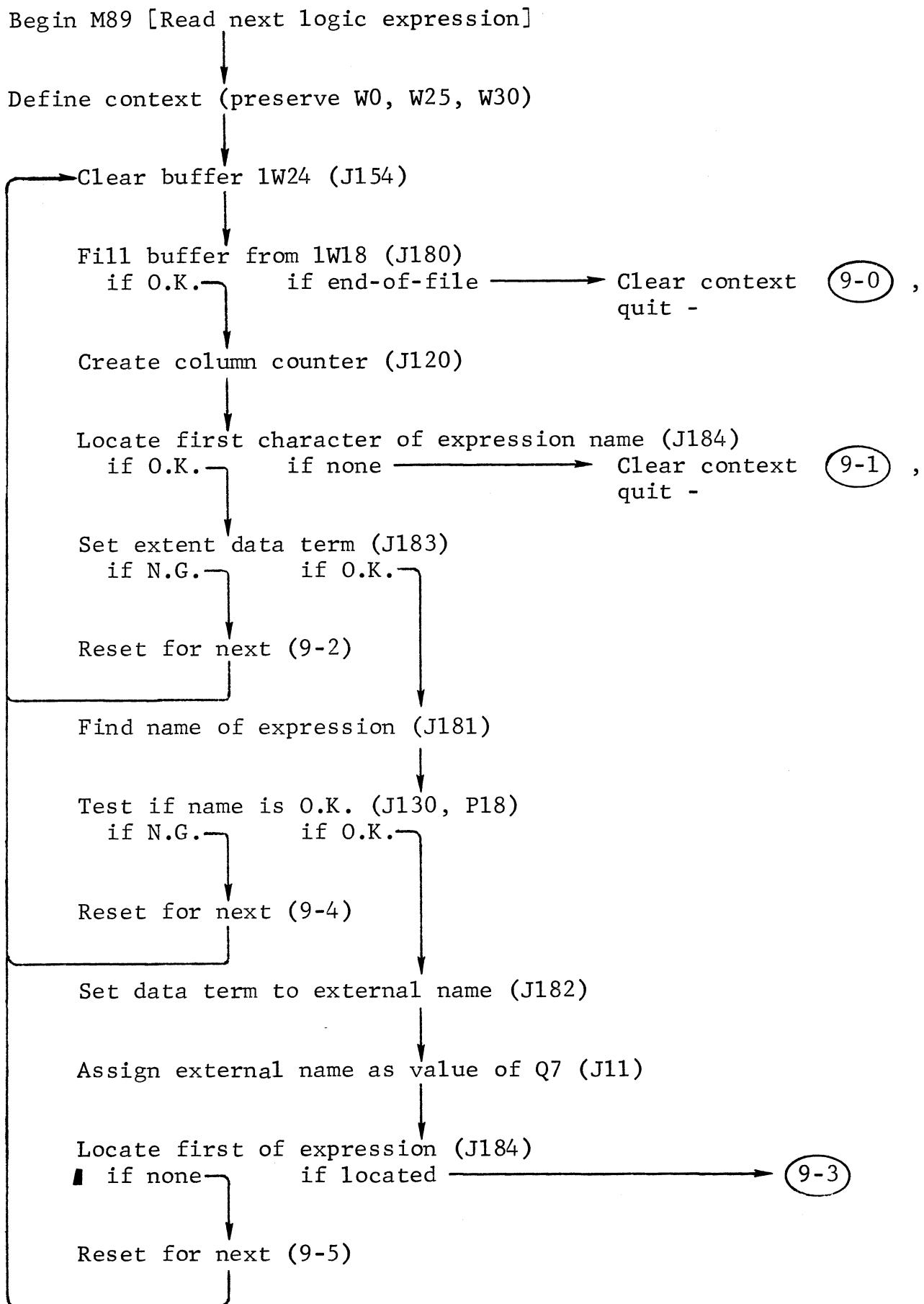


Fig. 35--Read Next Logic Expression

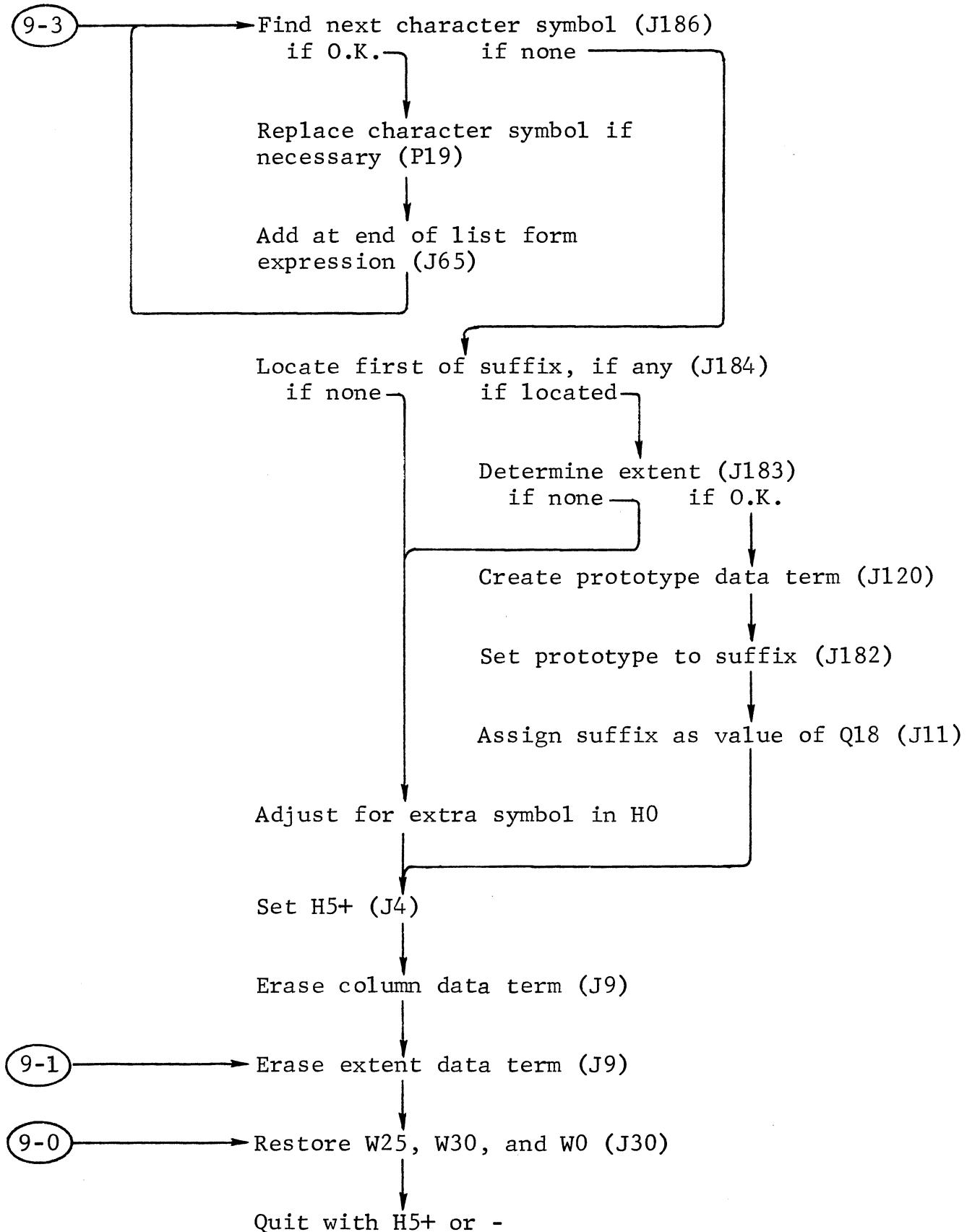


Fig. 35--(Continued)

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 "R0," 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 "R0", 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 1W31 to full trace mode ($1W31 = 1$) 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,0,099,C	STEFFERUD
ASSIGN	A6=SYSAR2	
ASSIGN	B6=SYSAR3	
IPL		
LOGIC THEORIST TEST	9	
RELOAD FROM TAPE 2	5 4 2	
MODIFIED ROUTINES	5 00	R
	1	R
Q17 FIND LEVEL OF SUBSEGMENT	Q17 40W0	Q017R000
REPLACEMENT IN TEX (0).	60W0	Q017R010
	10Q17	Q017R020
FIND CURRENT LEVEL.	J10	Q017R030
IF NONE,	70 J30	Q017R040
	11W0	Q017R050
FIND NUMBER OF LEVELS,	Q2	Q017R060
IF NONE, QUIT -.	70J30	Q017R065
COPY,	J120	Q017R070
SAVE ONE FOR OUTPUT,	40H0	Q017R080
	11WC	Q017R090
	J6	Q017R100
AND ASSIGN AS CURRENT LEVEL.	10Q17	Q017R110
	30W0 J11	Q017R120

MODIFIED DATA		5	01	D
/16 DUMMY EXPRESSION --		/16	9-1	/016D000
'DEFINITIONS'.			9-2	/016D010
		9-1	0	/016D020
			Q15	/016D030
			Q15	/016D040
			Q7	/016D050
			0	/016D060
EXTERNAL NAME			21	/016D070
CONNECTIVE 'I'.		9-2	I0	/016D080
			9-10	/016D090
			9-20	0 /016D100
DUMMY VARIABLE 'DEFIN'.		9-10	0	/016D110
			0	/016D120
			Q5	/016D130
			Q5	/016D140
			Q9	/016D150
			Q9	/016D160
			Q7	/016D170
			0	/016D180
EXTERNAL NAME.			21DEFIN	/016D190
DUMMY VARIABLE 'TIONS'.		9-20	0	/016D200
			0	/016D210
			Q5	/016D220
			Q5	/016D230
			Q9	/016D240
			Q9	/016D250
			Q7	/016D260
			0	/016D270
EXTERNAL NAME.			21TIONS	/016D280
RUN DATA HEADER		5	01	D -
LIMIT ON NUMBER OF SUBRROBLEMS	K20	+ 1	50	K020D000
LIMIT ON NUMBER OF SUBSTITUTIONS	K21	+ 1	50	K021D000
LIMIT ON EFFORT	K22	+ 1	20 0000	K022D000
R= ADD PROVED THEOREMS TO THEOREMS	K30	R		K030D000
Y = PRINT REJECTED SUBPROBLEMS.	K31	YES		K031D000
L6 LIST OF METHODS FOR ORIG PROBS	L6	C		L006D000
		M16		L006D010
		M17		L006D020
L7 LIST OF METHODS FOR PROBLEMS.	L7	0		L007D000
REPLACEMENT.		M13		L007D010
DETACHMENT.		M11		L007D010
FORWARD CHAINING.		M14		L007D010
BACKWARD CHAINING.		M15		L007D010
W12 SET-UP ENTRY SNAP ACTION.	W12	X13	0	W012D000
W13 SET-UP EXIT SNAP ACTION.	W13	X13	0	W013D000
	W15	X15	0	W015D000
	X21	0		X021D000
	X22	0		X022D000
DESCRIPTION LIST OF TRAP ACTIONS.	X23	0		X023D000

KICK OFF FOR PROVING THEOREMS.	5	X1
*1.01	((PIQ) . = . (-PVQ)) DEF.	
*2.33	((PVQVR) . = . ((PVQ)VR)) DEF.	X
*3.01	((P*Q) . = . -(-PV-Q)) DEF.	
4.01	((P=Q) . = . ((PIQ)(QIP))) DEF.	
*1.2	((AVA)IA)	
*1.3	((BI(AVB))	
*1.4	((AVB)I(BVA))	
*1.5	((AV(BVC))I(BV(ABC)))	
*1.6	((BIC)I((AVB)I(ABC)))	
 *2.01	((PI-P)I-P)	
*2.02	(QI(PIQ))	
*2.03	((PI-Q)I(QI-P))	
*2.04	((PI(QIR))I(QI(PIR)))	
*2.05	((QIR)I((PIQ)I(PIR)))	
*2.06	((PIQ)I((QIR)I(PIR)))	
*2.07	(PI(PVP))	
*2.08	(PIP)	
*2.10	(-PVP)	
*2.11	(PV-P)	
*2.12	(PI--P)	
*2.13	(PV---P)	
*2.14	(--PIP)	
*2.15	((-PIQ)I(-QIP))	
*2.20	(PI(PVQ))	
*2.21	(-PI(PIQ))	
*2.24	(PI(-PIQ))	
*3.13	((P*Q)I(-PV-Q))	
*3.14	((-PV-Q)I-(P*Q))	
3.24	-(P-P)	
*4.13	(P---P)	
*4.20	(P=P)	
*4.24	(P=(P*P))	
*4.25	(P=(PVP))	
 3.02	((PIQIR) . = . ((PIQ)(QIR))) DEF.	
4.02	((P=Q=R) . = . ((P=Q)(Q=R))) DEF.	X
*4.34	((P*Q*R) . = . ((P*Q)*R)) DEF.	X
*2.16	((PIQ)I(-QI-P))	
*2.17	((-QI-P)I(PIQ))	
*2.18	((-PIP)IP)	
*2.25	(PV((PVQ)IQ))	
*2.26	(-PV((PIQ)IQ))	
*2.27	(PI((PIQ)IQ))	
*2.30	((PV(QVR))I(PV(RVQ)))	
*2.31	((PV(QVR))I((PVQ)VR))	
*2.32	((PVQ)VR I(PV(QVR)))	
*2.36	((QIR)I((PVQ)I(RVP)))	
*2.37	((QIR)I((QVP)I(PVR)))	
*2.38	((QIR)I((QVP)I(RVP)))	

*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(PI-Q))
*2.52	(-(PIQ)I(-PI-Q))
*2.53	(-(PIQ)I(QIP))
*2.54	((PVQ)I(-PIQ))
*2.55	(-PI((PVQ)IQ))
*2.56	(-QI((PVQ)IP))
*2.60	((-(PIQ)I((PIQ)IQ))
*2.61	((PIQ)I((-PIQ)IQ))
*2.62	((PVQ)I((PIQ)IQ))
*2.621	((PIQ)I((PVQ)IQ))
*2.63	((PVQ)I((-PVQ)IQ))
*2.64	((PVQ)I((PV-Q)IP))
*2.65	((PIQ)I((PI-Q)I-P))
*2.67	((((PVQ)IQ)I(PIQ))
*2.68	((((PIQ)IQ)I(PVQ))
*2.69	((((PIQ)IQ)I((QIP)IP))
*2.73	((PIQ)I((PVQVR)I(QVR)))
*2.74	((QIP)I((PVQVR)I(PVR)))
*2.75	((PVQ)I((PV(QIR))I(PVR)))
*2.76	((PV(QIR))I((PVQ)I(PVR)))
*2.77	((PI(QIR))I((PIQ)I(PIR)))
*2.80	((QVR)I((-RVS)I(QVS)))
*2.81	((QI(RIS))I((PVQ)I((PVR)I(PVS))))
*2.82	((PVQVR)I((PV-RVS)I(PVQVS)))
*2.83	((PI(QIR))I((PI(RIS))I(PI(QIS))))
*2.85	((((PVQ)I(PVR))I(PV(QIR)))
*2.86	((((PIQ)I(PIR))I(PI(QIR)))
*3.10	((P*Q)I-(-PV-Q))
*3.11	(-(-PV-Q)I(P*Q))
*3.12	((-PV-Q)V(P*Q))
*3.20	(PI(QI(P*Q)))
*3.21	(QI(PI(P*Q)))
*3.22	((P*Q)I(Q*P))
*3.26	((P*Q)IP)
*3.27	((P*Q)IQ)
*3.30	((((P*Q)IR)I(PI(QIR)))
*3.31	((PI(QIR))I((P*Q)IR))
3.33	((((PIQ)(QIR))I(PIR))
3.34	((((QIR)(PIQ))I(PIR))
3.35	((P(PIQ))IQ)

*3.37 (((P*Q)IR)I((P*-R)I-Q))
*3.40 ((P*Q)I(PIQ))
*3.41 ((PIR)I((P*Q)IR))
*3.42 ((QIR)I((P*Q)IR))
3.43 (((PIQ)(PIR))I(PI(Q*R)))
3.44 (((QIR)(RIP))I((QVR)IP))
*3.45 ((PIQ)I((P*R)I(Q*R)))
3.47 (((PIR)(QIS))I((P*Q)I(R*S)))
3.48 (((PIR)(QIS))I((PVQ)I(RVS)))
*4.10 ((PIQ)=(-QI-P))
*4.11 ((P=Q)=(-P=-Q))
*4.12 ((P=-Q)=(Q=-P))
*4.14 (((P*Q)IR)=((P*-R)I-Q))
*4.15 (((P*Q)I-R)=((Q*R)I-P))
*4.21 ((P=Q)=(Q=P))
4.22 (((P=Q)(Q=R))I(P=R))
*4.30 ((P*Q)=(Q*P))
*4.31 ((PVQ)=(QVP))
*4.32 (((P*Q)*R)=(P*(Q*R)))
*4.33 (((PVQ)VR)=(PV(QVR)))
*4.36 ((P=Q)I((P*R)=(Q*R)))
*4.37 ((P=Q)I((PVR)=(QVR)))
4.38 (((P=R)(Q=S))I((P*Q)=(R*S)))
4.39 (((P=R)(Q=S))I((PVQ)=(RVS)))
4.40 ((P(QVR))=((P*Q)V(P*R)))
*4.41 ((PV(Q*R))=((PVQ)*(PVR)))
*4.42 (P=((P*Q)V(P*-Q)))
4.43 (P=((PVQ)(PV-Q)))
*4.44 (P=(PV(P*Q)))
4.45 (P=(P(PVQ)))
*4.50 ((P*Q)=-(-PV-Q))
*4.51 (- (P*Q)=(-PV-Q))
4.52 ((P-Q)=-(-PVQ))
4.53 (- (P-Q)=-(PVQ))
*4.54 ((-P*Q)=- (PV-Q))
*4.55 (- (-P*Q)=(PV-Q))
4.56 ((-P-Q)=- (PVQ))
4.57 (- (-P-Q)=(PVQ))
*4.60 ((PIP)=(-PVQ))
4.61 (- (PIQ)=(P-Q))
*4.62 ((PI-Q)=(-PV-Q))
*4.63 (- (PI-Q)=(P*Q))
*4.64 ((-PIQ)=(PVQ))
4.65 (- (-PIQ)=(-P-Q))
4.66 ((-P-Q)=(PV-Q))
*4.67 (- (-PI-Q)=(-P*Q))
*4.70 ((PIQ)=(PI(P*Q)))
*4.71 ((PIQ)=(P=(P*Q)))
*4.72 ((PIQ)=(Q=(PVQ)))
*4.73 (QI(P=(P*Q)))
*4.74 (-PI(Q=(PVQ)))

4.76 (((PIQ)(PIR))=(PI(Q*R)))
4.77 (((QIP)(RIP))=((QVR)IP))
*4.78 (((PIQ)V(PIR))=(PI(QVR)))
*4.79 (((QIP)V(RIP))=((Q*R)IP))
*4.80 ((PI-P)=-P)
*4.81 ((-PIP)=P)
4.82 (((PIQ)(PI-Q))=-P)
4.83 (((PIQ)(-PIQ))=Q)
*4.84 ((P=Q)I((PIR)=(QIR)))
*4.85 ((P=Q)I((RIP)=(RIQ)))
*4.86 ((P=Q)I((P=R)=(Q=R)))
*4.87 (((P*Q)IR)=(PI(QIR))=(QI(PIR)=(Q*P)IR)) X
*5.30 (((P*Q)IR)=((P*Q)I(P*R)))
*5.31 (((R*P)IQ)I(PI(Q*R)))
*5.32 ((PI(Q=R))=((P*Q)=(P*R)))
5.33 ((P(QIR))=(P*((P*Q)IR)))
5.35 (((PIQ)(PIR))I(PI(Q=R)))
5.36 ((P(P=Q))=(Q*(P=Q)))
*5.40 ((PI(PIQ))=(PIQ))
*5.41 (((PIQ)I(PIR))=(PI(QIR)))
*5.42 ((PI(QIR))=(PI(QI(P*R))))
*5.44 ((PIQ)I((PIR)=(PI(Q*R))))
*5.50 (PI((PIQ)=Q))
*5501 (PI(Q=(P=Q)))
5.53 (((PVQVR)IS)=((PIS)(QIS)*(RIS))) X
*5.54 (((P*Q)=P)V((P*Q)=Q))
*5.55 (((PVQ)=P)V((PVQ)=Q))
5.60 (((P-Q)IR)=(PI(QVR)))
5.61 (((PVQ)-Q)=(P*-Q))
*5.62 (((P*Q)V-Q)=(PV-Q))
*5.63 ((PVQ)=(PV(-P*Q)))
*5.70 (((PVR)=(QVR))=(RV(P=Q)))
*5.71 ((QI-R)I(((PVQ)*R)=(P*R)))
*5.74 ((PI(Q=R))=((PIQ)=(PIR)))
5.75 (((RI-Q)(P=(QVR))))I((P*-Q)=R))

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.

-78-

JOB 8168,LTNEW1,EAS826,5MIN,0,099,C STEFFERUD 1 003650 06/05/63
ASSIGN A6=SYSAR2
ASSIGN B6=SYSAR3
IPL

LOGIC THEORIST TEST 9

RELOAD FROM TAPE 2 5 4 2
MEMORY RELOADED FROM TAPE A6. 3956 0 0 J166 J165

MODIFIED ROUTINES		5 00	R	
		1	R	
671	4 0 24587 3976	Q17	4 0 WO	Q017R000
3976	6 0 24587 3975		6 0 WO	Q017R010
3975	1 0 671 3977		1 0 Q17	Q017R020
3977	0 0 24770 3978		J10	Q017R030
3978	7 0 3979 24790		7 0 J30	Q017R040
3979	1 1 24587 3980		1 1 WO	Q017R050
3980	0 0 656 3981		Q2	Q017R060
3981	7 0 24790 3982		7 0 J30	Q017R065
3982	0 0 24880 3983		J120	Q017R070
3983	4 0 24574 3984		4 0 HO	Q017R080
3984	1 1 24587 3985		1 1 WO	Q017R090
3985	0 0 24766 3986		J6	Q017R100
3986	1 0 671 3987		1 0 Q17	Q017R110
3987	3 0 24587 24771		3 0 WO J11	Q017R120

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		5	01	D	
1540 0 2 3989 3988	/16	9-1	/16 DUMMY EXPRESSION --		/016D000
3988 0 2 3990 0		9-2 0	'DEFINITIONS'.		/016D010
3989 0 4 0 3991	9-1	0			/016D020
3991 0 0 669 3992		Q15			/016D030
3992 0 0 669 3993		Q15			/016D040
3993 0 0 661 3994		Q7			/016D050
3994 0 2 3995 0		0			/016D060
3995 2 1	2 1	EXTERNAL NAME			/016D070
3990 0 0 134 3996	9-2	I0	CONNECTIVE 'I'.		/016D080
3996 0 2 3998 3997		9-10			/016D090
3997 0 2 3999 0		9-20 0			/016D100
3998 0 2 4000 0	9-10	0	DUMMY VARIABLE 'DEFIN'.		/016D110
4000 0 4 0 4001		0			/016D120
4001 0 0 659 4002		Q5			/016D130
4002 0 0 659 4003		Q5			/016D140
4003 0 0 663 4004		Q9			/016D150
4004 0 0 663 4005		Q9			/016D160
4005 0 0 661 4006		Q7			/016D170
4006 0 2 4007 0		0			/016D180
4007 2 1 DEFIN	2 1 DEFIN	0	EXTERNAL NAME.		/016D190
3999 0 2 4008 0	9-20	0	DUMMY VARIABLE 'TIONS'.		/016D200
4008 0 4 0 4009		0			/016D210
4009 0 0 659 4010		Q5			/016D220
4010 0 0 659 4011		Q5			/016D230
4011 0 0 663 4012		Q9			/016D240
4012 0 0 663 4013		Q9			/016D250
4013 0 0 661 4014		Q7			/016D260
4014 0 2 4015 0		0			/016D270
4015 2 1 TIONS	2 1 TIONS	0	EXTERNAL NAME.		/016D280

RUN DATA HEADER	5	01	D -	
164 0 1 50	K20	+	1	50 LIMIT ON NUMBER OF SUBPROBLEMS K020D000
165 0 1 50	K21	+	1	50 LIMIT ON NUMBER OF SUBSTITUTIONS K021D000
166 0 1 200000	K22	+	1 20 0000	LIMIT ON EFFORT K022D000
174 0 0 704 0	K30	R		R= ADD PROVED THEOREMS TO THEOREMS K030D000
175 0 0 834 0	K31	YES		Y = PRINT REJECTED SUBPROBLEMS. K031D000
250 0 4 0 4016	L6	O		L6 LIST OF METHODS FOR ORIG PROBS L006D000
4016 0 0 310 4017		M16		L006D010
4017 0 0 311 0		M17		L006D020
251 0 4 0 4018	L7	O		L7 LIST OF METHODS FOR PROBLEMS. L007D000
4018 0 0 307 4019		M13		REPLACEMENT. L007D010
4019 0 0 305 4020		M11		DETACHMENT. L007D010
4020 0 0 308 4021		M14		FORWARD CHAINING. L007D010
4021 0 0 309 0		M15		BACKWARD CHAINING. L007D010
24599 0 0 797 0	W12	X13 0		W12 SET-UP ENTRY SNAP ACTION. W012D000
24600 0 0 797 0	W13	X13 0		W13 SET-UP EXIT SNAP ACTION. W013D000
24602 0 0 799 0	W15	X15 0		W015D000
805 0 4 0 0	X21	O		X021D000
806 0 4 0 0	X22	O		X022D000
807 0 4 0 0	X23	O		X023D000
KICK OFF FOR PROVING THEOREMS.				DESCRIPTION LIST OF TRAP ACTIONS.
	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 ((PVQ/UGH/VR).=.((PVQ)VR))  
*3.01  (A*B).=.(-(-AV-B)) DEF.  
*4.01  (A=B).=.((AIB)*(BIA)) DEF.  
*1.2   (AVA) IA  
*1.3   BI(AVB)  
*1.4   (AVB)I(BVA)  
*1.5   (AV(BVC))I(BV(AVC))  
*1.6   (BIC)I((AVB)I(ABC))  
  
*2.01  (PI-P)I-P  
*2.02  QI(PIQ)  
*2.03  (PI-Q)I(QI-P)  
*2.04  (PI(QIR))I(QI(PIR))  
*2.05  (QIR)I((PIQ)I(PIR))  
*2.06  (PIQ)I((QIR)I(PIR))  
*2.07  PI(PVP)  
*2.08  PIP  
*2.10  -PVP  
*2.11  PV-P  
*2.12  PI--P  
*2.13  PV---P  
*2.14  --PIP  
*2.15  (-PIQ)I(-QIP)  
*2.20  PI(PVQ)  
*2.21  -PI(PIQ)  
*2.24  PI(-PIQ)  
*3.13  (-(P*Q))I(-PV-Q)  
*3.14  (-PV-Q)I(-(P*Q))  
*3.24  -(P*-P)  
*4.13  P=--P  
*4.20  P=P  
*4.24  P=(P*P)  
*4.25  P=(PVP)
```

TO PROVE

*2.01 $(P \rightarrow P) \rightarrow P$

1. $(\neg P \vee P) \rightarrow P$

, SUBLVEL REPLACEMENT

PROOF FOUND.

GIVEN	*1.2	(AV)IA
SUBSTITUTION	1.	$(\neg P \vee P) \rightarrow P$
GIVEN		DEFINITIONS
SUBLVEL REPLACEMENT	*2.01	$(P \rightarrow P) \rightarrow P$
Q.E.D.		

EFFORT	LIMIT 200000	ACTUAL 8062
SUBPROBLEMS	LIMIT 50	ACTUAL 1
SUBSTITUTIONS	LIMIT 50	ACTUAL 2

REMEMBER PROVED THEOREM

*2.01 $(A \rightarrow A) \rightarrow A$

TO PROVE

*2.02 QI(PIQ)
1. QI(-PVQ)

, SUBLVEL REPLACEMENT

PROOF FOUND.

GIVEN	*1.3	BI(AVB)
SUBSTITUTION	1.	QI(-PVQ)
GIVEN		DEFINITIONS
SUBLVEL REPLACEMENT	*2.02	QI(PIQ)
Q.E.D.		

EFFORT	LIMIT 200000	ACTUAL 6041
SUBPROBLEMS	LIMIT 50	ACTUAL 1
SUBSTITUTIONS	LIMIT 50	ACTUAL 2

REMEMBER PROVED THEOREM

*2.02 AI(BIA)

TO PROVE

*2.03 $(PI-Q)I(QI-P)$
1. $(-PV-Q)I(-QV-P)$

, SUBLVEL REPLACEMENT

PROOF FOUND.

GIVEN	*1.4	$(AVB)I(BVA)$
SUBSTITUTION	1.	$(-PV-Q)I(-QV-P)$
GIVEN		DEFINITIONS
SUBLVEL REPLACEMENT	*2.03	$(PI-Q)I(QI-P)$
Q.E.D.		

EFFORT	LIMIT 200000	ACTUAL 11904
SUBPROBLEMS	LIMIT 50	ACTUAL 1
SUBSTITUTIONS	LIMIT 50	ACTUAL 2

REMEMBER PROVED THEOREM

*2.03 $(AI-B)I(BI-A)$

TO PROVE

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

1. (PI(-QVR))I(QI(-PVR))
2. (-PV(QIR))I(-QV(PIR))
3. (-PV(-QVR))I(-QV(-PVR))

- SUBLVEL REPLACEMENT
- SUBLVEL REPLACEMENT
- SUBLVEL REPLACEMENT

PROOF FOUND.

GIVEN	#1.5	(AV(BVC))I(BV(AVC))
SUBSTITUTION	3.	(-PV(-QVR))I(-QV(-PVR))
GIVEN		DEFINITIONS
SUBLVEL REPLACEMENT	#2.04	(PI(QIR))I(QI(PIR))
Q.E.D.		

EFFORT	LIMIT 200000	ACTUAL 32262
SUBPROBLEMS	LIMIT 50	ACTUAL 3
SUBSTITUTIONS	LIMIT 50	ACTUAL 4

REMEMBER PROVED THEOREM

*2.04 (AI(BIC))I(BI(AIC))

TC PROVE

*2.05 (QIR)I((PIQ)I(PIR))
1. (QIR)I((-PVQ)I(-PVR)) , SUBLVEL REPLACEMENT

PROOF FOUND.

GIVEN	*1.6	(BIC)I((AVB)I(VC))
SUBSTITUTION	1.	(QIR)I((-PVQ)I(-PVR))
GIVEN		DEFINITIONS
SUBLVEL REPLACEMENT	*2.05	(QIR)I((PIQ)I(PIR))
Q.E.D.		

EFFORT	LIMIT 200000	ACTUAL 13440
SUBPROBLEMS	LIMIT 50	ACTUAL 1
SUBSTITUTIONS	LIMIT 50	ACTUAL 2

REMEMBER PROVED THEOREM

*2.05 (AIB)I((CIA)I(CIB))

TC PROVE

*2.06 (PIQ)I((QIR)I(PIR))
1. (PIQ)I((-QVR)I(-PVR)) , SUBLVEL REPLACEMENT
2. (-PVQ)I((-(QIR)V(PIR)) , SUBLVEL REPLACEMENT
3. (-PVQ)I((-(QVR)V(-PVR)) , SUBLVEL REPLACEMENT
4. (-(PIQ))V((QIR)I(PIR)) *1.01, REPLACEMENT
5. (QIR)I((PIQ)I(PIR)) *2.04, DETACHMENT

PROOF FOUND.

GIVEN *2.05 (AIB)I((CIA)I(CIP))
SUBSTITUTION 5. (QIR)I((PIQ)I(PIR))
GIVEN *2.04 (AI(BIC))I(BI(AIC))
DETACHMENT *2.06 (PIQ)I((QIR)I(PIR))
Q.E.D.

EFFORT LIMIT 200000 ACTUAL 46665
SUBPROBLEMS LIMIT 50 ACTUAL 5
SUBSTITUTIONS LIMIT 50 ACTUAL 6

REMEMBER PROVED THEOREM

*2.06 (AIB)I((BIC)I(AIC))

-89-

TO PROVE
*2.07 PI(PVP)

PROOF FOUND.

GIVEN	*1.3	BI(AVB)
SUBSTITUTION	*2.07	PI(PVP)
Q.E.D.		

EFFORT	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)V(PIP)

2. (PVP)IP

*1.01, REPLACEMENT

*2.02, DETACHMENT. REJECTED PROBLEM

*1.2, DETACHMENT. REJECTED PROBLEM

*2.07, FORWARD CHAINING

PROOF FOUND.

GIVEN

*1.2 (AVA)IA

SUBSTITUTION

2. (PVP)IP

GIVEN

*2.07 AI(AVA)

FORWARD CHAINING

*2.08 PIP

Q.E.D.

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

REMEMBER PROVED THEOREM

*2.08 AIA

TO PROVE

*2.10 -PVP

1. PIP

*1.01, REPLACEMENT

PROOF FOUND.

GIVEN	*2.08	AIA
SUBSTITUTION	1.	PIP
GIVEN	*1.01	(AIB) . = . (-AVB) DEF.
REPLACEMENT	*2.10	-PVP
Q.E.D.		

EFFORT	LIMIT 200000	ACTUAL 5150
SUBPROBLEMS	LIMIT 50	ACTUAL 1
SUBSTITUTIONS	LIMIT 50	ACTUAL 2

REMEMBER PROVED THEOREM

*2.10 -AVA

TO PROVE

*2.11 PV-P

1. -PVP

*1.4, DETACHMENT

PROOF FOUND.

GIVEN	*2.10	-AVA
SUBSTITUTION	1.	-PVP
GIVEN	*1.4	(AVB)I(BVA)
DETACHMENT	*2.11	PV-P
Q.E.D.		

EFFORT	LIMIT 200000	ACTUAL 6190
SUBPROBLEMS	LIMIT 50	ACTUAL 1
SUBSTITUTIONS	LIMIT 50	ACTUAL 2

REMEMBER PROVED THEOREM

*2.11 AV-A

TO PROVE

*2.12 PI--P
1. -PV--P

*1.01, REPLACEMENT

PROOF FOUND.

GIVEN
SUBSTITUTION
GIVEN
REPLACEMENT
Q.E.D.

*2.11 AV-A
1. -PV--P
*1.01 (AIB).=.(AVB) DEF.
*2.12 PI--P

EFFORT	LIMIT 200000	ACTUAL 8215
SUBPROBLEMS	LIMIT 50	ACTUAL 1
SUBSTITUTIONS	LIMIT 50	ACTUAL 2

REMEMBER PROVED THEOREM

*2.12 AI--A

TO PROVE

*2.13 PV---P

1. ---PVP
5062 ---P
5304 PV---P
5199 (PV---P)V(PV---P)
2. -PI---P

*1.4, DETACHMENT
*1.3, DETACHMENT. REJECTED PROBLEM
*2.08, DETACHMENT. REJECTED PROBLEM
*1.2, DETACHMENT. REJECTED PROBLEM
*2.11, FORWARD CHAINING

PROOF FOUND.

GIVEN	*2.12	AI--A
SUBSTITUTION	2.	-PI---P
GIVEN	*2.11	AV-A
FORWARD CHAINING	*2.13	PV---P
Q.E.D.		

EFFORT	LIMIT 200000	ACTUAL 18263
SUBPROBLEMS	LIMIT 50	ACTUAL 2
SUBSTITUTIONS	LIMIT 50	ACTUAL 3

REMEMBER PROVED THEOREM

*2.13 AV---A

TO PROVE

*2.14 --PIP
1. ---PVP
5187 P
5090 --PIP
5149 (--PIP)V(--PIP)
2. ----PIP
5339 --PIP
3. (--PV--P)IP
4. (BI--P)IP
5. (AV--P)IP
5379 --PIP
6. --P(PVP)
6. --PI(PVP)
7. ---PV(PVP)
5051 PVP
5245 --PI(PVP)
5552 (--PI(PVP))V(--PI(PVP))
8. ----PI(PVP)
5612 --PI(PVP)
9. (--PV--P)I(PVP)
10. (BI--P)I(PVP)
11. (AV--P)I(PVP)
5500 --PIP
5737 --PI(PVP)
5780 --PIP
5653 --PI(PVP)
12. --PI((PVP)V(PVP))
12. --PI((PVP)V(PVP))
13. ---PV((PVP)V(PVP))
5286 (PVP)V(PVP)
4842 --PI((PVP)V(PVP))
5854 (--PI((PVP)V(PVP)))V(--PI((PVP)V(PVP))) *1.2, DETACHMENT. REJECTED PROBLEM
14. ----PI((PVP)V(PVP))
5846 --PI((PVP)V(PVP))
15. (--PV--P)I((PVP)V(PVP))
16. (BI--P)I((PVP)V(PVP))
17. (AV--P)I((PVP)V(PVP))
6031 --PI(PVP)
18. --PI(PV((PVP)V(PVP)))
6059 --PI((PVP)V(PVP))
6092 --PI(PVP)
5995 --PI((PVP)V(PVP))
19. --PI(((PVP)V(PVP))V((PVP)V(PVP))) *1.2, BACKWARD CHAINING
1. ---PVP
6102 --PIP
20. PV---P

*1.01, REPLACEMENT
*2.02, DETACHMENT. REJECTED PROBLEM
*2.08, DETACHMENT. REJECTED PROBLEM
*1.2, 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
*2.08, BACKWARD CHAINING. REJECTED PROBLEM
*1.2, BACKWARD CHAINING
*1.01, REPLACEMENT
*2.02, DETACHMENT. REJECTED PROBLEM
*2.08, DETACHMENT. REJECTED PROBLEM
*1.2, 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
*2.07, BACKWARD CHAINING. REJECTED PROBLEM
*1.4, BACKWARD CHAINING. REJECTED PROBLEM
*1.3, BACKWARD CHAINING. REJECTED PROBLEM
*2.08, BACKWARD CHAINING. REJECTED PROBLEM
*1.2, BACKWARD CHAINING
*1.01, REPLACEMENT
*2.02, DETACHMENT. REJECTED PROBLEM
*2.08, DETACHMENT. REJECTED PROBLEM
*1.2, 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
*2.07, BACKWARD CHAINING. REJECTED PROBLEM
*1.4, BACKWARD CHAINING. REJECTED PROBLEM
*1.3, BACKWARD CHAINING. REJECTED PROBLEM
*2.08, BACKWARD CHAINING. REJECTED PROBLEM
*1.2, BACKWARD CHAINING
*1.01, REPLACEMENT. REJECTED PROBLEM
*1.4, DETACHMENT

PROOF FOUND.

GIVEN	*2.13	AV---A
SUBSTITUTION	20.	PV---P
GIVEN	*1.4	(AVB)I(BVA)
DETACHMENT	1.	---PVP
GIVEN	*1.01	(AIB).=.-(-AVB) DFF.
REPLACEMENT	*2.14	--PIP
Q.E.D.		

-96-

EFFCRT	LIMIT 200000	ACTUAL 181261
SUBPROBLEMS	LIMIT 50	ACTUAL 20
SUBSTITUTIONS	LIMIT 50	ACTUAL 21

REMEMBER PROVED THEOREM

*2.14 --AIA

TO PROVE

*2.15 $\{\neg \text{PIQ}\} \{\neg \text{QIP}\}$

1. $(\neg \text{PVQ}) \{\neg \text{QVP}\}$, SUBLVEL REPLACEMENT
5126 $(\neg \text{PVQ}) \{\neg \text{QVP}\}$, SUBLVEL REPLACEMENT. REJECTED PROBLEM
2. $\{\neg (\neg \text{PIQ})\} \vee \{\neg \text{QIP}\}$
3. $\neg \text{QI} \{(\neg \text{PIQ}) \text{IP}\}$
4. $\neg \text{QIP}$
5. $\neg \{\neg (\neg \text{PIQ}) \{\neg \text{QIP}\}\}$
5539 $\{\neg \text{PIQ}\} \{\neg \text{QIP}\}$
5273 $\{\neg (\neg \text{PIQ}) \{\neg \text{QIP}\}\} \vee \{(\neg \text{PIQ}) \{\neg \text{QIP}\}\}$ *1.01, REPLACEMENT
6. $\{(\text{QIC}) \{\neg \text{PIQ}\}\} \{\neg \text{QIP}\}$ *2.04, DETACHMENT
7. $\{(\text{CI-P}) \{\text{CQI}\}\} \{\neg \text{QIP}\}$ *2.02, DETACHMENT
8. $\{(\text{AV-P}) \{\text{AVQ}\}\} \{\neg \text{QIP}\}$ *2.14, DETACHMENT
9. $\{\neg (\neg \text{PIQ})\} \{\neg \text{QIP}\}$ *2.08, DETACHMENT. REJECTED PROBLEM
6071 $\{\neg \text{PIQ}\} \{\neg \text{QIP}\}$ *1.2, DETACHMENT. REJECTED PROBLEM
10. $\{(\neg \text{PIQ}) \vee (\neg \text{PIQ})\} \{\neg \text{QIP}\}$ *2.06, FORWARD CHAINING
11. $\{\text{BI}(\neg \text{PIQ})\} \{\neg \text{QIP}\}$ *2.05, FORWARD CHAINING
12. $\{\text{AV}(\neg \text{PIQ})\} \{\neg \text{QIP}\}$ *1.6, FORWARD CHAINING
13. $\{\neg \text{PIQ}\} \text{IP}$ *2.12, FORWARD CHAINING
14. $\{\neg \text{PIQ}\} \{\neg (\neg (\neg \text{QIP}))\}$ *2.08, FORWARD CHAINING. REJECTED PROBLEM
5757 $\{\neg \text{PIQ}\} \{\neg \text{QIP}\}$ *1.2, BACKWARD CHAINING
15. $\{\neg \text{PIQ}\} \{\neg (\neg \text{QIP}) \vee (\neg \text{QIP})\}$ *2.08, BACKWARD CHAINING. REJECTED PROBLEM

4. $\neg \text{QIP}$
16. $\neg \text{QVP}$ *1.01, REPLACEMENT
5882 P *2.02, DETACHMENT. REJECTED PROBLEM
17. $\{\neg (\neg \text{QIP})\}$ *2.14, DETACHMENT
6171 $\neg \text{QIP}$ *2.08, DETACHMENT. REJECTED PROBLEM
5396 $\{\neg \text{QIP}\} \vee (\neg \text{QIP})$ *1.2, DETACHMENT. REJECTED PROBLEM
18. $\neg \neg \text{QIP}$ *2.12, FORWARD CHAINING
5396 $\neg \text{QIP}$ *2.08, FORWARD CHAINING. REJECTED PROBLEM
19. $\{\neg \text{QV-Q}\} \text{IP}$ *2.07, FORWARD CHAINING
20. $\{\text{BI-Q}\} \text{IP}$ *2.02, FORWARD CHAINING
21. $\{\text{AV-Q}\} \text{IP}$ *1.3, FORWARD CHAINING
22. $\neg \text{QI--P}$ *2.14, BACKWARD CHAINING
6183 $\neg \text{QIP}$ *2.08, BACKWARD CHAINING. REJECTED PROBLEM
23. $\neg \text{QI(PVP)}$ *1.2, BACKWARD CHAINING

23. $\neg \text{QI(PVP)}$
24. $\neg \text{QV(PVP)}$ *1.01, REPLACEMENT
5836 PVP *2.02, DETACHMENT. REJECTED PROBLEM
25. $\{\neg (\neg \text{QI(PVP)})\}$ *2.14, DETACHMENT
6364 $\neg \text{QI(PVP)}$ *2.08, DETACHMENT. REJECTED PROBLEM
6420 $\{\neg \text{QI(PVP)}\} \vee (\neg \text{QI(PVP)})$ *1.2, DETACHMENT. REJECTED PROBLEM
26. $\neg \neg \text{QI(PVP)}$ *2.12, FORWARD CHAINING
6459 $\neg \text{QI(PVP)}$ *2.08, FORWARD CHAINING. REJECTED PROBLEM
27. $\{\neg \text{QV-Q}\} \text{I(PVP)}$ *2.07, FORWARD CHAINING
28. $\{\text{BI-Q}\} \text{I(PVP)}$ *2.02, FORWARD CHAINING
29. $\{\text{AV-Q}\} \text{I(PVP)}$ *1.3, FORWARD CHAINING
6580 $\neg \text{QIP}$ *2.07, BACKWARD CHAINING. REJECTED PROBLEM
6601 $\neg \text{QI(PVP)}$ *1.4, BACKWARD CHAINING. REJECTED PROBLEM
6563 $\neg \text{QIP}$ *1.3, BACKWARD CHAINING. REJECTED PROBLEM
30. $\neg \text{QI} \{\neg (\neg \text{PVP})\}$ *2.14, BACKWARD CHAINING
6627 $\neg \text{QI(PVP)}$ *2.08, BACKWARD CHAINING. REJECTED PROBLEM
31. $\neg \text{QI} \{\text{PVP}\} \vee \{\text{PVP}\}$ *1.2, BACKWARD CHAINING

NO PROOF FOUND

EFFORT	LIMIT 200000	ACTUAL 206728
SUBPROBLEMS	LIMIT 50	ACTUAL 31
SUBSTITUTIONS	LIMIT 50	ACTUAL 32

TO PROVE

*2.20 PI(PVQ)

- | | | |
|------|---------------------|---|
| 1. | -PV(PVQ) | *1.01, REPLACEMENT |
| 2. | PVQ | *2.02, DETACHMENT |
| 3. | -{-(PI(PVQ))} | *2.14, DETACHMENT |
| 5676 | PI(PVQ) | *2.08, DETACHMENT. REJECTED PROBLEM |
| 5675 | (PI(PVQ))V(PI(PVQ)) | *1.2, DETACHMENT. REJECTED PROBLEM |
| 4. | --PI(PVQ) | *2.12, FORWARD CHAINING |
| 6012 | PI(PVQ) | *2.08, FORWARD CHAINING. REJECTED PROBLEM |
| 5. | (PVP)I(PVQ) | *2.07, FORWARD CHAINING |
| 6. | (BIP)I(PVQ) | *2.02, FORWARD CHAINING |
| 7. | (AVP)I(PVQ) | *1.3, FORWARD CHAINING |

PROOF FOUND.

GIVEN	*1.4	(AVB)I(BVA)
SUBSTITUTION	7.	(CVP)I(PVQ)
GIVEN	*1.3	BI(AVB)
FORWARD CHAINING	*2.20	PI(PVQ)
Q.E.D.		

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

REMEMBER PROVED THEOREM

*2.20 AI(AVB)

TO PROVE
*2.21 -PI(PIQ)
1. -PI(-PVQ)

, SUBLVEL REPLACEMENT

PROOF FOUND.

GIVEN	*2.20	AI(AVB)
SUBSTITUTION	1.	-PI(-PVQ)
GIVEN		DEFINITIONS
SUBLVEL REPLACEMENT	*2.21	-PI(PIQ)
Q.E.D.		

EFFORT	LIMIT 200000	ACTUAL 8021
SUBPROBLEMS	LIMIT 50	ACTUAL 1
SUBSTITUTIONS	LIMIT 50	ACTUAL 2

REMEMBER PROVED THEOREM

*2.21 -AI(AIB)

-100-

TC PROVE

*2.24 PI(-PIQ)
1. PI(--PVQ) , SUBLVEL REPLACEMENT
4942 PI(--PVQ) , SUBLVEL REPLACEMENT. REJECTED PROBLEM
2. -PV(-PIQ) *1.01, REPLACEMENT
5291 -P *2.21, DETACHMENT. REJECTED PROBLEM
3. -PI(PIQ) *2.04, DETACHMENT

PROOF FOUND.

GIVEN	*2.21	-AI(AIB)
SUBSTITUTION	3.	-PI(PIQ)
GIVEN	*2.04	(AI(BIC))I(BI(AIC))
DETACHMENT	*2.24	PI(-PIQ)
Q.E.D.		

EFFORT	LIMIT 200000	ACTUAL 21521
SUBPROBLEMS	LIMIT 50	ACTUAL 3
SUBSTITUTIONS	LIMIT 50	ACTUAL 4

REMEMBER PROVED THEOREM

*2.24 AI(-AIB)

TO PROVE

*3.13 $\neg(P \cdot Q) \vdash \neg(P \vee Q)$
1. $\neg(\neg(\neg P \vee Q)) \vdash \neg(P \vee Q)$

SUBLEVEL REPLACEMENT

PROOF FOUND.

GIVEN	*2.14	--AIA
SUBSTITUTION	1.	$\neg(\neg(\neg P \vee Q)) \vdash \neg(P \vee Q)$
GIVEN		DEFINITIONS
SUBLEVEL REPLACEMENT	*3.13	$\neg(P \cdot Q) \vdash \neg(P \vee Q)$
Q.E.D.		

EFFORT	LIMIT 200000	ACTUAL 13293
SUBPROBLEMS	LIMIT 50	ACTUAL 1
SUBSTITUTIONS	LIMIT 50	ACTUAL 2

REMEMBER PROVED THEOREM

*3.13 $\neg(A \cdot B) \vdash \neg(A \vee B)$

TO PROVE

*3.14 $(\neg P \vee Q) \vdash (\neg(P \vee Q))$
1. $(\neg P \vee Q) \vdash (\neg(\neg(\neg P \vee Q)))$

SUBLEVEL REPLACEMENT

PROOF FOUND.

GIVEN	*2.12	AI--A
SUBSTITUTION	1.	$(\neg P \vee Q) \vdash (\neg(\neg(\neg P \vee Q)))$
GIVEN		DEFINITIONS
SUBLEVEL REPLACEMENT	*3.14	$(\neg P \vee Q) \vdash (\neg(P \vee Q))$
Q.E.D.		

EFFORT	LIMIT 200000	ACTUAL 12753
SUBPROBLEMS	LIMIT 50	ACTUAL 1
SUBSTITUTIONS	LIMIT 50	ACTUAL 2

REMEMBER PROVED THEOREM

*3.14 $(\neg A \vee B) \vdash (\neg(A \vee B))$

TO PROVE
*3.24 $-(P \ast -P)$
1. $-(-(-PV--P))$, SUBLVEL REPLACEMENT
5417 $-(-(-PV--P))$, SUBLVEL REPLACEMENT. REJECTED PROBLEM
2. $-PV--P$ *3.14, DETACHMENT

PROOF FOUND.

GIVEN	*2.11	AV-A
SUBSTITUTION	2.	$-PV--P$
GIVEN	*3.14	$(-AV-B) I(-(A \ast B))$
DETACHMENT	*3.24	$-(P \ast -P)$
Q.E.D.		

EFFORT	LIMIT 200000	ACTUAL 14541
SUBPROBLEMS	LIMIT 50	ACTUAL 2
SUBSTITUTIONS	LIMIT 50	ACTUAL 3

REMEMBER PROVED THEOREM

*3.24 $-(A \ast -A)$

TO PROVE

*4.13 P=--P

1. (P1--P) * (--PIP)
2. -(-(P=--P))
- 5588 P=--P
- 5260 (P=--P) V (P=--P)
3. PIP

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

PROOF FOUND.

GIVEN	*2.08	AIA
SUBSTITUTION	3.	PIP
GIVEN	*2.12	AI--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

TO PROVE

*4.20 P=P

1. (PIP)•(PIP)
2. -(P=P)
- 6082 P=P
- 5765 (P=P)V(P=P)
3. --PIP

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

PROOF FOUND.

GIVEN	*2.14	--AIA
SUBSTITUTION	3.	--PIP
GIVEN	*4.13	A=--A
FORWARD CHAINING	*4.20	P=P
Q.E.D.		

EFFORT	LIMIT 200000	ACTUAL 12404
SUBPROBLEMS	LIMIT 50	ACTUAL 3
SUBSTITUTIONS	LIMIT 50	ACTUAL 4

REMEMBER PROVED THEOREM

*4.20 A=A

TO PROVE

*4.24 $P = (P \cdot P)$

1. $P = (\neg(\neg(P \vee P)))$, SUBLVEL REPLACEMENT
5377 $P = (\neg(\neg(P \vee P)))$, SUBLVEL REPLACEMENT. REJECTED PROBLEM
2. $(PI(P \cdot P)) * ((P \cdot P) IP)$ *4.01, REPLACEMENT
3. $\neg(\neg(P = (P \cdot P)))$ *2.14, DETACHMENT
5457 $P = (P \cdot P)$ *2.08, DETACHMENT. REJECTED PROBLEM
5321 $(P = (P \cdot P)) IV(P = (P \cdot P))$ *1.2, DETACHMENT. REJECTED PROBLEM
4. $PI(P \cdot P)$ *4.20, FORWARD CHAINING
5. $\neg PI(P \cdot P)$ *4.13, FORWARD CHAINING
6. $PI(\neg(\neg(P \cdot P)))$ *2.14, BACKWARD CHAINING
6563 $PI(P \cdot P)$ *2.08, BACKWARD CHAINING. REJECTED PROBLEM
7. $PI((P \cdot P) V(P \cdot P))$ *1.2, BACKWARD CHAINING
4. $PI(P \cdot P)$ *1.01, REPLACEMENT
8. $\neg PV(P \cdot P)$ *2.21, DETACHMENT. REJECTED PROBLEM
6597 $\neg P$ *2.02, DETACHMENT
9. $P \cdot P$ *2.14, DETACHMENT
10. $\neg(\neg(PI(P \cdot P)))$ *2.08, DETACHMENT. REJECTED PROBLEM
5348 $PI(P \cdot P)$ *1.2, DETACHMENT. REJECTED PROBLEM
6056 $(PI(P \cdot P)) V(PI(P \cdot P))$ *2.24, FORWARD CHAINING
11. $(\neg PIB) I(P \cdot P)$ *2.20, FORWARD CHAINING
12. $(PVB) I(P \cdot P)$ *2.12, FORWARD CHAINING. REJECTED PROBLEM
6514 $\neg \neg PI(P \cdot P)$ *2.08, FORWARD CHAINING. REJECTED PROBLEM
6235 $PI(P \cdot P)$ *2.07, FORWARD CHAINING
13. $(PVP) I(P \cdot P)$ *2.02, FORWARD CHAINING
14. $(BIP) I(P \cdot P)$ *1.3, FORWARD CHAINING
15. $(AVP) I(P \cdot P)$ *2.14, BACKWARD CHAINING. REJECTED PROBLEM
5907 $PI(\neg(\neg(P \cdot P)))$ *2.08, BACKWARD CHAINING. REJECTED PROBLEM
6607 $PI(P \cdot P)$ *1.2, BACKWARD CHAINING. REJECTED PROBLEM
5352 $PI((P \cdot P) V(P \cdot P))$ *3.01, REPLACEMENT. REJECTED PROBLEM
9. $P \cdot P$ *2.14, DETACHMENT
16. $\neg(\neg(P \cdot P))$ *2.08, DETACHMENT. REJECTED PROBLEM
5352 $P \cdot P$ *1.2, DETACHMENT. REJECTED PROBLEM
6484 $(P \cdot P) V(P \cdot P)$ *2.14, BACKWARD CHAINING
17. $PI \neg P$

PROOF FOUND.

GIVEN	*2.12	AI--A
SUBSTITUTION	17.	PI--P
GIVEN	*2.14	--AIA
BACKWARD CHAINING	9.	P·P
GIVEN	*2.02	AI(BIA)
DETACHMENT	4.	PI(P·P)
GIVEN	*4.20	A=A
FORWARD CHAINING	*4.24	P=(P·P)
Q.E.D.		

EFFORT	LIMIT 200000	ACTUAL 92331
SUBPROBLEMS	LIMIT 50	ACTUAL 17
SUBSTITUTIONS	LIMIT 50	ACTUAL 18

REMEMBER PROVED THEOREM

*4.24 $A = (A * A)$

TO PROVE

*4.25 $P = (PVP)$

1. $(PI(PVP)) * ((PVP) IP)$
2. $-(-(P=(PVP)))$
- 5528 $P=(PVP)$
- 5797 $(P=(PVP)) V (P=(PVP))$
3. $(P=P) I (PVP)$
4. $PI(PVP)$

- *4.01, REPLACEMENT
- *2.14, DETACHMENT
- *2.08, DETACHMENT. REJECTED PROBLEM
- *1.2, DETACHMENT. REJECTED PROBLEM
- *4.24, FORWARD CHAINING
- *4.20, FORWARD CHAINING

PROOF FOUND.

GIVEN	*2.20	AI(AVB)
SUBSTITUTION	4.	PI(PVP)
GIVEN	*4.20	A=A
FORWARD CHAINING	*4.25	P=(PVP)
Q.E.D.		

EFFORT	LIMIT 200000	ACTUAL 20788
SUBPROBLEMS	LIMIT 50	ACTUAL 4
SUBSTITUTIONS	LIMIT 50	ACTUAL 5

REMEMBER PROVED THEOREM

*4.25 $A = (AVA)$

PROGRAM RAN TO COMPLETION.

IPL-V POST-MORTEM

H0	*1	
	*2	
	*3	
	*4	
H1	J7	
H2	16207 WORDS	879155
H3		
H4	0	
H5	J4	
H6	0	
H7		0
H8	0	
H9	0	
H10	0	
H11	0	
H12	0	
W0	0	
W1	0	
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
	24630	0
W21	28088	1
	24631	1
W22	28088	1
	24632	1
W23	28090	
W24	24633	
W25	24754	16
W26	24756	
	0	
	J0	
W27		
W28		
W29		
W30	24755	1
W31	24759	2

W32 0
W33
24153 0
24152 0
24150 0
24151 0
24156 0
24145 0
24154 0
24155 0
24160 0
24157 0
24159 1
24158 1
32531 32548*
*4.25 A=(AVA)

THE FOLLOWING RESULTED FROM EXECUTING 1W15...

L4 0
=0
9-1
-0
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
-0
9-19
5398* 9-8 0
*0
9-20
5059* 9-9 9-21
-0
9-22
5138* 9-10 9-23
-0

4239*	9-11	9-24 9-25 -0 9-26 10 9-27 V0 9-28
4209*	9-12	9-29 -0 9-30 10 9-31 V0 9-32
4077*	9-13	0 =0 9-33 *0 9-34 10 9-35
4051*	9-14	0 *0 9-36 -0 9-37 V0 9-38
5336*	9-15	0 *425 *424 *420 *413
5037*	9-16	0 *420
5789*	9-17	0 9-39 9-40
5765*	9-18	0 9-41 9-42
5454*	9-19	0 9-43
6035*	9-20	0 9-44 9-45
5173*	9-21	0 *213 *211
5262*	9-22	0 9-46
5013*	9-23	0 *210
5066*	9-24	0 9-47
4270*	9-25	0 *224 *220

		*212
		*208
		*207
		*202
		*13
5109*	9-26	0
		9-48
4368*	9-27	0
		9-49
		9-50
4250*	9-28	0
		9-51
		9-52
4242*	9-29	0
		*214
		*208
		*12
4875*	9-30	0
		9-53
4371*	9-31	0
		9-54
		9-55
4253*	9-32	0
		9-56
		9-57
4181*	9-33	0
		9-58
		9-59
4131*	9-34	0
		9-60
		9-61
4066*	9-35	0
		9-62
		9-63
4204*	9-36	0
		9-64
		9-65
4113*	9-37	0
		9-66
4069*	9-38	0
		9-67
		9-68
5585*	9-39	9-69
5771*	9-40	9-70
6331*	9-41	9-71
6109*	9-42	9-72
5534*	9-43	0
		-0
		9-73
5819*	9-44	9-74
5411*	9-45	0
		-0
		9-75
5010*	9-46	9-76
4983*	9-47	9-77
		-0
		9-78
5217*	9-48	9-79
		*0

		9-80
		-0
		9-81
4300*	9-49	9-82
4367*	9-50	9-83
		10
		9-84
		-0
		9-85
4212*	9-51	9-86
		-0
		9-87
4249*	9-52	9-88
		-0
		9-89
		V0
		9-90
4860*	9-53	9-91
		*0
		9-92
		-0
		9-93
4370*	9-54	9-94
		-0
		9-95
		10
		9-96
		V0
		9-97
4350*	9-55	9-98
		10
		9-99
		-0
		9-100
		V0
		9-101
4278*	9-56	9-102
		-0
		9-103
4276*	9-57	9-104
		-0
		9-105
		V0
		9-106
4202*	9-58	9-107
4165*	9-59	9-108
4133*	9-60	9-109
4112*	9-61	9-110
4061*	9-62	9-111
4046*	9-63	9-112
4116*	9-64	0
		10
		9-113
4178*	9-65	0
		10
		9-114
4109*	9-66	0
		V0
		9-115

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
5417*	9-75	0
		9-119
5136*	9-76	0
		*210
4995*	9-77	0
		*211
5080*	9-78	0
		9-120
5419*	9-79	0
		*221
5608*	9-80	0
		9-121
		9-122
5089*	9-81	0
		9-123
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

	9-129
	9-130
4871*	9-91 0
	*201
5384*	9-92 0
	9-131
	9-132
5143*	9-93 0
	9-133
4828*	9-94 0
	*221
	*204
	*203
	*202
5950*	9-95 0
	9-134
4979*	9-96 0
	9-135
	9-136
4404*	9-97 0
	9-137
	9-138
4889*	9-98 0
	*224
	*221
	*202
4836*	9-99 0
	9-139
	9-140
4835*	9-100 0
	9-141
4407*	9-101 0
	9-142
	9-143
4283*	9-102 0
	*220
	*207
	*15
	*14
	*13
5549*	9-103 0
	9-144
4285*	9-104 0
	*220
	*207
	*14
	*13
5292*	9-105 0
	9-145
4352*	9-106 0
	9-146
	9-147
4205*	9-107 0
	*401
4176*	9-108 0
	*401
4140*	9-109 0
	*301
4139*	9-110 0

		*301
4050*	9-111	0
		*101
4078*	9-112	0
		*101
4210*	9-113	0
		9-148
		9-149
4215*	9-114	0
		9-150
		9-151
4148*	9-115	0
		9-152
		9-153
4076*	9-116	0
		9-154
4075*	9-117	0
		*101
5854*	9-118	9-155
5579*	9-119	9-156
4910*	9-120	0
		-0
		9-157
5604*	9-121	9-158
5683*	9-122	9-159
6080*	9-123	9-160
4883*	9-124	9-161
4947*	9-125	9-162
4838*	9-126	9-163
6122*	9-127	9-164
5373*	9-128	9-165
4284*	9-129	9-166
4274*	9-130	9-167
5473*	9-131	9-168
5542*	9-132	9-169
5038*	9-133	9-170
6130*	9-134	9-171
5125*	9-135	9-172
5113*	9-136	9-173
4401*	9-137	9-174
4403*	9-138	9-175
5070*	9-139	9-176
4973*	9-140	9-177
4969*	9-141	9-178
4406*	9-142	9-179
4412*	9-143	9-180
5659*	9-144	9-181
5802*	9-145	9-182
4345*	9-146	9-183
4324*	9-147	9-184
4213*	9-148	9-185
4199*	9-149	9-186
4214*	9-150	9-187
4220*	9-151	9-188
4137*	9-152	0
		-0
		9-189
4145*	9-153	0
		-0

		9-190
4092*	9-154	9-191
6091*	9-155	0
		*413
5452*	9-156	0
		*324
5114*	9-157	0
		9-192
5397*	9-158	0
		*313
5428*	9-159	0
		*313
5909*	9-160	0
		*214
4948*	9-161	0
		*204
5058*	9-162	0
		*204
4924*	9-163	0
		*203
		*201
5801*	9-164	0
		*314
6079*	9-165	0
		*314
4323*	9-166	0
		*15
4346*	9-167	0
		*15
5690*	9-168	0
		*314
5599*	9-169	0
		*314
4960*	9-170	0
		*212
6110*	9-171	0
		*224
5166*	9-172	0
		*206
		*205
5133*	9-173	0
		*206
		*205
4408*	9-174	0
		*16
4410*	9-175	0
		*16
5160*	9-176	0
		*206
		*205
		*204
4978*	9-177	0
		*206
		*205
		*204
4903*	9-178	0
		*203
4419*	9-179	0
		*16

4421*	9-180	0
		*16
4807*	9-181	0
		*313
5917*	9-182	0
		*313
4356*	9-183	0
		*15
4358*	9-184	0
		*15
4216*	9-185	0
		*401
4218*	9-186	0
		*401
4227*	9-187	0
		*401
4229*	9-188	0
		*401
4134*	9-189	0
		9-193
4154*	9-190	0
		9-194
4097*	9-191	0
		*101
5207*	9-192	9-195
4158*	9-193	9-196
4163*	9-194	9-197
5261*	9-195	0
		*213
4161*	9-196	0
		*301
4168*	9-197	0
		*301

-118-

ACCOUNTING SUMMARY

06/05/63

JOB 8168,LTNEW1,EAS826,5MIN,0,099,C STEFFERUD 1 003650

PHASE	PROGRAM	TIME
1	IPL	04MIN 30SEC
	TOTAL	04MIN 30SEC

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.

A0,B0,----,G0	Free variables
P0,Q0,----,T0	Bound variables
I0,V0,-0,*0,=0,=1	Connectives
Other zeroth symbols	Character symbols (except H0,J0,W0)
/1,/2,----,/14	Substitute character symbols
N1,N2,----,N10	Integer data terms 1,2,----,0
K1-K99	Constants and parameters
L1-L49	Control and working lists
T1-T39	Text lists
Q1-Q49	Attributes of terms and expressions
P1-P99	Routines to operate on terms and expressions
M1-M199	Logic theorist routines
M1---M9	Executives, setup
M10--M19	Methods
M40--M49	Utility measures
M50--M59	Information recorders
M60--M69	Information retrievers
M70--M89	Input-output routines
M90--M99	Limit testers
M110-M119	Match processes
X1-X49	Run housekeeping and executives
X1---X9	Executives
X10--X19	Debugging routines
X20--X29	Debugging lists

LIST OF IPL-V BASIC PROCESSES

* Indicates processes which set H5

General Processes

J0 No operation
J1 Execute (0) after restoring HO
*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 HO
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), subprocess (1)
*J18 Execute subprocess of Gen
*J19 Gen clean up

Working Storage Processes

J2n 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 (0) before symbol in cell (1)
J64 INSERT (0) 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
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 (1) + (2) - (0), leave (0)
J111 (1) - (2) - (0), leave (0)
J112 (1) x (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 Set (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 Rewind 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) to not trace

Print Processes

J150 Print list structure (0)
J151 Print list (0)
J152 Print symbol (0)
J153 Print data term (0) w/o 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
Save for Restart (\$ 20.0)
*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 LW19
J177
J178
J179

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 (0)
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 H0)
P = 2 Output to S (then restore H0)
P = 3 Restore (pop up) S
P = 4 Preserve (push down) S
P = 5 Replace (0) by S
P = 6 Copy (0) 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: $\pm PQ$ SYMB LINK
Decimal integer 1 dddd dddd
Floating point 11 ddddd d *tee*
Alphanumeric 21 aaaaa
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
symbolic
Q = 1 Data; internals
symbolic
Q = 2 Routines; internals
symbolic; reset inter-
nal symbol table
Q = 3 Data; internals sym-
bolic; reset internal
symbol table
Q = 4 Routines; internals
absolute
Q = 5 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
1W19; 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.

A0	FREE VARIABLE -A--.	A000V000
B0	FREE VARIABLE -B--.	B000V000
C0	FREE VARIABLE -C--.	C000V000
D0	FREE VARIABLE -D--.	D000V000
E0	FREE VARIABLE -E--.	E000V000
F0	FREE VARIABLE -F--.	F000V000
G0	FREE VARIABLE -G--.	G000V000
I0	LOGICAL CONNECTIVE -IMPLIES-	I000V000
K0	SYMBOL FOR CHARACTER K.	K000V000
K1	HOLDS -OR-.	K001V000
K2	HOLDS -NOT-.	K002V000
K3	HOLDS -AND-.	K003V000
K4	HOLDS -PROVEN EQUIVALENCE-	K004V000
K5	HOLDS -DEFINITIONAL EQUIVALENCE.	K005V000
K6	HOLDS -IMPLIES-	K006V000
K7	HOLDS DELIMIER SYMBOL FOR DELIMITABLE CHARACTERS.	K007V000
K10	PREVIOUS SUBPROBLEM NUMBER. (DATA TERM)	K010V000
K11	SUBSTITUTION COUNT.	K011V000
K12	EFFORT BASE (AND TOTAL).	K012V000
K20	LIMIT ON NO. OF SUBPROBLEMS.	K020V000
K21	LIMIT ON NO. OF SUBSTITUTIONS.	K021V000
K22	LIMIT ON EFFORT.	K022V000
K30	HOLDS R0 IF PROVED THEOREMS ARE TO BE REMEMBERED.	K030V000
K31	HOLDS Y0 IF REJECTED SUBPROBLEMS ARE TO BE PRINTED.	K031V000
K41	D.T. COLUMN FOR PRINTING METHODS IN PROOF SEQUENCES.	K041V000
K42	D.T. COLUMN FOR PRINTING NAMES IN PROOF SEQUENCES.	K042V000
K43	D.T. COLUMN FOR PRINTING EXPRESSIONS IN PROOF SEQUENCES.	K043V000
K44	D.T. COLUMN FOR PRINTING 'LIMIT'.	K044V000
K45	D.T. COLUMN FOR PRINTING 'ACTUAL'.	K045V000
K46	D.T. COLUMN FOR PRINTING 'REJECTED SUBPROBLEM'.	K046V000
K47	D.T. COLUMN FOR PRINTING NAME OF NEW SUBPROBLEM.	K047V000
K48	D.T. COLUMN FOR PRINTING 'THEOREM, METHOD' OF NEW PROBLEMS.	K048V000
K51	DATA TERM '(.'	K051V000
K52	DATA TERM ')'	K052V000
K53	DATA TERM '..'	K053V000
K54	DATA TERM ','	K054V000
L0	SYMBOL FOR CHARACTER L.	L000V000
L1	LIST OF TRUE EXPRESSIONS. (AXIOMS, DEFINITIONS, THEOREMS)	L001V000
L2	LIST OF FREE VARIABLES.	L002V000
L3	LIST OF UNPROVED EXPRESSIONS. (PROBLEMS)	L003V000
L4	LIST STRUCTURE MAP OF ALL TRUE EXPRESSIONS.	L004V000
L5	HOLDS LIST STRUCTURE MAP OF TRUE EXPRESSIONS.	L005V000
L6	LIST OF SPECIAL METHODS FOR ORIGINAL PROBLEMS.	L006V000
L7	LIST OF REGULAR METHODS FOR ALL PROBLEMS.	L007V000
L8	DESCRIPTION LIST TABLE OF DELIMITABLE CHARACTERS.	L008V000
L9	DESCRIPTION LIST TABLE OF CHARACTER SYMBOLS FOR READING TEXT.	L009V000
L10	LIST OF UNTRIED PROBLEMS.	L010V000
L11	LIST OF FOUND PROBLEMS.	L011V000
M0	SYMBOL FOR CHARACTER M.	M000V000
M1	SINGLE PROBLEM EXECUTIVE FOR PROBLEM (0).	M001V000
M2	MULTIPLE PROBLEM EXECUTIVE FOR LIST L3.	M002V000

M3	SET-UP FOR NEW PROBLEM.	M003V000
M7	APPLY METHODS (1) TO PROBLEM (0), ERASE (1) WHEN THRU.	M007V000
M8	CREATE A LIST OF METHODS FOR PROBLEM (0).	M008V000
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.	M011V000 M011V010 M011V020
M12	SUBSTITUTION METHOD FOR PROBLEM (0). SETS H5+ AND OUTPUTS SUCCESSFUL SUBPROBLEM IF SOLUTION IS FOUND. SETS H5- IF NO OUTPUT.	M012V000 M012V010 M012V020
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.	M013V000 M013V010 M013V020
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.	M014V000 M014V010 M014V020
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.	M015V000 M015V010 M015V020
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).	M016V000 M016V010 M016V020 M016V030 M016V040
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.	M017V000 M017V010 M017V020 M017V030 M017V040
M19	FINISH BUILDING NEW SUBPROBLEM TEX (3) FROM (2) VIA THEOREM (1) BY METHOD(0).	M019V000 M019V010
	H5- MEANS NEW TEX WAS ERASED DUE TO LOW UTILITY.	M019V020
	H5+ MEANS OUTPUT TEX (0) HAS SATISFACTORY UTILITY.	M019V030
M40	TEST MATCH OF TOTAL EXPRESSIONS (0),(1) WITHOUT SUBSTITUTION.	M040V000
M41	TEST MATCH OF SEGMENTS (0),(1) WITHOUT SUBSTITUTION.	M041V000
M42	ADD PROBLEM(0) TO FOUND PROBLEMS LIST IF CAN.	M042V000
	H5- MEANS THIS PROBLEM WAS PREVIOUSLY FOUND.	M042V010
M43	MEASURE UTILITY OF (0). H5+ IF GOOD, H5- IF NO GOOD.	M043V000
M50	ADD TRUE EXPRESSION (0) TO TRUE EXPRESSIONS LIST AND MAP.	M050V000
M51	ADD PROBLEM (0) TO UNTRIED PROBLEMS LIST.	M051V000
M54	ADD EXPRESSION (0) TO MAP OF TRUE EXPRESSIONS.	M054V000
M60	FIND AND REMOVE NEXT UNTRIED SUBPROBLEM ON UNTRIED LIST.	M060V000
	H5- MEANS NO SUBPROBLEM ON LIST AND NO OUTPUT.	M060V010
M62	CREATE A LIST OF FEASIBLE MATCHES FOR SEGMENT (0) FROM MAP (1).	M062V000
M63	CREATE A LIST OF FEASIBLE MATCHES FOR TOTAL EXPRESSION (1) FROM MAP (0).	M063V000 M063V010
M70	PRINT EXPRESSION (0).	M070V000
M71	PRINT PROOF SEQUENCE WITH SUCCESSFUL SUBPROBLEM (0).	M071V000
M72	PRINT FAILURE TO FIND PROOF.	M072V000
M73	ENTER SEGMENT (0).	M073V000
M74	ENTER TOTAL EXPRESSION (0).	M074V000
M75	PRINT NEW SUBPROBLEM (0).	M075V000

M76	ENTER LIST OF DATA TERMS (0).	M076V000
M77	PRINT CURRENT STATUS OF LIMITS.	M077V000
M78	PRINT TO PROVE PROBLEM (0).	M078V000
M79	ENTER NAME OF (0). USE EXTERNAL NAME IF CAN.	M079V000
M80	PRINT PROOF LINE FOR METHOD (0) AND TEX (1).	M080V000
M81	PRINT REJECTED PROBLEM.	M081V000
M82	PRINT 'REMEMBER PROVED THEOREM'.	M082V000
M88	PRINT LIST FORM EXPRESSION (0).	M088V000
M89	READ NEXT LOGIC EXPRESSION FROM UNIT 1W18. H5-MEANS NONE THERE.	M089V000 M089V010
M90	TEST IF A LIMIT HAS BEEN REACHED.	M090V000
M110	MAKE FREE VARIABLES OF TOTAL EXPRESSIONS (0),(1) DISJOINT. SUBSTITUTES NEW VARIABLES IN (1) AS REQUIRED.	M110V000 M110V010
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 SUBSTITUTOR.	M111V000 M111V010 M111V020
M112	EXPAND SUBSTITUTION LIST (0).	M112V000
	REPLACE EXPRESSIONS WITH COMPLETELY SUBSTITUTED LOCAL COPIES.	M112V010
M113	MATCH SEGMENTS (0),(1) WITH SUBSTITUTION AS REQUIRED.	M113V000
	H5+ MEANS OUTPUT (0) IS EXPANDED SUBSTITUTION LIST.	M113V010
M114	MATCH SEGMENTS (0),(1) WITH SUBSTITUTION AS REQUIRED. NO OUTPUT, H5- MEANS THE MATCH FAILED.	M114V000 M114V010
M115	SUBSTITUTE IN SEGMENT (0) FROM SUBSTITUTION LIST (1).	M115V000
M116	CREATE LIST OF FREE VARIABLES IN TOTAL EXPRESSION (0). H5- MEANS NO OUTPUT.	M116V000 M116V010
M117	CREATE LIST OF BOUND VARIABLES IN TOTAL EXPRESSION (0). H5- MEANS NO OUT PUT.	M117V000 M117V010
N0	SYMBOL FOR CHARACTER N.	N000V000
N1	INTEGER DATA TERM = 1	N001V000
N2	INTEGER DATA TERM = 2	N002V000
N3	INTEGER DATA TERM = 3	N003V000
N4	INTEGER DATA TERM = 4	N004V000
N5	INTEGER DATA TERM = 5	N005V000
N6	INTEGER DATA TERM = 6	N006V000
N7	INTEGER DATA TERM = 7	N007V000
N8	INTEGER DATA TERM = 8	N008V000
N9	INTEGER DATA TERM = 9	N009V000
N10	INTEGER DATA TERM = 0	N010V000
O0	SYMBOL FOR CHARACTER O.	O000V000
P0	VARIABLE TERM -P-	P000V000
P2	TEST IF (0) IS A BOUND VARIABLE.	P002V000
P3	CLEAR DESCRIPTIONS OF EXPRESSION (0).	P003V000
P4	GO THRU NOTS OF SEGMENT (0). LEAVE FIRST UNNOTTED SEGMENT. H5- MEANS NO OUTPUT DUE TO FAULTY EXPRESSION.	P004V000 P004V010
P5	TEST IF MAIN CONNECTIVE OF EXPRESSION (0) IS -IMPLIES-.	P005V000
P6	TEST IF CONNECTIVE (0) IS NON-UNARY.	P006V000
P7	TEST IF (0) IS A CONNECTIVE.	P007V000
P8	TEST IF (0) IS VARIABLE TERM.	P008V000
P9	TEST IF (0) IS FREE VARIABLE.	P009V000
P12	FIND MAIN EXPRESSION OF TOTAL EXPRESSION (0).	P012V000
P13	FIND LEFT SEGMENT OF TOTAL EXPRESSION (0).	P013V000

P14	FIND RIGHT SEGMENT OF TOTAL EXPRESSION (0).	P014V000
P15	TEST IF TOTAL EXPRESSION (0) IS IN TREE FORM.	P015V000
P16	FIND MAIN CONNECTIVE OF TOTAL EXPRESSION (0).	P016V000
P17	COPY SEGMENT (0). IF (0) IS A VARIABLE, OUTPUT THE INPUT.	P017V000
P18	TEST IF (0) IS A CHARACTER SYMBOL.	P018V000
P19	GET APPROPRIATE CHARACTER SYMBOL FOR (0).	P019V000
P20	MAKE LEFT SUBEXPRESSION OF EXPRESSION (0) INTO EXPRESSION, H5- MEANS NO OUTPUT DUE TO FAULTY EXPRESSION.	P020V000 P020V010
P21	MAKE RIGHT SUBEXPRESSION OF EXPRESSION (0) INTO EXPRESSION, H5- MEANS NO OUTPUT DUE TO FAULTY EXPRESSION.	P021V000 P021V010
P22	CREATE NEW PROBLEM WITH SEGMENT (0) AS LEFT SUBEXPRESSION, SEGMENT (1) AS RIGHT SUBEXPRESSION, IMPLIES AS CONNECTIVE.	P022V000 P022V010
P23	ERASE MADE EXPRESSION (0).	P023V000
P24	MAKE SEGMENT (0) INTO A NEW TOTAL EXPRESSION.	P024V000
P25	COPY TEX (0) FOR SUBSTITUTION.	P025V000
P26	GENERATE LOCATIONS OF NON-VARIABLE SEGMENTS FROM TOTAL EXPRESSION (1) AT THE LEVEL OF DATA TERM(2) FOR PROCESS (0).	P026V000 P026V010 P026V020
P27	REPLACE BOUND VARIABLES BY FREE VARIABLES IN TEX (0).	P027V000
P28	GENERATE LOCATIONS OF FREE VARIABLES IN SEGMENT (1) FOR (0).	P028V000
P29	GENERATE LOCATIONS OF BOUND VARIABLES IN SEGMENT (1) FOR (0).	P029V000
P30	CREATE LIST OF FREE VARIABLES IN EXPRESSION (0).	P030V000
P31	CREATE LIST OF BOUND VARIABLES IN EXPRESSION (0).	P031V000
P50	CONVERT LOGIC EXPRESSION (0) TO INTERNAL TREE FORM IF IN EXTERNAL FORM. NO OUTPUT. H5- MEANS NO CONVERSION. ENTIRE EXPRESSION MUST BE PARENTHESIZED.	P050V000 P050V010 P050V020
P51	REPLACE ALL DELIMITED SYMBOLS IN EXPRESSION (0) IF (0) IS IN LIST FORM.	P051V000 P051V010
P52	CREATE A TREE FORM MAIN SEGMENT FROM LIST FORM EXPRESSION (0). H5- MEANS NO OUTPUT DUE TO A FAULTY INPUT EXPRESSION.	P052V000 P052V010
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.	P055V000 P055V010 P055V020
Q0	VARIABLE TERM -Q-	Q000V000
Q1	FIND CONNECTIVE OF SEGMENT (0).	Q001V000
Q2	FIND NO. OF LEVELS OF EXPRESSION (0).	Q002V000
Q3	FIND NO. OF DISTINCT VARIABLES OF EXPRESSION (0).	Q003V000
Q4	FIND NO. OF VARIABLE PLACES OF EXPRESSION (0).	Q004V000
Q5	ATTRIBUTE--VARIABLE TERM.	Q005V000
Q6	ATTRIBUTE--FREE VARIABLE.	Q006V000
Q7	ATTRIBUTE--EXTERNAL NAME.	Q007V000
Q8	FIND PROBLEM NO. OF EXPRESSION (0).	Q008V000
Q9	ATTRIBUTE--BOUND VARIABLE.	Q009V000
Q10	FIND PROBLEM EXPRESSION (0) DERIVED FROM.	Q010V000
Q11	FIND METHOD OF DERIVATION FOR EXPRESSION (0).	Q011V000
Q12	FIND THEOREM USED TO DERIVE PROBLEM (0).	Q012V000
Q13	FIND PROVING THEOREM FOR PROBLEM (0).	Q013V000
Q14	FIND TYPE OF CONNECTIVE (0).	Q014V000
Q15	ATTRIBUTE--INTERNAL FORM.	Q015V000
Q16	FIND EXTERNAL NAME OF (0) IN TABLE T10.	Q016V000
Q17	FIND CURRENT LEVEL OF SUBSEGMENT REPLACEMENT OF PROBLEM (0).	Q017V000

Q18	FIND SUFFIX OF EXPRESSION (0).	Q018V000
Q19	FIND APPROPRIATE CHARACTER SYMBOL FOR (0).	Q019V000
R0	VARIABLE TERM -R-	R000V000
S0	VARIABLE TERM -S-	S000V000
T0	VARIABLE TERM -T-	T000V000
T1	TEXT LIST 'GIVEN'	T001V000
T2	TEXT LIST 'PROOF FOUND'	T002V000
T3	TEXT LIST 'SUBSTITUTION'	T003V000
T4	TEXT LIST 'Q.E.D.'	T004V000
T5	TEXT LIST OF 5 BLANK CHARACTERS.	T005V000
T6	TEXT LIST 'NO PROOF FOUND'	T006V000
T7	TEXT LIST 'EFFORT'	T007V000
T8	TEXT LIST 'SUBPROBLEMS'	T008V000
T9	TEXT LIST 'SUBSTITUTIONS'	T009V000
T10	DESCRIPTION LIST TABLE OF NAMES.	T010V000
T12	TEXT LIST 'DETACHMENT'	T012V000
T13	TEXT LIST 'REPLACEMENT'	T013V000
T14	TEXT LIST 'FORWARD CHAINING'	T014V000
T15	TEXT LIST 'BACKWARD CHAINING'	T015V000
T16	TEXT LIST 'SUBLEVEL REPLACEMENT'	T016V000
T19	TEXT LIST 'REJECTED PROBLEM'	T019V000
T20	TEXT LIST 'ACTUAL'	T020V000
T21	TEXT LIST 'LIMIT'	T021V000
T22	TEXT LIST 'TO PROVE'	T022V000
T23	TEXT LIST 'REMEMBER PROVED THEOREM'	T023V000
T24	TEXT LIST 'BAD EXPRESSION'	T024V000
U0	SYMBOL FOR CHARACTER U.	U000V000
V0	LOGICAL CONNECTIVE -OR-	V000V000
X0	SYMBOL FOR CHARACTER X.	X000V000
X1	RUN EXECUTIVE	X001V000
X9	SAVE AND CONTINUE EXECUTIVE.	X009V000
X21	LIST OF ROUTINES TO BE MARKED TO TRACE.	X021V000
X22	LIST OF ROUTINES TO BE MARKED TO PROPAGATE TRACE.	X022V000
X23	DESCRIPTION LIST OF TRAP ACTIONS. (ATTRIBUTE/VALUE PAIRS)	X023V000
Y0	SYMBOL FOR CHARACTER Y.	Y000V000
Z0	SYMBOL FOR CHARACTER Z.	Z000V000
=0	LOGICAL CONNECTIVE -PROVEN EQUIVALENCE-	=000V000
=1	LOGICAL CONNECTIVE -DEFINITIONAL EQUIVALENCE-	=001V000
-0	LOGICAL CONNECTIVE -NOT-	-000V000
*0	LOGICAL CONNECTIVE -AND-	*000V000
'0	SYMBOL FOR QUOTE MARK.	'000V000
.0	SYMBOL FOR PERIOD.	.000V000
)0	SYMBOL FOR RIGHT PAREN.)000V000
/0	SYMBOL FOR SLASH.	/000V000
/1	SYMBOL FOR DIGIT 1.	/001V000
/2	SYMBOL FOR DIGIT 2.	/002V000
/3	SYMBOL FOR DIGIT 3.	/003V000
/4	SYMBOL FOR DIGIT 4.	/004V000
/5	SYMBOL FOR DIGIT 5.	/005V000
/6	SYMBOL FOR DIGIT 6.	/006V000
/7	SYMBOL FOR DIGIT 7.	/007V000

/8	SYMBOL FOR DIGIT 8.	/008V000
/9	SYMBOL FOR DIGIT 9.	/009V000
/10	SYMBOL FOR DIGIT 0.	/010V000
/11	SYMBOL FOR CHARACTER H.	/011V000
/12	SYMBOL FOR CHARACTER J.	/012V000
/13	SYMBOL FOR CHARACTER W.	/013V000
/14	DUMMY CHARACTER SYMBOL WITH EXTERNAL NAME '/UGH/'	/014V000
/16	DUMMY EXPRESION TO SUPPLY TEXT 'THE DEFINITIONS'	/016V000
,0	SYMBOL FOR COMMA.	,000V000
(0	SYMBOL FOR LEFT PAREN.	(000V000
+0	SYMBOL FOR PLUS SIGN.	+000V000
\$0	SYMBOL FOR DOLLAR SIGN.	\$000V000

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.

JOB	8168,LTNEW1,EAS826,5MIN,0,099,C	STEFFERUD
ASSIGN	A6=SYSAR2	
ASSIGN	B6=SYSAR3	
IPL		
LOGIC THEORIST (IPL-V)	9	-
	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 I	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	5		R -
M1 SINGLE PROBLEM EXECUTIVE FOR PROBLEM (0). H5 + IF SUCCEEDS.		M1	M001R000
		40W0	M001R010
		60WC	M001R020
		M43	IWO=PROB M001R030
TEST UTILITY. IF NO GOOD, QUIT.		709-4	M001R040
		11W0	M001R050
PRINT 'TO PROVE' PROBLEM IWO.		M78	M001R060
TRY SUBSTITUTION. IF WORKED, PRINT PROOF.		11W0	M001R070
	9-1	M12	M001R080
CREATE LIST OF METHODS FOR PROB.		70	M001R090
APPLY METHODS. IF PROOF FOUND, PRINT IT.		11W0	M001R100
TEST IF ANY LIMITS EXCEEDED.		M8	M001R110
IF YES, QUIT.		11W0	M001R120
FIND NEXT SUBPROBLEM.		M7	M001R130
IF NONE, QUIT.		70	M001R140
IF ONE,		M90	M001R150
PRINT SUBPROBLEM, TRY METHODS.		70	M001R160
PRINT PROOF FROM (0).	9-2	709-3	M001R170
AND QUIT +.		60WC	M001R180
PRINT FAILURE, QUIT -.	9-3	M60	M001R190
	9-4	709-3	M001R200
PRINT REJECTED PROBLEM AND QUIT -.	9-5	60WC	M001R210
		M70	M001R220
M2 MULTIPLE PROBLEM EXECUTIVE.	1	M71	M001R230
GENERATE PROBLEMS FOR PROOF.		30W0	M001R240
CONVERT TO INTERNAL (TREE) FORM. IF FAILED, TAKE ERROR ACTION.		J4	M001R250
TEST IF REMEMBERING.		9-5	M001R260
INPUT PROBLEM FOR M1.		30W0	R
TRY FOR PROOF TO BE REMEMBERED, H5- MEANS NO PROOF FOUND.		J3	M002R000
PRINT 'REMEMBER PROVED THEOREM'		M2	M002R010
ADD TO TRUE EXPRESSIONS LIST.		10L3	M002R020
TRY FOR PROOF TO BE FORGOTTEN.		109-100	M002R030
CLEAN UP W0, H5+ FOR GEN.		J100	M002R040
		0	M002R050
		J50	M002R060
		11W0	M002R070
		P50	M002R080
		709-103	M002R090
		11K30	M002R100
		10R	M002R110
		J2	M002R120
		11W0	M002R130
		709-101	M002R140
		M1	M002R150
		709-102	M002R160
		11W0	M002R170
		M82	M002R180
		M50	
		9-102	
	9-101	M1	
	9-102	30W0	
	9-103	J4	
		11W0	
		M88	
		9-102	

			R	
M3 SET UP FOR NEW PROBLEM.	1 M3	10L10 J75	M003R000 M003R010	
CLEAR UNTRIED PROBLEMS LIST.		J72 10L11 J75	M003R020 M003R030 M003R040	
CLEAR FOUND PROBLEMS LIST.		J72 10K10	M003R050 M003R060	
CLEAR PREVIOUS PROBLEM NUMBER.		J124 50K11	M003R070 M003R080	
CLEAR SUBSTITUTIONS COUNT.		J124 50H3 10K12	M003R090 M003R100 M003R110	
SET EFFORT BASE.		J121 J8	M003R120	
	1		R	
M7 APPLY METHODS (1) TO PROBLEM (0), ERASE LIST (1) WHEN THRU. H5+ MEANS OUTPUT (0) IS SOL'N. H5- MEANS NO OUTPUT, NO SOL'N.	M7	J50 40H0 109-100 J100 J5 30W0	M007R000 M007R010 M007R020 M007R030 M007RC35 M007R040	
IF NO PROOF, ERASE METHOD LIST. ELSE SAVE PROOF AND ERASE LIST.		70J71 J6	M007R045 M007R050	
9-100 SUBPROCESS, APPLY METHOD (0) TO PROBLEM 1W0.	9-100	11W0 J6	M007R060 M007R070	
REVERSE H5 FOR GENERATOR.		J1 J5	M007R080	
	1		R	
M8 CREATE METHOD LIST FOR (0). COPY REGULAR LIST.	M8	10L7 J73 J50	M008R000 M008R010 M008R020	
TEST IF ORIGINAL PROBLEM. IF NOT, NO SPECIAL METHODS. IF YES,		J130 709-1 11W0 10L6	M008R030 M008R040 M008R050 M008R070	
COPY SPECIAL LIST, INSERT AFTER HEAD OF REGULAR LIST.		J73 J76	M008R080 M008R090	
SET UP OUTPUT AND QUIT.	9-1	51W0 11W0	J30 J30	M008R100 M008R110
	1		R	
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	M11	J45 60W1 J81 70J35 60W0 50L5 11K6	M011R000 M011R010 M011R020 M011R030 1W0=PRBMEXM011R040 1L5=L4 1K6=IMPLY M011R070 M011R080	
FIND IMPLIES MAPS.		J10 70J35	M011R090	
FIND MAP OF RIGHT SIDES OF MAIN CONNECTIVE IMPLIES.		J82 70J35 11W1	M011R100 M011R110	

GET LIST OF FEASIBLE THEOREMS.	J6	M011R120		
SAVE LIST FOR ERASURE.	M63	M011R130		
GENERATE FEASIBLE THEOREMS.	60W5	M011R140		
CLEAN UP AND REVERSE H5 AFTER GENERATOR.	109-100	M011R150		
9-100 SUBPROCESS, TRY PROOF WITH THEOREM (0).	J100	M011R160		
MAKE FREE VARIABLES DISJOINT FIND RIGHT SUBSEGMENT OF THM	11W5	1W5=THMLSTM011R170		
MATCH, MAKE THM RIGHT LIKE PROB.	J71	M011R180		
FIND THM LEFT SUBSEGMENT.	J5	J35	M011R190	
MAKE NEW TEX FROM COPY OF THMLFT.	9-100	60W2	1W2=THM	M011R200
SUBSTITUTE INTO NEWTEX FROM (1)	11W1	M011R210		
ERASE SUBSTL.	11W2	M011R220		
ASSIGN DERIVATION	M11C	M011R230		
COMPLETE NEW SUBPROBLEM DESCRIPTION AND MEASURE UTILITY.	P14	M011R240		
H5- MEANS NEW PROBLEM WAS ERASED.	70J4	M011R250		
TRY SUBSTITUTION, OUTPUT SOL'N.	11W0	M011R260		
M12 SUBSTITUTION METHOD FOR PROBLEM (0). H5+ MEANS OUTPUT (0) IS A SOLUTION	M113	(1)=THMRT	M011R270	
H5- MEANS NO OUTPUT.	70J4	M011R280		
TALLY SUBSTITUTION COUNTER.	60W3	1W3=SUBLSTM011R290		
GET A LIST OF FEASIBLE THEOREMS.	11W2	M011R300		
SAVE THE LIST FOR ERASURE.	P13	M011R310		
GENERATE THEOREMS.	709-101	(0)=THMLFTM011R320		
ERASE LIST OF THEOREMS,	P17	M011R340		
	P24	M011R345		
	60W4	1W4=NEWTEXM011R350		
	M115	M011R360		
	9-101	M011R370		
	11W4	NEWPROMLEM011R390		
	11W1	PROBLEM M011R400		
	11W2	THEOREM M011R410		
	10M11	M011R420		
	M19	M011R430		
	70J4	M011R440		
	M12	M011R460		
	9-101	M011R470		
	11W3	M011R480		
	J72	R		
	J4	M012R000		
1	M12	1W1=PROB M012R010		
	J43	M012R020		
	60W1	M012R030		
	J81	1W0=PRBMEXM012R040		
	70J33	M012R050		
	60W0	M012R060		
	5CK11	M012R070		
	J125	M012R080		
	51W1	M012R090		
	10L4	1W3=THMLSTM012R100		
	M63	M012R110		
	60W3	M012R120		
	109-100	M012R130		
	J100	M012R140		
	11W3			
	J71			

CLEAN-UP AND REVERSE H5.		J33	J5	M012R150
9-100 SUBPROCESS, TRY PROBLEM MEX 1WO WITH THEOREM (0).	9-100	60W2	1W2=THM	M012R160
		11W1		M012R170
		11W2		M012R180
MAKE FREE VARIABLES DISJOINT.		M110		M012R190
FIND MAIN EXPRESSION OF THEOREM.		J81		M012R200
		70J4		M012R210
		11W0		M012R220
TEST FOR MATCH.		M114		M012R230
IF H5-, SET + FOR GENERATOR,		70J4		M012R240
		11W1		M012R250
		11W2		M012R260
OTHERWISE ASSIGN PROOF AND OUTPUT SUCCESSFUL PROBLEM 1W1 AS SOLUTION. QUIT WITH H5- FOR GEN.		10Q13		M012R270
		J11		M012R280
	1	11W1	J3	M012R290
				R
M13 REPLACEMENT METHOD FOR PROB (0)	M13	J46		M013R000
ADD NEW SUBPROBLEMS TO UNTRIED LIST IF CAN.		60W1	1W1=PROB	M013R010
H5+ MEANS OUTPUT (0) IS A SOLUTION.		J81		M013R020
H5- MEANS NO SOL'N, NO OUTPUT.		70J36		M013R030
		60W0	1W0=PRBMEXM013R040	
FIND DEFINITIONAL EQUIVALENCE MAPS.		50L5	1L5=L4	M013R050
		11K5	1K5=DEFEQ	M013R060
		J10		M013R070
		70J36		M013R080
		60W5	1W5=DEFMPMSM013R090	
FIND MAP OF LEFT SIDES OF DEFEQ.		J81		M013R100
		709-10		M013R110
		11W1		M013R120
		J6		M013R130
GET LIST OF FEASIBLE DEFINITIONS. SAVE LIST FOR ERASURE.		M63		M013R140
TRY LEFT SIDES.		60W6	1W6=FSBLS	M013R150
GENERATE FEASIBLE DEFINITIONS.		109-100		M013R160
		J100		M013R170
ERASE FEASIBLES LIST.		11W6		M013R180
REVERSE H5 FROM GENERATOR.		J71		M013R190
QUIT IF PROOF IN H0, OR TRY RIGHTS.	9-10	709-10	J36	M013R200
		11W5		M013R210
FIND MAP OF RIGHT SIDES OF DEFEQ.		J82		M013R220
		70J36		M013R230
		11W1		M013R240
		J6		M013R250
GET LIST OF FEASIBLE DEFINITIONS. SAVE LIST.		M63		M013R260
TRY RIGHT SIDES.		60W6	1W6=FSBLS	M013R270
GENERATE FEASIBLE DEFINITIONS.		109-200		M013R280
		J100		M013R290
ERASE FEASIBLES LIST.		11W6		M013R300
REVERSE H5, CLEAN UP AND QUIT.		J71		M013R310
TRY LEFT SIDE OF (0) W/1WO.	9-100	50L5	J36	M013R320
		60W2	1W2=DEF	M013R330
				M013R340

MAKE FREE VARIABLES DISJOINT.	11W1	M013R350
FIND LEFT SUB SEG OF DEF.	11W2	M013R360
	M110	M013R370
	P13	M013R380
	70J4	M013R390
	11W0	M013R400
	M113	M013R410
	70J4	M013R420
	60W3	1W3=SUBLT M013R430
	11W2	M013R440
	P14	M013R445
	70	9-300
	J72	J4
	9-200	1W2=DEF
	60W2	M013R460
	11W1	M013R470
	11W2	M013R480
	M110	M013R490
	P14	M013R500
	70J4	M013R510
	11W0	M013R520
	M113	M013R530
	70J4	M013R540
	60W3	1W3=SUBLT M013R550
	11W2	M013R560
	P13	M013R565
	70	9-300
	J72	J4
	9-300	1W4=NEWTEX M013R600
	P17	M013R610
	P24	M013R620
	60W4	M013R630
	M115	M013R640
	9-301	M013R650
	11W4	M013R660
	11W1	M013R670
	11W2	M013R680
ASSIGN DERIVATION, ADD FOUND LIST,	10M13	M013R700
MEASURE UTILITY, ERASE IF NO GOOD.	M19	M013R710
	70J4	M013R720
TRY SUBSTITUTION, H5+ OUTPUT PROOF.	M12	R
	9-301	1W1=PROB
ERASE SUBSTITUTION LIST.	11W3	M014R000
	J72	M014R010
	J4	M014R020
	1	M014R030
M14 FORWARD CHAINING METHOD FOR	M14	M014R040
PROBLEM (0). ADDS NEW SUBPROBS	J44	M014R050
TO UNTRIED LIST IF CAN.	60W1	M014R060
H5+ MEANS OUTPUT (0) IS	P13	M014R070
SOLUTION.	70J34	M014R080
H5- MEANS NO SOLUTION, NO OUTPT	609-1	
FIND MAIN CONNECTIVE OF PROB.	51W1	
	P16	
	70J34	
	10L5	

FIND APPROPRIATE THEOREM MAPS.	J6		M014R090	
	J10		M014R100	
	70J34		M014R110	
FIND MAP OF LEFT SIDES.	J81		M014R120	
	70J34		M014R130	
INPUT FAKE TEX.	109-0		M014R140	
	J6		M014R150	
GET FEASIBLE THEOREM LEFT SIDES.	M63		M014R160	
TRY FEASIBLE THM LEFTS WITH PROBLEM LEFT. GENERATE FSBLS.	60W0	IWO=FSBLS	M014R170	
	109-100		M014R180	
	J100		M014R190	
	11W0		M014R200	
ERASE LIST OF FEASIBLES.	J71		M014R210	
	J5	J34	M014R220	
FAKE TEX . . . 9-1 HOLDS MEX. 9-100 SUBPROCESS, TRY LEFT SIDES.	9-0	0	9-1	M014R230
	9-100	60W2	IW2=THM	M014R240
		11W1	IW1=PROB	M014R250
		11W2		M014R260
MAKE FREE VARIABLES DISJOINT.	M110		M014R270	
FIND LEFT SEGMENT OF THM TEX.	P13		M014R280	
	70J4		M014R290	
INPUT PROB LEFT MATCH, OUTPUT LIST OF SUBSTITUTIONS WILL MAKE THM LIKE PROB IF CAN.	119-1		M014R300	
	M113		M014R310	
	70J4		M014R320	
	60W3	IW3=SUBSTL	M014R330	
	51W1		M014R340	
FIND RIGHT SIDE OF PROB	P14		M014R350	
	709-101		M014R360	
	11W2		M014R370	
FIND RIGHT SIDE OF THEOREM.	P14		M014R380	
	709-102		M014R390	
CREATE NEW TEX WITH COPIES.	P22		M014R400	
THM LEFT, PROB RIGHT.	60W4	IW4=NEWTEX	M014R405	
SUBSTITUTE INTO NEW TEX.	11W3		M014R410	
ERASE SUBSTL.	J6		M014R420	
	M115		M014R430	
	9-101		M014R510	
	11W4		M014R520	
	11W1		M014R530	
	11W2		M014R540	
ASSIGN DERIVATION, ADD TO FOUND LIST	10M14		M014R550	
MEASURE UTILITY, ERASE IF NO, GOCD.	M19		M014R560	
TRY SUBSTITUTION, H5+ OUTPUT PROOF.	70J4		M014R570	
	M12	J5	M014R590	
	9-102	30HC	M014R600	
ERASE SUBSTITUTION LIST.	9-101	11W3	M014R610	
		J72	M014R620	
		J4	R	
M15 BACKWARD CHAINING METHOD FOR PROBLEM (0). ADDS NEW SUBPROBS TO UNTRIED LIST IF CAN.	M15	J44	M015R000	
		60W1	IW1=PROB	M015R010
		P14		M015RC20

H5+ MEANS OUTPUT (0) IS A SOLUTION.	70J34	M015R030
H5- MEANS NO SOLUTION, NO OUTPT	609-1	9-0=FAKTEXM015R040
FIND APPROPRIATE MAPS.	5CL5	L5 IS MAP M015R050
FIND MAP OF RIGHT SIDES.	1IK6	1K6=IMPLY M015R060
GET FEASIBLE THEOREM RIGHT SIDES.	J10	M015R070
TRY FEASIBLE THM RIGHTS WITH PROBLEM RIGHT. GENERATE FSBLS.	70J34	M015R080
ERASE LIST OF FEASIBLES.	J82	M015R090
FAKE TEX . . . 9-1 HOLDS MEX. 9-100 SUBPROCESS, TRY RIGHT SIDES.	70J34	M015R100
MAKE FREE VARIABLES DISJOINT. FIND RIGHT SEGMENT OF THM TEX.	109-0	M015R110
INPUT PROB RIGHT. MATCH, OUTPUT LIST OF SUBSTITUTIONS WILL MAKE THM LIKE PROB IF CAN.	J6	M015R120
FIND LEFT SIDE OF THEOREM.	M63	M015R130
FIND LEFT SIDE OF PROBLEM.	60W0	1W0=FSBLS M015R140
CREATE NEW TEX WITH COPIES.	109-100	M015R150
PROB ON LEFT, THM ON RIGHT.	J100	M015R160
SUBSTITUTE INTO NEW TEX. ERASE SUBSTL.	11W0	1W2=THM M015R170
ASSIGN DERIVATION, ADD TO FOUND LIST MEASURE UTILITY, ERASE IF NO GOOD.	J71	M015R180
TRY SUBSTITUTION, H5+ OUTPUT PROOF.	J5	M015R190
ERASE SUBSTITUTION LIST.	9-0	M015R200
	9-1	1W2=THM M015R210
	9-100	M015R220
	60W2	M015R230
	11W1	M015R240
	11W2	M015R250
	M110	M015R260
	P14	M015R270
	70J4	M015R280
	119-1	M015R290
	M113	1W3=SUBLT M015R300
	70J4	M015R310
	60W3	M015R320
	51W2	M015R330
	P13	M015R340
	709-101	M015R350
	11W1	M015R360
	P13	M015R370
	709-102	1W4=NEWTEX M015R375
	P22	M015R380
	60W4	M015R390
	11W3	M015R400
	J6	M015R480
	M115	M015R490
	9-101	M015R500
	11W4	M015R510
	11W1	M015R520
	11W2	M015R530
	10M15	M015R540
	M19	M015R560
	70J4	M015R570
	M12	M015R580
	J5	M015R590
	9-102	R
	30HC	
	9-101	
	11W3	
	J72	
	J4	

M16 SUBLVEL REPLACEMENT METHOD FOR PROBLEM (0). ADD ALL NEW SUBPROBLEMS TO UNTRIED LIST.	M16	J48 60WC 50L5 11K5 J10 70J38 60W2 J81 70J38 60W1 51W2 J82 70J38 60W2 51W0 40HO 10Q17 J14 Q17 70J38 20W3 9-3 70J38 9-10 11W0 P25 60W4 10J3 60W5 509-100 P26 01W5 70 9-1 9-20 709-2 J120 10Q17 11W4 J11 11W3 11W4 9-11 9-1 9-2 11W4 J72 J38 11W4 11W0 10/16 10M16 M19 709-3 M12 709-3 J38 10N2	IW0=PROB 1W1=LFTMAP 1W2=RTMAP 1W3=CURLEV 1W4=CPROB 1W5=FLAG	M016R000 M016R010 M016R020 M016R030 M016R040 M016R050 M016R060 M016R070 M016R080 M016R090 M016R100 M016R110 M016R120 M016R130 M016R140 M016R142 M016R145 M016R148 M016R150 M016R160 M016R165 M016R170 M016R175 M016R180 M016R190 M016R200 M016R210 M016R220 M016R230 M016R240 M016R250 M016R260 M016R270 M016R280 M016R290 M016R300 M016R310 M016R320 M016R330 M016R340 M016R350 M016R360 M016R370 M016R380 M016R390 M016R400 M016R410 M016R420 M016R430 M016R440 M016R470
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,	9-3 9-10	9-20 70J38 11W0 P25 60W4 10J3 60W5 509-100 P26 01W5 70 9-1 9-20 709-2 J120 10Q17 11W4 J11 11W3 11W4 11W4 J72 J38 11W4 11W0 10/16 10M16 M19 709-3 M12 709-3 J38 10N2		
COPY PROBLEM FOR REPLACEMENT.				
SET 'NEW SUBPROBLEM FLAG' TO NO.	9-11			
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,	9-2	9-11 11W4 11W4 J72 J38 11W4 11W0 10/16 10M16 M19 709-3 M12 709-3 J38 10N2		
ERASE LEFTOVER COPY, QUIT.				
SET UP DERIVATION ASSOCIATIONS, AND	9-1			
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.	11W3	M016R480
TEST IF 1W3 GREATER THAN N2.	J115	M016R490
IF NO, QUIT SUBPROCESS--.	700	NO OUTPUT M016R500
IF YES,	10N1	M016R510
SUBTRACT 1, QUIT SUBPROCESS+.	11W3	M016R520
9-100 SUBPROCESS FOR SUBSEGMENT REPLACEMENT IN LOCATION (0).	9-100 60W6	J111 OUTPUT 1W3M016R530 1W6=SEGLOCM016R540
CREATE LIST OF FEASIBLE DEFs.	51W1	M016R550
SAVE LIST FOR LATER ARASURE.	12W6	M016R560
GENERATE DEFs FOR LEFT SIDE MATCH.	M62	M016R570
IF MATCHED, GO CLEAN UP.	40H0	M016R580
IF FAILED, ERASE OLD LIST,	109-200	M016R590
CREATE NEW LIST OF FEASIBLE DEFs.	J100	M016R600
SAVE LIST FOR LATER ERASURE.	709-101	M016R610
GENERATE DEFs FOR RIGHT SIDE MATCH.	J71	M016R620
ERASE LIST AND QUIR + FOR GEN.	11W2	M016R630
9-200 SUBPROCESS, TRY REPLACEMENT BY MATCHING SEGMENT TO LEFT SIDES.	12W6	M016R640
MAKE FREE VARIABLES DISJOINT.	M62	M016R650
FIND LEFT SEGMENT OF DEF.	40H0	M016R660
IF NONE, QUIT + FOR GEN.	109-300	M016R670
IF FOUND,	J100	M016R680
MATCH SEGMENT TO LEFT SIDE.	9-101 J71 J4	M016R690
IF NO MATCH, QUIT + FOR GEN.	9-200 60W7	1W7=DEF M016R700
IF MATCHED, SAVE LIST,	11W4	1W4=CPROB M016R710
FIND RIGHT SIDE OF DEF.	11W7	M016R720
9-300 SUBPROCESS, TRY REPLACEMENT BY MATCHING SEGMENT TO RIGHT SIDES.	M110	M016R730
MAKE FREE VARIABLES DISJOINT	P13	M016R740
FIND RIGHT SEGMENT OF DEF.	70J4	M016R750
IF NONE, QUIT + FOR GEN.	12W6	M016R760
IF FOUND,	M113	M016R770
MATCH SEGMENT TO RIGHT SIDE.	70J4	M016R780
IF NO MATCH, QUIT + FOR GEN.	60W8	1W8=SUBST M016R790
IF MATCHED, SAVE LIST,	51W7	M016R800
FIND LEFT SIDE OF DEF.	P14	M016R810
IF NONE, CLEAN UP, QUIT +.	9-300 60W7	1W7=DEF M016R820
IF FOUND, COPY IT,	11W4	1W4=CPROB M016R830
ERASE OLD SEGMENT, AND REPLACE OLD WITH COPY FROM DEF.	11W7	M016R840
	M110	M016R850
	P14	M016R860
	70J4	M016R870
	12W6	M016R880
	M113	M016R890
	70J4	M016R900
	60W8	1W8=SUBST M016R910
	51W7	M016R920
	P13	M016R930
	709-302	M016R935
	J74	M016R940
	12W6	M016R945
	J72	M016R950
	21W6	M016R955

SUBSTITUTE IN PROB. TEX PER 1W8.		11W8 11W4 M115 10J4 20W5 J4 9-302 11W8 ERASE 1W8, REVERSE H5 FOR GEN.	J72	J5	M016R960 M016R965 M016R970 M016R975 M016R980 M016R985 M016R990 M016R995 R
1					
M17 SUBLVEL REPLACEMENT METHOD FOR PROBLEM (0). ADD ALL NEW SUBPROBLEMS TO UNTRIED LIST. H5-MEANS NO SOL'N, NO OUTPUT. (M17 TRIES ALL LEVELS A ONCE)		J48 60W0 50L5 11K5 J10 70J38 60W2	IWC=PROB		M017R000 M017R010 M017R020 M017R030 M017R040 M017R050 M017R060 M017R070 M017R080
FIND MAP OF DEF. LEFT SIDES. IF NONE, QUIT--.		J81 70J38 60W1 51W2 J82 70J38 60W2 51W0 40H0 10Q17	IW1=LFTMAP	M017R090 M017R100 M017R110 M017R120	
FIND MAP OF DEF. RIGHT SIDES. IF NONE, QUIT--.		J14 Q17 70J38 20W3	IW2=RTMAP	M017R130 M017R140 M017R142 M017R145 M017R148 M017R150 M017R160	
CLEAR LEVEL FIND LOWEST LEVEL IN PROBLEM. IF NONE, QUIT--. IF YES,		9-3 9-20 70J38 20W3	IW3=CURLEV	M017R165 M017R170 M017R175 M017R180 M017R190	
BUMP 1W3, TEST IF GREATER THAN 1. IF NOT, QUIT--. IF YES,	9-10	11W0 P25 60W4	IW4=CPROB	M017R200 M017R210 M017R220 M017R230 M017R240	
COPY PROBLEM FOR REPLACEMENT.	9-11	10J3 60W5 509-100 P26 9-20 709-2 J120 10Q17 11W4 J11 11W3 11W4 01W5 70 11W4 J72	IW5=FLAG	M017R250 M017R260 M017R270 M017R280 M017R290 M017R300 M017R310 M017R320 M017R330 M017R340 M017R350 M017R360	
SET 'NEW SUBPROBLEM FLAG' TO NO.		9-11			
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.	9-2				

SET UP DERIVATION ASSOCIATIONS,	9-1	11W4	M017R370
AND		11W0	M017R380
FINISH BUILDING THE NEW SUBPROBLEM.		10/16	M017R390
IF NO GOOD, QUIT -.		10M16	M017R400
IF GOOD, TRY SUBSTITUTION.		M19	M017R410
IF PROOF, QUIT+. IF NOT, QUIT-		70J38	M017R420
9-20 SUBPROCESS--BUMP AND TEST	9-20	M12	M017R430
IF 1W3 GREATER THAN 1.		J38	M017R440
TEST IF 1W3 GREATER THAN N2.		10N2	M017R470
IF NO, QUIT SUBPROCESS--.		11W3	M017R480
IF YES,		J115	M017R490
SUBTRACT 1, QUIT SUBPROCESS+.		700	NO OUTPUT M017R500
9-100 SUBPROCESS FOR SUBSEGMENT	9-100	10N1	M017R510
REPLACEMENT IN LOCATION (0).		11W3	M017R520
CREATE LIST OF FEASIBLE DEFs.		11W3	J111 OUTPUT 1W3M017R530
SAVE LIST FOR LATER ARASURE.		60W6	1W6=SEGLOCM017R540
GENERATE DEFs FOR LEFT SIDE MATCH.		51W1	M017R550
IF MATCHED, GO CLEAN UP.		12W6	M017R560
IF FAILED, ERASE OLD LIST,		M62	M017R570
CREATE NEW LIST OF FEASIBLE DEFs.		40H0	M017R580
SAVE LIST FOR LATER ERASURE.		109-200	M017R590
GENERATE DEFs FOR RIGHT SIDE MATCH.		J100	M017R600
ERASE LIST AND QUIR + FOR GEN.		709-101	M017R610
9-200 SUBPROCESS, TRY REPLACEMENT	9-200	J71	M017R620
BY MATCHING SEGMENT TO		J4	M017R630
LEFT SIDES.		60W7	M017R640
MAKE FREE VARIABLES DISJOINT.		11W4	M017R650
FIND LEFT SEGMENT OF DEF.		11W7	M017R660
IF NONE, QUIT + FOR GEN.		M110	M017R670
IF FOUND,		P13	M017R680
MATCH SEGMENT TO LEFT SIDE.		70J4	M017R690
IF NO MATCH, QUIT + FOR GEN.		12W6	M017R700
IF MATCHED, SAVE LIST,		M113	M017R710
FIND RIGHT SIDE OF DEF.		70J4	M017R720
9-300 SUBPROCESS, TRY REPLACEMENT	9-300	60W8	M017R730
BY MATCHING SEGMENT TO		51W7	M017R740
RIGHT SIDES.		P14	M017R750
MAKE FREE VARIABLES DISJOINT		9-301	M017R760
FIND RIGHT SEGMENT OF DEF.		60W7	M017R770
IF NONE, QUIT + FOR GEN.		11W4	M017R780
IF FOUND,		11W7	M017R790
MATCH SEGMENT TO RIGHT SIDE.		M110	M017R800
		P14	M017R810
		11W4	M017R820
		11W7	M017R830
		M110	M017R840
		P14	M017R850
		70J4	M017R860
		12W6	M017R870
		M113	M017R880
			M017R890

IF NO MATCH, QUIT + FOR GEN.	70J4	M017R900	
IF MATCHED, SAVE LIST,	60W8	1W8=SUBST M017R910	
	51W7	M017R920	
FIND LEFT SIDE OF DEF.	P13	M017R930	
IF NONE, CLEAN UP, QUIT +.	9-301 709-302	M017R935	
IF FOUND, COPY IT,	J74	MC17R940	
	12W6	M017R945	
ERASE OLD SEGMENT, AND	J72	M017R950	
REPLACE OLD WITH COPY FROM DEF.	21W6	M017R955	
	11W8	M017R960	
SUBSTITUTE IN PROB TEX PER 1W8.	11W4	M017R965	
	M115	M017R970	
SET 'NEW SUBPROBLEM FLAG' TO ON.	10J4	M017R975	
SET H5 TO QUIT-.	20W5	M017R980	
	J4	M017R985	
ERASE 1W8, REVERSE H5 FOR GEN.	9-302 11W8	M017R990	
	J72 J5	M017R995	
	R		
M19 FINISH BUILDING NEW SUBPROBLEM	1	M19 J53	1W3=NEWTEX M019R000
(3) FROM (2) VIA THM(1) BY		11W3	1W2=PROBL M019R010
METHOD (0). MEASURE UTILITY.		11W2	1W1=THM M019R020
H5-, NO OUTPUT, TEX ERASED.		10Q10	1W0=METHOD M019R030
H5+ MEANS OUTPUT (0) IS OK.		J11	M019R040
		11W3	M019R050
FILL		11W0	M019R060
QUIT		10Q11	M019R070
DESCRIPTION.		J11	M019RC80
		11W3	M019R090
		11W1	M019R100
MARK LOCAL FOR FOUND LIST.		10Q12	M019R110
EVALUATE UTILITY.		J11	M019R120
		11W3	M019R130
IF N.G., REJECT IT.		J136	M019R135
IF O.K., ADD TO UNTRIED LIST.		M43	M019R140
QUIT +, OUTPUT NEW PROBLEM.		11W3	M019R150
	9-1	709-1	M019R160
		M51	M019R170
		11W3	M019R180
		J33 J4	M019R190
TEST IF PRINTING REJECTS.		11K31	M019R200
IF NO, SKIP IT.		10Y	M019R210
IF YES, PRINT IT.		J2	M019R220
ERASE N. G. SUBPROBLEM.	9-2	709-2	M019R230
QUIT -.		M81	M019R240
		11W3	M019R250
		J72	M019R260
		J33 J3	M019R270
		R	
M40 TEST IF TOTAL EXPRESSIONS	1	M40 J51	1W0=TEX1 M040R000
(0) AND (1) MATCH.		11W1	1W1=TEX2 M040R010
FIND MAIN SEGMENT.		J81	M040R020

FIND MAIN SEGMENT.		709-0 11W0 J81 709-1 M41	J31	M040R030 M040R040 M040R050 M040R060 M040R070 M040R080 M040R090 M040R100 M040R110 M040R120 R
TEST IF SEGMENTS MATCH. TEST IF OTHER MAIN SEGMENT EXISTS.	9-0	11W0 J81 J5		M040R030 M040R040 M040R050 M040R060 M040R070 M040R080 M040R090 M040R100 M040R110 M040R120
NO, QUIT W/H5+ FOR MATCH YES, QUIT W/H5- FOR NO MATCH.	9-1	709-1 30H0	J31	M040R110 M040R120
M41 TEST IF SEGMENTS (0) AND (1) MATCH. TEST IF 1ST IS VARIABLE. IF NOT, EXAMINE SUBSEGMENTS. IF YES,		M41 11W0 P8 709-1 11W0 11W1 J2		IWO=SEG1 IW1=SEG2 M041R000 M041R010 M041R020 M041R030 M041R040 M041R050 M041R060 M041R070 M041R080 M041R090 M041R100 M041R110 M041R120
TEST IF SAME VARIABLE. IF YES, QUIT, H5+. IF NO,		70 11W1 P9	J31	M041R070 M041R080 M041R090
TEST IF 2ND IS FREE VARIABLE. IF NO, QUIT, H5-. IF YES,		70J31 11W0 P9		M041R100 M041R110 M041R120
TEST IF 1ST IS FREE VARIABLE. EXAMINE SUBSEGMENTS.	9-1	11W1 P8	J31	M041R130 M041R140
TEST IF NOT SEGMENT. IF NOT, QUIT, H5- IF YES,		70 12W0 12W1 J2	9-2	M041R150 M041R160 M041R170 M041R180
TEST IF SAME CONNECTIVES. IF NOT, QUIT, H5- IF YES,		70J31 11W0 J81		M041R190 M041R200 M041R210
FIND LEFT SUBSEGMENT. IF NONE, CHECK OTHER SIDE.		709-3 11W1 J81 709-3		M041R220 M041R230 M041R240 M041R250
FIND OTHER LEFT SUBSEGMENT. IF NONE, QUIT, H5-.		11W0 709-7 J81 709-7		M041R260 M041R270 M041R280 M041R290
TEST IF LEFT SEGMENTS MATCH. IF NOT, QUIT, H5-		M41 70J31 11W0 J82		M041R300 M041R310 M041R320 M041R330
FIND RIGHT SUBSEGMENT. IF NONE, CHECK OTHER SIDE.		709-4 11W1 J82 709-4		M041R340 M041R350 M041R360 M041R370
FIND OTHER RIGHT SUBSEGMENTS. IF NONE, QUIT, H5-		709-7 M41 J31 9-2 9-3 9-4	AND QUIT.	M041R380 M041R390
TEST IF RIGHT SEGMENTS MATCH, QUIT, REVERSE H5.		J5 J31 11W1 J81 709-2 11W1		
NO FIRST SEGMENT ON IWO, FIND FIRST SEGMENT ON IW1. IF NONE, H5+. ELSE H5-.			9-5 9-6	
NO SECOND SEGMENT ON IWO,				

FIND SECOND SEGMENT ON 1W1.		J82	9-5	M041R400
REVERSE H5 AND	9-6	J5		M041R410
QUIT, DISCARD (0).	9-7	30H0	J31	M041R420
	1			R
M42 ADD PROBLEM (0) TO FOUND LIST.	M42	J42		M042R000
IF CANNOT, SET H5 -.		60WC		M042R010
		50L11		M042R030
		11WC		M042R040
GET NUMBER OF LEVELS		Q2		M042R050
IF NONE, QUIT -.		709-300		M042R055
GET SUBLIST		9-100		M042R060
GET NUMBER OF DISTINCT VARIABLES		11W0		M042R070
IF NONE, QUIT -.		Q3		M042R080
GET SUBLIST		709-300		M042R085
GET NUMBER OF VARIABLE PLACES		9-100		M042R090
IF NONE, QUIT -.		11W0		M042R100
GET SUBLIST		Q4		M042R110
GENERATE SUBLIST FOR MATCH		709-300		M042R115
INSERT AT END OF LIST		9-100		M042R120
9-200 SUBPROCESS.	9-1	40H0		M042R130
COMPARE EXPRESSIONS	9-200	109-200		M042R140
9-100 SUBPROCESS, GET SUBLIST.	9-100	J100		M042R150
LOCATE SUBLIST.		709-1		M042R160
		11W0		M042R170
CREATE NEW SUBLIST.		J32	J65	M042R180
SAVE SUBLIST FOR OUTPUT.		9-1	30H0	M042R190
INSERT NEW SUBLIST.		9-200	11W0	M042R200
COPY DATA TERM.			M40	M042R210
MARK LOCAL.			J5	M042R220
INSERT BEFORE NEW SUBLIST.		9-100	64W1	M042R230
GET SUBLIST AND QUIT.			P55	M042R240
		70	J80	M042R250
		40H0		M042R260
		J90		M042R270
		J136		1W2=SUBLSTM042R280
		60W2		M042R290
		J64		M042R300
		11W1		M042R310
		J120		M042R320
		J136		M042R330
		J64		M042R340
		11W2	0	M042R350
	1	9-300	30H0	R
M43 MEASURE UTILITY SUBPROBLEM (0).	M43	J43		M043R000
SET H5+ IF GOOD, H5- IF N.G.		60W0		1W0=PROB M043R010
FIND MEX.		J81		M043R020
GO THRU 'NOTS'		70J33		M043R030
		P4		M043R040
		70J33		M043R045
SAVE UNNOTTED MEX.		60W1		1W1=MEX M043R050
TEST IF VARIABLE		P8		M043R060

IF H5-, QUIT (VARIABLE ONLY)	J5	M043R070	
	70J33	M043R080	
TEST IF MAIN CONNECTIVE 'OR' IF NOT 'OR', LOOK ON FOUND LIST	12W1	2W1=MCONN M043R090	
	11K1	1K1='OR' M043R100	
LOCATE RIGHT SIDE	J2	M043R110	
	709-10	M043R120	
GET RIGHT SIDE	11W1	M043R130	
	J60	M043R140	
LOCATE LEFT SIDE	709-0	M043R150	
	12H0	M043R160	
GET LEFT SIDE	J6	M043R170	
TEST IF SIDES MATCH.	J60	M043R180	
	709-1	M043R190	
IF SAME QUIT W/H5-	52H0	M043R200	
	M114	M043R210	
ADD TO FOUND LIST IF CAN	J5	M043R220	
	70J33	M043R230	
9-10	11WC	M043R240	
	M42	J33	M043R260
9-1	30H0	M043R270	
9-0	30H0	J33	M043R280
	1	R	
M50 ADD TEX(0) TO TRUE EXPRESSIONS LIST AND TRUE EXPRESSIONS MAP. MAKE ALL VARIABLES FREE.	M50	40H0	M050R000
		J50	M050R010
		P27	M050R020
ADD TO LIST	10L1	M050R030	
	11W0	M050R040	
	J65	M050R050	
	10L4	M050R060	
ADD TO MAP	11W0	M050R070	
	M54	M050R080	
PRINT EXPRESSION AND QUIT.	11W0	M050R090	
	M70	J30	M050R100
		R	
M51 PRINT NEW SUBPROBLEM (0) AND ADD TO UNTRIED SUBPROBLEM LIST.	M51	J41	M051R000
		60WC	IWO=PROB M051R010
		10K10	M051R020
TALLY PREVIOUS SUBPROBLEM NUMBER.	J125	M051R030	
	J120	M051R040	
	J136	M051R050	
ASSIGN PROBLEM NO.	10Q8	M051R060	
	J11	M051R070	
FIND NO. OF LEVELS IF NONE, QUIT -.	10L10	M051R080	
	11W0	M051R110	
	Q2	M051R120	
LOCATE CORRESPONDING LIST.	709-3	M051R125	
	60W1	IWI=LEVELSM051R130	
	P55	M051R140	
GET LIST.	709-1	M051R150	
	52H0	M051R160	
	11W0	M051R170	

ADD NEW SUBPROBLEM.		J65	9-2	M051R190
	9-1	40HO		M051R200
		11W0		M051R210
CREATE LIST OF ONE SUBPROBLEM.		J91		M051R223
INSERT NEW LIST.		J136		M051R226
COPY LEVEL DATA TERM.		J64		M051R230
INSERT BEFORE NEW LIST.		11W1		M051R240
PRINT NEW SUBPROBLEM.		J120		M051R250
	9-2	J136		M051R260
		J64	9-2	M051R270
M54 ADD TOTAL EXPRESSION (0) TO MAP OF TRUE EXPRESSIONS (1).	9-2	11W0		M051R280
		M75	J31	M051R290
	9-3	30HO	J31	M051R300
				R
ADD MAIN SEGMENT (1) TO MAP (0).	1	M54	40WO	M054R000
9-100 SUBPROCESS, ADD SEGMENT (1) TO MAP (0).			60WO	1WO=THMNAMM054R010
TEST IF SIMPLE VARIABLE. IF NO, CONTINUE DOWN MAP. IF YES, ADD THMNAME.			J81	M054R020
TEST IF NAME LIST IN MAP HEAD. IF NO, GO MAKE ONE. IF YES,			709-0	(0)=MEX M054R030
INSERT NAME AND QUIT.			J6	M054R040
CREATE LIST OF ONE NAME.			9-100	M054R050
PLACE IN MAP HEAD AND QUIT.	9-0	30HO	J30	M054R060
INPUT MAP HOLDER.	9-100	04J43		1WO=THMNAMM054R070
INPUT SEGMENT CONNECTIVE.		20W1		1W1=MAP M054R080
FIND SUBMAPS LIST. IF FOUND, CONTINUE. IF NONE,		60W2		1W2=SEGMNTM054R090
CREATE 1ST LOCAL SUBMAP.		P8		M054R100
TEST IF 2ND SUBLIST NEEDED. IF NO, SKIP IT. IF YES,		709-102		M054R110
CREATE 2ND LOCAL SUBMAP.		11W1		M054R120
CREATE SUBMAP LIST.		J79		M054R130
	9-101	709-101		M054R140
		12W1		M054R150
		11W0		M054R160
		J64	J33	M054R170
	9-101	11W0		M054R180
		J91		M054R190
		J136		M054R200
		21W1	J33	M054R210
	9-102	10W1		M054R220
		12W2		M054R230
		J10		M054R240
		70	9-110	M054R250
		10W1		M054R260
		J90		M054R270
		J136		M054R280
		12W2		M054R290
		P6		M054R300
		709-111		M054R310
		40HO		M054R320
		J120		M054R330
		J92	9-112	M054R340
	9-111	J91		M054R350
	9-112	J136		M054R360

ASSIGN AS SUBMAP LIST OF CONNECTIVE	60W3 12W2 J11 11W3 9-110 20W3 11W2 J81 70J33 11W3 J81 709-114 9-100 12W2 P6 70J33 11W2 J82 70J33 11W3 J82 709-114 9-100 J33 9-114 30H0 J33	1W3=MAPLSTM054R370 M054R380 M054R390 M054R400 1W3=MAPLSTM054R410 M054R420 M054R430 M054R440 M054R450 M054R460 M054R470 M054R480 M054R490 M054R500 M054R510 M054R520 M054R530 M054R540 M054R550 M054R560 M054R570 M054R580 M054R600 R
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.	M60 10L10 9-1 J60 J60 70J8 12H0 J81 709-1 J6 52H0 J60 70J7 J68 J138 J4	M060R000 M060R010 M060R020 M060R030 M060R040 M060R050 M060R060 M060R070 M060R080 M060R090 M060R100 M060R110 M060R120 R
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.	M62 J45 20W0 60W1 52W1 J73 60W2 11W0 P8 70 J35 51W0 J60 709-1 60W3	1W0=SEG M062R000 1W1=MAP M062R010 M062R020 M062R030 M062R040 1W2=THMLSTM062R050 M062R060 M062R070 M062R080 M062R090 M062R100 M062R110 1W3=SEGLOCM062R120 M062R120
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.	50W1	M062R130
IF NONE, OUTPUT 1W2, QUIT.	12W0	M062R140
LOCATE 1ST SUBMAP.	J10	M062R150
IF NONE, OUTPUT 1W2, QUIT.	709-0	M062R160
IF THERE, SAVE LOCATION,	J60	M062R170
SET UP HO AND	709-1	M062R180
CREATE LIST FROM MAP (1) FOR (0).	60W4	1W4=MAPLOCM062R190
TEST IF CONNECTIVE WAS NON-UNARY.	52W4	M062R200
IF UNARY, FIX OUTPUT, QUIT.	12W3	M062R210
IF NON-UNARY, SAVE LIST AND	M62	M062R220
LOCATE NEXT SUBSEGMENT.	12W0	M062R230
IF NONE, FIX OUTPUT, QUIT.	P6	M062R240
LOCATE NEXT SUBMAP.	709-2	M062R250
IF NONE, FIX OUTPUT, QUIT.	60W5	1W5=ANDLSTM062R260
CREATE LIST FROM MAP (1) FOR (0),	9-4 51W3	M062R270
'AND' RESULT WITH LIST 1W5, LOOP.	J60	M062R280
FIX OUTPUT - - -	60W3	M062R290
'OR' W5 WITH 1W2,	709-3	M062R300
LEAVE RESULT AS 1W2.	51W4	M062R310
OUTPUT 1W2, CLEAR CONTEXT.	J60	M062R320
SUBPROCESS - 'AND' (0) WITH 1W5.	60W4	M062R330
GENERATE 1W5 FOR PROCESS MARKING.	709-3	M062R340
GENERATE '(0)' TO UNMARK MARKED.	52W4	M062R350
ERASE '(0)'	12W3	M062R360
LOCATE NEXT OF 1W5.	M62	M062R370
IF NONE, QUIT SUBPROCESS.	9-100 9-4	M062R380
TEST IF EXPRESSION MARKED.	9-3 51W5	M062R390
IF NO, LOOP TO NEXT.	11W2	M062R400
IF YES, UNMARK IT,	J6	M062R410
SAVE LOCATION AND DELETE THIS	J76	M062R420
EXPRESSION DUE NOT ON BOTH LISTS.	9-1 51W2 J35	M062R430
IF MORE, LOOP WITH NEXT.	9-0 11W2 J35	M062R440
SUBPROCESS - MARK PROCESSED.	9-100 11W5	M062R450
SUBPROCESS - UNMARK IF MARKED.	109-200	M062R460
	J100	M062R470
	40HO	M062R480
	109-300	M062R490
	J100	M062R500
	J71	M062R510
	11W5	M062R520
	9-101 J60	M062R530
	700	M062R540
	9-102 12HO	M062R550
	J133	M062R560
	709-101	M062R570
	32HO	M062R580
	40HO	M062R590
	J68	M062R600
	700 9-102	M062R610
	9-200 J137 J8	M062R620
	9-300 40HO	M062R630

M63 CREATE A LIST OF TRUE
EXPRESSIONS FROM MAP (0) FOR
FEASIBLE MATCH WITH TEX (1).
OUTPUT MAY BE AN EMPTY LIST.

M70 PRINT EXPRESSION (0), WITH OR
WITHOUT A SUFFIX.
ENTER NAME.

TAB TO COLUMN (0).

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 (0).

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

		J133		M062R640
		709-201		M062R650
		31H0 J8		M062R660
	9-201	30H0 J4		M062R670
			R	
1	M63	J6		M063R000
		J81		M063R010
		70 M62		M063R020
		30H0 J90		M063R030
			R	
1	M70	J154		M070R000
		40H0		M070R010
		M79		M070R020
		10N8		M070R030
		J160		M070R040
		40H0		M070R050
		J81		M070R060
		709-1		M070R070
		M73		M070R080
9-1		Q18		M070R090
		70J155		M070R100
		10N1		M070R110
		J161		M070R120
		J157 J155		M070R130
			R	
1	M71	J50	IWO=TEX	M071R000
		J154		M071R010
		J155		M071R020
		J155		M071R030
		10T2		M071R040
		M76		M071R050
		J155		M071R060
		J154		M071R070
		J155		M071R080
		11WC		M071R090
		Q13		M071R100
		70 9-1		M071R110
		10/14		M071R120
9-1		10T1		M071R130
		M80		M071R140
		11W0		M071R150
		10T3		M071R160
	9-6	M80		M071R170
		11W0		M071R180
		Q12		M071R190
		709-2		M071R200
		10T1		M071R210
		M80		M071R220
		11W0		M071R230
		Q11		M071R240
		709-3		M071R250

FIND EXTERNAL NAME OF METHOD.		Q16		M071R260
		70	9-4	M071R270
IF NONE, USE BLANKS.	9-3	10T5		M071R280
	9-4	11WC		M071R290
FIND PROBLEM USED IN DERIVATION.		Q10		M071R300
IF NONE, USE DUMMY CHARACTER.	9-5	70	9-5	M071R310
		10/14		M071R320
LOOP TO PRINT NEXT EVEN LINE.	9-2	60WC		M071R330
		J6	9-6	M071R340
TAB TO COLUMN K41.		J154		M071R350
		10K41		M071R360
ENTER 'Q.E.D.' AND		J160		M071R370
PRINT.		10T4		M071R380
PRINT LIMITS, CLEAR CONTEXT/QUIT.		M76		M071R390
		J155		M071R400
		M77	J30	M071R410
				R
PRINT -NO PROOF FOUND-	M72	J154		M072R000
		10T6		M072R010
ENTER MESSAGE		M76		M072R020
PRINT MESSAGE, LIMITS		J155	M77	M072R030
				R
ENTER SEGMENT (0)	M73	4CH0		M073R000
TEST IF VARIABLE		P8		M073R010
IF YES, ENTER VARIABLE.		70	M79	M073R020
		12H0		M073R030
		11K2		M073R040
TEST IF CONNECTIVE NOT		J2		M073R050
		709-1		M073R060
		12H0		M073R070
ENTER NOT.		M79		M073R080
		J81		M073R090
		70J7	9-200	M073R095
		700	9-200	M073R100
	9-1	J41		M073R110
LOCAT FIRST SEGMENT.		60WC	1W0=SEG	M073R120
		J60		M073R130
IF NONE, QUIT.		20W1	1W1=LOC'N	M073R140
		70J31		M073R150
ENTER SEGMENT.		12W1	2W1=SUBSEG	M073R160
		9-200		M073R170
LOCATE NEXT SUBSEGMENT.	9-2	11W1		M073R180
		J60		M073R190
IF NONE, QUIT.		20W1		M073R195
		70J31		M073R200
ENTER CONNECTIVE.		12W0	2W0=CONN	M073R205
		M79		M073R210
ENTER SEGMENT.		12W1		M073R215
		9-200		M073R220
		11W1	9-2	M073R225
	9-200	40HC		M073R230

TEST IF VARIABLE	P4 709-201 P8 70 M73 10K51 J157 M73 10K52 J157 9-201 50/14 M79 1 M74 40HO J81 70J8 M73 Q18 700 10N1 J161 J157 1 M75 J154 10K47 J160 40HO PSV PROB M79 10N3 J161 40HO PSV PROB M74 11W25 10K48 J116 70 9-2 10K48 J160 9-2 10N2 J161 40HO Q12 709-1 M79 10K54 J157 10N1 J161 Q11 70J155 Q16 70J155 M76 J155 1 M76 10J157	M073R235 M073R238 M073R240 M073R250 M073R260 M073R270 M073R280 M073R290 M073R300 R M074R000 M074R010 M074R020 M074R030 M074R040 M074R050 M074R060 M074R070 R M075R000 M075R010 M075R020 M075R030 M075R040 M075R050 M075R060 M075R070 M075R080 M075R090 M075R100 M075R110 M075R120 M075R130 M075R140 M075R160 M075R170 M075R180 M075R190 M075R200 M075R210 M075R220 M075R230 M075R240 M075R250 M075R260 M075R270 M075R280 M075R290 M075R300 R M076R000
------------------	---	---

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,
BUMP 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.

		J100	0	M076RC10
			R	
M77 PRINT LIMITS OF PROOF.	1	M77	J154	M077R000
			J155	M077R010
DOUBLE SPACE.			J155	M077R020
			10K12	M077R030
			10H3	M077R040
SET K12 TO ACTUAL EFFORT.			10K12	M077R050
			J111	M077R060
INPUT 'EFFORT'			10K22	M077R070
PRINT LINE.			10T7	M077R080
			9-100	M077R090
INPUT 'SUBPROBLEMS'			10K10	M077R100
PRINT LINE.			10K20	M077R110
			10T8	M077R120
			9-100	M077R130
INPUT 'SUBSTITUTIONS'			10K11	M077R140
9-100 SUBPROCESS, PRINT LINE.	9-100	J154	9-100	M077R160
ENTER MESSAGE.			M76	M077R170
TAB TO COLUMN K44.			10K44	M077R180
			J160	M077R190
ENTER 'LIMIT'			10T21	M077R200
			M76	M077R210
BUMP COLUMN.			10N1	M077R220
ENTER LIMIT.			J161	M077R230
TAB TO COLUMN K45.			J157	M077R240
			10K45	M077R250
ENTER 'ACTUAL'			J160	M077R260
			10T20	M077R270
BUMP COLUMN.			M76	M077R280
ENTER ACTUAL AND PRINT LINE.			10N1	M077R290
			J161	M077R300
			J157	M077R310
			J155	M077R320
			R	
M78 PRINT 'TO PROVE' PROBLEM (0).	1	M78	J154	(0)=PROB
INPUT TEXT.			10T22	M078R000
ENTER MESSAGE.			M76	M078R010
SET UP TO PRINT ON NEW PAGE.			40W22	M078R020
RESTORE SPACING AND PRINT (0).			10N3	M078R030
			20W22	M078R040
			J155	M078R050
			30W22	M078R060
			M70	M078R070
			R	
M79 ENTER NAME OF (0).	1	M79	40H0	PSV (0)
FIND EXTERNAL NAME.			Q7	M079R000
IF THERE,			709-1	M079R010
ENTER IT, DISCARD (0).			J157	M079R020
IF NOT THERE,	9-1	40H0	J8	M079R030
FIND SUBPROBLEM NUMBER.			Q8	M079R040
				M079R050

IF NOT THERE, ENTER INTERNAL.	70J156	M079R060
IF THERE, ENTER NO.	J157	M079R070
AND ENTER PERIOD.	50K53 J157	M079R080 R
M80 PRINT PROOF LINE.	M80 J154	M080R000
INPUT (0) IS METHOD OR 'GIVEN'	10K41	M080R010
INPUT (1) IS TEX	J160	M080R020
ENTER METHOD	M76	M080R030
	10K42	M080R040
	J160	M080R050
	40HO	M080R060
	M79	M080R070
ENTER NAME	10K43	M080R080
	J160	M080R090
ENTER EXPRESSION AND PRINT.	M74 J155	M080R100 R
M81 PRINT REJECTED PROBLEM (0).	M81 J154	M081R000
TAB TO COLUMN K47.	10K47	M081R010
	J160	M081R020
	40HO	M081R030
ENTER NAME.	M79	M081R040
BUMP COLUMN.	10N3	M081R050
	J161	M081R060
	40HO	M081R070
ENTER TEX.	M74	M081R080
	11W25	M081R090
	10K48	M081R100
TEST IF TEX TOO LONG.	J116	M081R110
IF YES, SKIP RESET.	70 9-2	M081R120
IF NO,	10K48	M081R130
TAB TO COLUMN K48	J160	M081R140
	9-2 10N2	M081R150
BUMP COLUMN.	J161	M081R160
	40HO	M081R170
FIND THEOREM.	Q12	M081R180
IF NONE, SKIP IT.	709-1	M081R190
IF THERE, ENTER NAME.	M79	M081R200
ENTER COMMA, AND	10K54	M081R210
	J157	M081R220
BUMP COLUMN.	10N1	M081R230
FIND METHOD.	J161	M081R240
IF NONE, SKIP IT.	9-1 Q11	M081R250
IF THERE, FIND EXTERNAL NAME.	709-3	M081R260
IF NONE, SKIP IT.	Q16	M081R270
IF THERE, ENTER TEXT,	709-3	M081R280
	M76	M081R290
ENTER PERIOD, AND	10K53	M081R300
	J157	M081R310
BUMP COLUMN.	10N2	M081R320
	J161	M081R330
	9-3 10T19	M081R340

ENTER MESSAGE AND PRINT.		M76	J155	M081R350 R
M82 PRINT 'REMEMBER PROVED THEOREM.'	1	J154		M082R000
		J155		M082R010
		J155		M082R020
		10T23		M082R030
ENTER MESSAGE.		M76		M082R040
		J155		M082R050
		J154	J155	M082R060 R
M88 PRINT BAD LIST FORM EXPRESSION.	1	J154		M088R000
ENTER 'BAD EXPRESSION'		10T24		M088R010
BUMP COLUMN.		M76		M088R020
		10N3		M088R030
		J161		M088R040
		10M79		M088R050
GEN SYMBOLS FOR ENTRY, PRINT.	1	J100	J155	M088R060 R
M89 READ NEXT LOGIC EXPRESSION FROM NORMAL INPUT UNIT.	M89	40W0		M089R000
H5- MEANS NONE THERE.		40W25		M089R010
CLEAR AND FILL BUFFER.	9-10	40W30		M089R020
IF EOF, QUIT, H5-.		J154		M089R050
		J180		M089R060
LOCATE 1ST OF NAME.		709-0		M089R070
IF BLANK CARD, QUIT, H5-.		10N1		M089R080
		J120		M089R090
DETERMINE EXTENT.		J184		M089R110
IF REST OF CARD, RESET, GET NEXT		709-1		M089R120
INPUT NAME.		20W25	1W25=1ST	M089R125
		J90		M089R130
		J124		M089R140
		J183		M089R150
TEST IF REGIONAL.		709-2		M089R160
IF NOT, RESET, GET NEXT.		20W30	1W30=EXTNTM089R170	M089R170
		J181		M089R180
		40H0		M089R190
		J130		M089R200
TEST IF NAME IS A CHARACTER SYMBOL.		709-4		M089R210
IF YES, RESET, GET NEXT.		40HC		M089R213
IF OK, GET EXTERNAL NAME.		P18		M089R215
RESET COLUMN TO 1ST OF NAME.		709-4		M089R217
CREATE PROTOTYPE DATA TERM.		20W0	1W0=EXPR.	M089R220
MARK LOCAL.		11W30		M089R230
SET D.T. TO HOLD NAME.		11W25		M089R240
		11W25		M089R250
		J111		M089R260
		50K51		M089R265
		J120		M089R270
		J136		M089R280
		J182		M089R290
		11W0		M089R300

ASSIGN D.T. AS EXTERNAL NAME.		J6	M089R310
LOCATE 1ST OF EXPRESSION. IF NONE, RESET, GET NEXT.		10Q7	M089R320
FIND CHARACTER SYMBOL AT 1W25. IF NONE, EXPRESSION FINI. IF FOUND, GET ALTERNATE,	9-3	J11 11W25 J184 709-5 J186 709-6 P19 11W0 J6 J65 J125 9-3 J184 709-7 51W30 J124 J183 709-7 50K51 J120 J182 11W0 J6 10Q18 J11 11W0 9-8 51W0 9-8 J4 11W25 J9 11W30 J9 9-1 9-2 9-3 9-4 9-5 9-6 9-7 9-8 9-9 9-10 10K20 10K10 J116 70J5 10K21 10K11 J116 70J5 10K12	J184 M089R330 MC89R340 M089R350 M089R360 M089R370 M089R380 M089R390 MC89R400 M089R410 M089R420 M089R430 M089R440 M089R450 M089R460 M089R470 M089R480 M089R490 M089R495 M089R500 M089R510 M089R520 M089R530 M089R540 M089R550 M089R560 M089R570 M089R580 M089R590 M089R600 M089R610 M089R620 M089R630 M089R640 M089R650 M089R660 M089R670 M089R700 M089R710 M089R720 M089R730 M089R740 M089R750 M089R760 R M090R000 M090R010 M090R020 M090R030 MC90R040 M090R050 M090R060 M090R070 M090R080
ADD AT END OF LIST EXPRESSION, TALLY 1W25 AND LOOP FOR NEXT. LOCATE FIRST OF SUFFIX. IF NONE, SKIP IT.	9-6		
DETERMINE EXTENT. IF NONE, SKIP IT.			
CREATE PROTOTYPE. SET D.T. TO SUFFIX.			
ASSIGN AS SUFFIX OF 1W0.			
ADJUST FOR EXTRA IN H0. SET H5+	9-7 9-8		
DISCARD COLUMN D.T.			
DISCARD EXTENT D.T.	9-1 9-2 9-3 9-4 9-5 9-6 9-7 9-8 9-9		
RESET AFTER A BAD EXPRESSION AND GET NEXT CARD.			
M90 TEST IF PROBLEM LIMITS REACHED.	1 M90		
TEST NO. OF SUBPROBLEMS SET UP			
TEST NO. OF SUBSTITUTIONS			

COMPUTE EFFORT	10H3 J90 J111 40H0 10K22 J6 J116 J9	J5	M090R090 M090R100 M090R110 M090R120 M090R130 M090R140 M090R150 M090R160 R
TEST EFFORT.	1		
M110 MAKE FREE VARIABLES OF TOTAL EXPRESSIONS (0) AND (1)	M110	J47 J21	M110R000 M110R010
DISJOINT. (SUBSTITUTES IN (1).)		11W1	M110R020
CREATE FREE VAR. LIST FOR (1)		M116	M110R030
		70J37	M110R040
		20W3	M110R050
		11W0	M110R060
CREATE FREE VAR. LIST FOR (0)		M116	M110R070
		709-1	M110R080
		20W2	M110R090
CREATE EMPTY SUBSTITUTION LIST.		J90	M110R100
		20W4	M110R110
CREATE EMPTY LIST OF MARKED PROCESSED		J90 20W7	M110R114 M110R116
SET LOCATION ON SYSTEM FREE VARIABLE LIST		10L2 20W5	M110R120 M110R130
MARK ALL OF (0)-S FREE VARS.		11W2 109-100 J100	M110R140 M110R150 M110R160
		11W3 109-200	M110R170 M110R180
ADD DUPLICATES TO SUBSTITUTION LIST		J100 11W4	M110R190 M110R200
TEST IF ANY DUPLICATES.		J78 70	M110R210 M110R220
UNMARK ALL MARKED VARIABLES	9-6	9-400	M110R225
LOCATE NEXT DUPLICATE	9-5	9-2	M110R230
		11W4	M110R240
		J60	M110R250
		20W6	M110R260
		709-3	M110R270
LOCATE NEXT SYSTEM FREE VAR.	9-4	11W5	M110R280
HALT DUE TO NOT ENOUGH FREE VAR.		J60	M110R290
TEST IF USED IN EITHER.		20W5 70J7	M110R300
		12W5	M110R310
		J133	M110R320
INSERT AS SUBSTITUTOR.	70	9-4	M110R330
		11W6	M110R340
		12W5	M110R350
		J64	M110R360
		11W6	M110R370
		J60	M110R380

NUMARK ALL MARKED VARIABLES.	9-3	9-400	M110R390		
		11W4	M110R420		
		11W1	M110R430		
GET MAIN SEGMENT OF (1).		J81	M110R440		
		70J7	M110R450		
SUBSTITUTE.		M115	M110R460		
	9-2	11W4	M110R470		
		J71	M110R480		
		11W2	M110R490		
		J71	M110R500		
	9-1	11W3	M110R510		
		J71	M110R520		
MARK PROCESSED	9-100	J137	M110R525		
		11W7	M110R530		
		J6	M110R535		
ADD TO LIST 1W7 FOR UNMARKING		J64	M110R540		
ADD THOSE MARKED TO SUBST. LIST.	9-200	40HO	M110R550		
MARK THOSE NOT MARKED.		J133	M110R560		
		709-100	M110R570		
		11W4	M110R580		
		J6	M110R590		
UNMARK PROCESSED.	9-300	31HO	M110R600		
		30HO	M110R610		
UNMARK PROCESSED ALL VARIABLES	9-400	11W7	M110R620		
ON LIST 1W7		109-300	M110R630		
		J100	M110R640		
		R			
M111 MATCH SEGMENTS (0) AND (1),	1	M111	J90	M111R000	
H5+ MEANS OUTPUT (0) IS LIST			209-10	M111R010	
OF PAIRS--1ST IS FREE VAR.,			9-100	M111R020	
2ND IS SUBSTITUTOR.			119-10	M111R030	
H5- MEANS NO MATCH, NO OUTPUT.			70J71	0	M111R040
9-100 MATCH SUBPROCESS	9-100	04J51		M111R050	
(EXPECTS FREE VARIABLES DISJCIN	9-104	11W0		M111R060	
IS (C) A VARIABLE.		P8		M111R070	
		709-101		M111R080	
IS (0) A FREE VARIABLE.		11W0		M111R090	
		P9		M111R100	
		709-102		M111R110	
IS THERE ALREADY A SUBSTITUTOR		109-10		M111R120	
FOR (0).		11W0		M111R130	
SET SUBSTITUTOR AS (0) AND MATCH.		J10		M111R140	
		709-103		M111R150	
TEST IF (1) IS A VARIABLE		20W0	9-104	M111R160	
	9-103	109-10		M111R170	
		11W1		M111R180	
		40HO		M111R185	
		P8		M111R190	
		709-105		M111R195	
		40HO		M111R200	
		11W0		M111R205	

(1) IS VARIABLE, TEST (1)=(0)	J2	M111R210
	709-114	M111R215
NO SUBSTITUTOR, QUIT W/H5+	9-116 3CHO	M111R220
	30HO J31	M111R225
IF NOT, MAKE EXPRESSION INTERNAL	9-105 J138	M111R230
ASSIGN (1) AS SUBSTITUTOR FOR (0)	9-114 11WO	M111R235
	9-109 J11	M111R240
	J31 J4	M111R250
((0) IS VARIABLE, NOT FREE.)	9-102 11W1	M111R260
IS (1) A VARIABLE.	P8	M111R270
	70J31	M111R280
IS (1) A FREE VARIABLE.	11W1	M111R290
	P9	M111R300
	709-106	M111R310
IS THERE ALREADY A SUBSTITUTOR	9-111 109-10	M111R320
FOR (1).	11W1	M111R330
SET SUBSTITUTOR AS (1) AND MATCH.	J10	M111R340
	709-1C7	M111R350
	20W1 9-104	M111R360
TEST IF (0) IS A VARIABLE	9-107 109-10	M111R370
	11WO	M111R380
	4CHO	M111R390
	P8	M111R400
(0) IS VARIABLE, TEST (0)=(1)	709-108	M111R405
IF NOT, MAKE EXPRESSION INTERNAL	40HC	M111R410
ASSIGN (0) AS SUBSTITUTOR FOR (1)	11W1	M111R415
(BOTH ARE VARIABLES, NOT FREE)	J2	M111R420
ARE VARIABLES IDENTICAL.	709-115 9-116	M111R425
((0) IS EXPRESSION.)	9-108 J138	M111R430
IS (1) A VARIABLE	9-115 11W1 9-109	M111R435
	9-106 11WO	M111R440
	11W1	M111R450
	J2 J31	M111R460
IS (1) A FREE VARIABLE.	9-101 11W1	M111R470
	P8	M111R480
(BOTH ARE EXPRESSIONS.)	709-110	M111R490
ARE CONNECTIVES IDENTICAL	11W1	M111R500
	P9	M111R510
	70J31 9-111	M111R520
LOCATE NEXT SUBSEGMENT ON (0)	9-110 12WO	M111R530
	12W1	M111R540
	J2	M111R550
	70J31	M111R560
LOCATE NEXT SUBSEGMENT ON (1)	9-113 11WO	M111R570
	J60	M111R580
	20WO	M111R590
	709-112	M111R600
	11W1	M111R610
	J60	M111R620
	20W1	M111R630
	70J31	M111R640

MATCH SUBSEGMENTS	12WC	M111R650		
	12WI	M111R660		
	9-100	M111R670		
	70J31	9-113	M111R680	
LOCATE NEXT SUBSEGMENT ON (1)	9-112	11W1	M111R690	
		J60	M111R700	
		20W1	M111R710	
		J5	1 J31	M111R720
			R	
M112 EXPAND SUBSTITUTION LIST (0).	M112	40HO	M112R000	
REPLACE EXPRESSIONS WITH		J51	M112R010	
COMPLETELY SUBSTITUTED LOCALLY	9-1	11WC	M112R020	
NAMED COPIES.		J60	M112R030	
LOCATE NEXT SUBSTATION		J60	M112R040	
		20WO	M112R050	
		70J31	M112R060	
		12WO	M112R070	
DELINEATE SEGENT AT THIS STATION		9-100	M112R080	
		21WO	9-1	M112R090
SEGMENT DELINEATION SUBPROCESS.	9-100	4J50	M112R100	
		11WC	M112R110	
IS INPUT A VARIABLE.		P8	M112R120	
		11WC	M112R130	
		709-101	M112R140	
		P9	M112R150	
IS INPUT A FREE VARIABLE.		709-102	M112R160	
		11WO	M112R170	
DELINEATE FREE VARIABLE.		9-200	J30	M112R180
CREATE SUBSTITUTED LOCAL COPY	9-102	11WC	J30	M112R190
	9-101	J74		M112R200
		J136		M112R210
		60WO		M112R220
		109-300		M112R230
GENERATE FREE VARIABLE LOCATIONS.		P28	9-102	M112R240
	9-300	J50		M112R250
		12WO		M112R260
		9-200		M112R270
		21WC		M112R280
		J30	J4	M112R290
DELINEATE FREE VARIABLE	9-200	4J50		M112R300
		10W1		M112R310
		11WC		M112R320
FIND CORRESPONDING SUBSTITUTOR		J10		M112R330
		709-201		M112R340
DELINEATE SUBSTITUTOR		9-100	J30	M112R350
OUTPUT FREE VARIABLE	9-201	11WC	J30	M112R360
			R	
MATCH SEGMENTS (0) AND (1) FOR	M113	M111	M113R000	
SUBSTITUTION. SETS H5, IF + OUTPUT		700	M113R010	
(0) IS EXPANDED SUBSTITUTION LIST.		40HO	M113R020	
		M112	J4	M113R030

TEST IF SEGMENT (0) MATCHES SEGMENT (1).	1	M114	M111	R
		700	J71	M114R000
	1			M114R010
				R
M115 SUBSTITUTE IN SEGMENT (0) FROM SUBSTITUTION LIST (1). SUBSTITUTES ONLY FOR VARIABLES.		M115	J42	M115R000
SAMPLE SUBST. LIST. IF EMPTY, QUIT.			20W0	M115R003
			60W1	M115R007
			J81	M115R010
			70J32	M115R013
			40HC	M115R017
TEST IF FREE VARIABLE. IF YES, GENERATE LOCATIONS OF FREE VARIABLES.			P9	M115R020
			709-1	M115R023
			51WC	M115R027
			109-100	M115R030
			P28 J32	M115R033
TEST IF BOUND VARIABLE. IF YES, GENERATE LOCATIONS OF BOUND VARIABLES.	9-1		P8	M115R037
			70J32	M115R040
			11WC	M115R043
			109-100	M115R047
			P29 J32	M115R050
9-100 SUBPROCESS, SUBST. IN LOC(C). •	9-100	60W2		M115R055
		50W1		M115R060
		12W2		M115R070
FIND SUBSTITUTOR, IF ANY		J10		M115R080
		70J4		M115R090
		40HC		M115R100
TEST IF A VARIABLE		P8		M115R110
		70 9-101		M115R120
IF AN EXPRESSION, COPY IT.		J74		M115R130
		J136		M115R140
STORE IN LOCATION	9-101	21W2	J4	M115R150
	1			R
CREATE LIST OF FREE VARIABLES IN TEX (0). SETS H5, NO OUTPUT IF -• GET MAIN SEGMENT.		M116	J90	M116R000
		J50		M116R010
		J81		M116R020
		709-1		M116R030
		109-100		M116R040
GENERATE LOCATIONS OF FREE VARS.		P28		M116R060
	9-1	11WC		M116R070
		40HC		M116R080
		30W0		M116R090
TEST IF ANY FREE VARS.		J78		M116R100
		70J71 0		M116R110
	9-100	52HC		M116R120
		11WC		M116R130
		J6		M116R140
ADD TO OUTPUT IF NOT ALREADY ON.		J66	J4	M116R150
	1			R
M117 FIND LIST OF BOUND VARIABLES IN TEX (0). H5- MEANS NO OUTPUT.		M117	P31	M117R000
		40HC		M117R010
		J78		M117R020

		70J9	0	M117R030
				R
P2 TEST IF (0) IS A BOUND VARIABLE.	P2	Q9 700	J8	P002R000 P002R010
				R
CLEAR DESCRIPTIONS OF TOTAL EXPRESSION (0)	P3	40HQ 10Q2 J14 40HQ 10Q3 J14 10Q4	J14	P003R000 P003R010 P003R020 P003R030 P003R040 P003RC50 P003R060
				R
P4 GO THRU NOTS OF SEGMENT (0), LEAVE 1ST UNNOTTED SEGMENT. H5- MEANS NO OUTPUT. QUIT, H5+ MEANS NORMAL EXIT. FIND SUBSEGMENT OF NOT. IF NONE QUIT -, ELSE LOOP.	P4	12HQ 11K2 J2 70J4 J81 700	P4	P004R000 P004R010 P004RC20 P004R030 P004R040 P004R050
				R
P5 TEST IF MAIN CONNECTIVE OF TOTAL EXPRESSION (0) IS IMPLIES.	P5	P16 700 11K6	J2	P005R000 P005R010 P005R020
				R
P6 TEST IF (0) IS NOT UNARY.	P6	Q14 700	J1	P006R000 P006R010
				R
P7-TEST IF (0) IS CONNECTIVE	P7	Q14 700	J8	P007R000 P007R010
				R
TEST IF (0) IS VARIABLE	P8	Q5 700	J8	P008R000 P008R010
				R
TEST IF (1) IS FREE VARIABLE	P9	Q6 700	J8	P009R000 P009R010
				R
P12 FIND MEX OF TEX (0).	P12	J81	0	P012R000
				R
P13 FIND LEFT SIDE OF TEX(0).	P13	J81 700	J81	P013R000 P013R030
				R
P14 FIND RIGHT SIDE OF TEX(0).	P14	J81 700	J82	P014R000 P014RC30
				R
P15 TEST IF (0) IS IN INTERNAL (TREE) FORM.	P15	Q15 700	J8	P015R000 P015R010
				R
P16 FIND MAIN CONNECTIVE OF TEX (0). GO THRU NOTS.	P16	J81 700 P4 700	Q1	P016R000 P016RC10 P016R020 P016R030

P17 CREATE COPY OF SEGMENT (0) IF NOT A SIMPLE VARIABLE.	1	P17	40H0 P8 70 J74	0 J136	R P017R000 P017R010 P017R020 P017R030
COPY AND MARK LOCAL.					R
P18 TEST IF (0) IS A CHARACTER SYMBOL.	1	P18	P19 Q7 J5 70J8	0	P018R000 P018R010 P018R020 P018R030
GET APPROPRIATE INTERNAL CHARACTER SYMBOL (0) FOR EXTERNAL CHARACTER SYMBOL (0). IF REPLACED, QUIT +.	1	P19	40HC Q19 70C J6	J8	R P019R000 P019R010 P019R020 P019R030
P20 MAKE FAKE TEX WITH LEFT SIDE OF TEX (0).	1	P20	P13 70C	P24	P020R000 P020R010
P21 MAKE FAKE TEX WITH RIGHT SIDE OF TEX (0).	1	P21	P14 700	P24	R P021R000 P021R010
P22 CREATE NEW SUBPROBLEM WITH SEGMENT (0) ON THE LEFT, SEGMENT (1) ON THE RIGHT, AND IMPLIES AS CONNECTIVE. CREATE MEX.	1	P22	40WC P17 J6 P17 J92 J136 60WC 11K6 21W0 P24	J30	R P022R000 P022R010 P022R020 P022R030 P022R040 P022R050 P022R060 P022R070 P022R080 P022R090
INSERT CONNECTIVE.					R
CREATE TEX, CLEAN UP, QUIT.					P023R000 P023R010
ERASE MADE EXPRESSION (0)	1	P23	40H0 J60	J72	P023R020
DELETE MAIN SEGMENT					R
P24 MAKE TEX FROM MEX (0).	1	P24	J91 40H0 10Q15 10Q15	J11	P024R000 P024R010 P024R020 P024R030
DESCRIBE AS IN TREE FORM.					R
P25 COPY TEX (0) FOR SUBSTITUTION.	1	P25	J74 40H0 10Q7 J14 40H0	P3	P025R000 P025R010 P025R020 P025R030 P025R040
CLEAR EXTERNAL NAME AND CLEAR DESCRIPTIONS.					R
P26 GENERATE SEGMENT LOCATIONS AT LEVEL (2) OF PROBLEM (1) FOR PROCESS (0).	1	P26	10W1 J17 J81		P026R000 P026R010 P026R020

TEST IF MEX IS A VARIABLE.	709-1	PC26R03C
IF YES, QUIT.	40HO	PO26R060
IF NO, REVERSE,	P8	PO26R070
CREATE COUNTER WITH VALUE 1, AND SAVE BOTH LEVEL AND COUNTER.	70 9-2	PO26R080
GENERATE SUBSEGMENT LOCATIONS FOR 9-100.	J6	PO26R090
ERASE COUNTER AND QUIT.	10N1	PO26R100
POP HO,	J120	PO26R110
POP HO, AND QUIT.	J21	1W1=LEVEL PO26R120
SUBPROCESS 9-100.	109-100	1W0=COUNT PO26R130
9-200		PO26R140
11WO		PO26R150
J9 J19		PO26R160
9-2 30HO		PO26R170
9-1 30HO	J19	PO26R180
9-100 11WO		PO26R190
11W1		PO26R200
J114		PO26R210
70 J18		PO26R220
52HO		PO26R225
109-100		PO26R230
9-200 0		PO26R240
9-200 14WO		PO26R250
J17		PO26R260
11WO		PO26R270
J120		PO26R280
J125		PO26R290
20WO		PO26R300
40WC		PO26R310
9-201 J60		PO26R320
709-203		PO26R330
40HO		PO26R340
12HO		PO26R350
P8		PO26R360
709-202		PO26R370
30HO 9-201		PO26R380
9-202 J18		PO26R390
70 9-201		PO26R400
9-203 51WO		PO26R410
30WO		PO26R420
J9 J19		PO26R430
R		
P27 REPLACE BOUND BY FREE IN (0).	1 P27 J43	PO27R000
CREATE LIST OF BOUND OF 1WO.	60WC	1W0=TEX PO27R010
LOCATE FIRST BOUND.	P31	PO27R020
IF NO BOUND, QUIT.	60W1	1W1=BNDLST PO27R030
CREATE LIST OF FREE OF 1WO.	J60	PO27R040
	20W2	1W2=BNDLOCP027R050
	709-1	PO27R060
	11WC	PO27R070
	P30	PO27R080
	60W3	1W3=FREELSP027R090

GEN FREE VAR TO BE MARKED.	10J137 J100 10L2 109-100 J100 70	J7 11W1 11WC J81 70 J8 9-2 9-3	P027R100 P027R110 L2=SYSFREEP027R120 P027R130 P027R140 P027R150 1W1=SUBLSTP027R160 P027R170 P027R175 P027R180 P027R185 P027R190 P027R195 P027R200 P027R210 P027R220 P027R230 P027R240 P027R250 P027R260 P027R270 P027R280 P027R290 P027R300 P027R310 P027R320 P027R330 P027R340 P027R350 P027R360 R
GEN SYS FREE VAR. TO REPLACE. H5+ MEANS NOT ENOUGH FREE VAR.	109-200 J100 11W3 109-200 J100 11W3 J71 9-1	J33 40HC J133 70 J8 11W2 J6 J64 11W2 J60 J60	P027R200 P027R210 P027R220 P027R230 P027R240 P027R250 P027R260 P027R270 P027R280 P027R290 P027R300 P027R310 P027R320 P027R330 P027R340 P027R350 P027R360
FIND MAIN SEGMENT. IF NONE, SKIP IT. REPLACE IN MEX FROM 1W1.	11W1 J71 9-1	J33 40HC J133 70 J8 11W2 J6 J64 11W2 J60 J60	P027R200 P027R210 P027R220 P027R230 P027R240 P027R250 P027R260 P027R270 P027R280 P027R290 P027R300 P027R310 P027R320 P027R330 P027R340 P027R350 P027R360
GEN FREE VAR TO BE UNMARKED.	20W2 0		
ERASE CREATED FREE LIST.	9-100	31HC J8	
ERASE BOUND LIST. 9-100 SUBPROCESS, INSERT (0) AFTER SYMBOL IN 1W2 IF (0) IS UNMARKED. THEN ADVANCE TO NEXT AFTER CELL HOLDING INSERTED SYMBOL.	1 P28 9-2	10WC J17 60WC P8 70 11WC J60 20WC 70J19 12WC P9 709-1 11WC J18 70J19 12WC P8 70 12WC 10J18	10WC J17 60WC P8 J19 11WC J60 20WC 70J19 12WC P9 709-1 11WC J18 70J19 12WC P8 J2 12WC 10J18
QUIT, H5- MEANS QUIT GENERATOR. 9-200 SUBPROCESS, UNMARK PROCESSED.	9-200		
GENERATE LOCATIONS OF FREE VARIABLES WITHIN SEGMENT (1) FOR PROCESS (0).	1		
TEST IF INPUT SEGMENT IS VARIABLE. IF SO, QUIT.	9-2	70 11WC J60 20WC 70J19 12WC P9 709-1 11WC J18 70J19 12WC P8 70 12WC 10J18	70 11WC J60 20WC 70J19 12WC P9 709-1 11WC J18 70J19 12WC P8 70 12WC 10J18
LOCATE NEXT SUBSEGMENT.			
TEST IF SUBSEGMENT IS FREE VAR.			
IF SO, GENERATE LOCATION.	9-3		
TEST IF SUBSEGMENT IS VARIABLE. IF NOT, GENERATE SUBSEGMENT.	9-1		

P29 GENERATE LOCATIONS OF BOUND VARIABLES WITHIN SEGMENT (1) FOR PROCESS (0).	1	P29	10W0	P28	9-3	P028R200 R
TEST IF INPUT SEGMENT IS VARIABLE. IF YES, QUIT H5+.			J17			P029R000
IF NO,			60W0			P029R010
LOCATE NEXT SEGMENT			P8			P029R020
IF NONE, QUIT, H5+.			70	J19		P029R030
TEST IF FREE VARIABLE. IF YES, GET NEXT.	9-2		11W0			P029R040
IF NO,			J60			P029R050
TEST IF BOUND VARIABLE. IF NO, GENERATE ON SEGMENT.			20W0			P029R060
IF YES, FEED LOCATION TO PROCESS.			70J19			P029R070
IF H5-, SUBPROCESS SAID QUIT.	9-3		12W0			P029RC80
INPUT SEGMENT.	9-1		P9			P029R090
INPUT PROCESS.			70	9-2		P029R100
GENERATE LOCATIONS OF BND. VAR.			12W0			P029R110
			P8			P029R120
P30 CREATE LIST OF FREE VARIABLES IN TEX (0).	1	P30	709-1			P029R130
FIND MAIN SEGMENT			11W0			P029R140
IF NONE, CLEAN UP, QUIT.			J18			P029R150
GEN. LOCATIONS OF FREE VARIABLES.			70J19	9-2		P029R160
SUBPROCESS, ADD FREE VARIABLE 2HC TO LIST 1WC IF NOT ON.	9-1		12W0			P029R170
	9-100		10CJ18			P029R180
QUIT, H5+ FOR GEN.			P29	9-3		P029R190
			J90			P029R200
P30 CREATE LIST OF FREE VARIABLES IN TEX (0).			J50		1WC=LIST	R
FIND MAIN SEGMENT.			J81			P030R000
IF NONE, CLEAN UP, QUIT.			709-1			P030R010
GEN. LOCATIONS OF FREE VARIABLES.			109-100			P030R020
SUBPROCESS, ADD FREE VARIABLE 2HC TO LIST 1WC IF NOT ON.	9-1		P28			P030R030
QUIT, H5+ FOR GEN.			9-1	11W0	J30	P030R040
P31 CREATE LIST OF BOUND VARIABLES. IN TEX (0).	1	P31	52HC			P030R050
FIND MAIN SEGMENT.			11WC			P030R060
IF NONE, CLEAN UP, QUIT.			J6			P030R070
GENERATE LOCATIONS OF BOUND VAR.			706			P030R080
SUBPROCESS, ADD BOUND VARIABLE 2HO TO LIST 1WC IF NOT ON.	9-1		J66	J4		P030R090
QUIT, H5+ FOR GEN.			1			P030R100
P50 CONVERT LOGIC EXPRESSION (0) TO INTERNAL (TREE) FORM IF IN EXTERNAL (LIST) FORM. ENTIRE		P50	J90			R
			J50			P031R000
			J81			P031R010
			709-1			P031R020
			109-100			P031R030
			P29			P031R040
			9-1	11W0	J30	P031R050
			52HC			P031R060
			11WC			P031R070
			J6			P031R080
			706			P031R090
			J66	J4		P031R100
			1			R
			40HO			P050R000
			P15			P050R010
			70	J8		P050R020

EXPRESSION MUST BE ENCLOSED IN PARENTHESES. NO OUTPUT. H5- MEANS FAILURE.	J41 60W0 P51 11W0 P52 70J31 60W1 51W0 J75 J71 11W0 11W1 J64 11W0 10Q15 10Q15	J31	P050R030 P050R040 P050R050 P050R060 P050R070 P050R075 P050R080 P050R090 P050R100 P050R110 P050R120 P050R130 P050R140 P050R190 P050R200 P050R210 P050R220 R
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.	J11	J31	
P51 REPLACE ALL DELIMITED EXTERNAL CONNECTIVES IN EXPRESSION (0) IF (0) IS IN EXTERNAL LIST FORM.	P51 1 40HO P15 70 J42 60WC 9-10 11K7 J62 709-0 60W1 J60 60W2 J60 709-0 12HO 11K7 J2 709-10 J68 10L8 12W2 J10 709-1 21W2 9-1 11W1 J68 11W0 9-10 30HO J32 1 P52 J41 20W1 9-100 J31 04JC 11W1	J8	P051R000 P051R005 P051R010 P051R015 1W0=LIST P051R020 P051RC25 P051R030 P051R040 1W1=LOC1STP051R050 P051R060 1W2=LOCONN P051R070 P051R080 P051R090 P051R100 P051R110 P051R120 P051R130 P051R140 P051R170 P051R180 P051R190 P051R200 P051R210 P051R220 P051R230 P051R240 P051R250 R
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.	9-0	9-10 J32	
P52 CREATE MAIN SEGMENT FROM LIST (0). H5- MEANS NO OUTPUT DUE TO BAD EXPRESSION. 9-100 SUBPROCESS CREATE NEXT SGMNT. H5- MEANS NO OUTPUT.	P52 9-100 04JC 11W1	J41 20W1 9-100 J31 04JC 11W1	P052R000 1W1=CURLOC P052R020 P052R030 P052R040 1W1=CURLOC P052R050

LOCATE FIRST OF EXPRESSION	J60	P052R060
IF NONE, QUIT.	20W1	P052R070
	709-101	P052R080
	12W1	P052R090
	10)	P052R100
TEST IF OPENING PAREN.	J2	P052R110
IF YES, BUILD SEGMENT.	70 9-110	P052R120
	12W1	P052R130
	11K2	1K2= NOT P052R140
TEST IF NOT.	J2	P052R150
IF YES, BUILD SEGMENT.	70 9-120	P052R160
	12W1	P052R170
TEST IF VARIABLE.	P8	P052R180
IF NO QUIT.	709-101	P052R190
OUTPUT VARIABLE.	12W1 0	P052R200
	9-101 11W1	P052R205
	10/14 J63	P052R210
TAKE ERROR ACTION, THEN	9-103 9-101	P052R220
ERASE USELESS SEGMENT.	9-102 11W0	P052R230
	J72 J30	P052R240
OUTPUT SEGMENT.	9-104 11W0	P052R250
	J136 J30	P052R260
BUILD SEGMENT	9-110 9-200	1W0=NEWSEG P052R270
CREATE 1ST SUBSEGMENT.	9-100	P052R280
IF NONE, CLEAN UP, QUIT.	709-102	P052R290
INSERT 1ST SUBSEGMENT.	9-300	P052R300
	11W1	P052R310
LOCATE NEXT SYMBOL.	J60	P052R320
	60W1	P052R330
	52W1	P052R340
TEST IF CONNECTIVE.	P7	P052R350
IF NOT, CLEAN UP, QUIT.	709-103	P052R360
	12W1	P052R370
INSERT CONNECTIVE.	21W0	P052R380
CREATE 2ND SUBSEGMENT.	9-100	P052R390
IF NONE, CLEAN UP, QUIT.	709-102	P052R400
INSERT 2ND SUBSEGMENT.	9-300	P052R410
	11W1	P052R420
LOCATE NEXT SYMBOL.	J60	P052R430
	60W1	P052R440
	52W1	P052R450
	10)	P052R460
TEST IF CLOSING PAREN.	J2	P052R470
IF NO, CLEANUP, QUIT.	709-103 9-104	P052R480
BUILD NOTTED SEGMENT.	9-120 9-200	P052R490
	12W1	P052R500
NOT THE HEAD.	21W0	P052R510
CREATE SUBSEGMENT.	9-100	P052R520
IF NONE, CLEAN UP AND QUIT.	709-102	P052R530
INSERT	9-300 9-104	P052R540
9-200 SUBPROCESS, SET UP EMPTY SEG.	9-200 J90 J50	P052R550

9-300 SUBPROCESS, INSERT SEGMENT.	9-300	11W0	J6	J65	P052R560 P052R570 R
P55 LOCATE SUBLIST FOLLOWING DATA TERM (0) ON LIST (1)) H5+ MEANS PUTPUT (0) IS CELL HOLDING SUBLIST.	P55	J41	20W0		P055R000 P055R010 P055R020 P055R030
H5- MEANS OUTPUT (0) IS CELL AFTER WHICH TO INSERT.	9-3	60W1	J60		P055R040 P055R050 P055R060
TEST IF PAST. IF YES, QUIT, H5-.		70J31	12H0		P055R070 P055R080
		12H0	11W0		P055R090 P055R100
TEST IF EQUAL. IF YES, QUIT, H5+. IF NO, MOVE DOWN THE LIST.		J114	70	9-1	P055R110 P055R120 P055R130
		70	J60		P055R140
SET UP CELL TO INSERT AFTER.	9-1	51W1	70J7	9-3	P055R150 P055R160
SET UP CELL HOLDING SUBLIST.	9-2	J3	J60		P055R170 P055R180
		J31	70J7	J31	R
Q1 FIND CONNECTIVE OF SEGMENT (0).	Q1	J80			Q001R000 Q001R010
		700			Q001R020
		40H0			Q001R030
TEST IF IT IS A CONNECTIVE.		P7	70J8	0	Q001R040
		70J8			R
Q2 FIND NO. OF LEVELS OF TEX (0). H5- MEANS DEFECTIVE EXPRESSION.	Q2	40H0			Q002R000 Q002R010
FIND VALUE ON DESCRIPTION LIST. IF NONE, GO COUNT LEVELS. IF THERE, CLEAN UP, QUIT +.		10Q2			Q002R020 Q002RC30
		J10			Q002R040
		709-0			Q002R050
		J6	J8		Q002R060
CREATE LEVEL DATA TERM = ZERO.	9-0	10N10			Q002R070 Q002R080
		J120			Q002R090
		J136			1W0=COUNTRQ002R100
		40H0			1W1=LEVEL Q002R110
CREATE COUNTER.		J120			1W2=TEX Q002R120
SAVE COUNTER, LEVEL, TEX.		J52			Q002R130
FIND MEX.		11W2			Q002R140
IF NONE, CLEANUP, QUIT-.		J81			Q002R150
IF THERE, COUNT LEVELS.	9-1	709-1			Q002R160
		9-100			Q002R170
ERASE COUNTER.		11W0			Q002R180
INPUT LEVEL,		J9			Q002R190
IF H5-, ERASE LEVEL, QUIT-.		11W1			Q002R200
IF H5+,		709-2			Q002R210
ASSIGN LEVEL 1W1 AS VALUE OF Q2 OF TEX 1W2.		11W2			
		11W1			
		10Q2			

QUIT +, OUTPUT (0) IS LEVEL.		J11	J32	Q002R220
	9-2	J9	J32	Q002R230
9-100 SUBPROCESS. COUNT SUBLVELS.	9-100	11W0		Q002R240
		J120		Q002R250
CREATE COUNTER EQUAL TO THIS LEVEL.		J125		Q002R260
PRESERVE PREVIOUS COUNTER.		J50		Q002R270
		40H0		Q002R300
TEST IF SIMPLE VARIABLE.		P8		Q002R310
IF YES, UPDATE LEVEL.	70	9-102		Q002R320
IF NO, COUNT SUBLVELS,	109-100			Q002R330
THEN QUIT + OR -.	J100	9-101		Q002R340
UPDATE LEVEL.	9-102	51W1		Q002R350
		11W0		Q002R360
TEST IF COUNTER GREATER THAN LEVEL.		J115		Q002R370
IF NO, QUIT +.	709-103			Q002R380
IF YES,	11W0			Q002R390
SET LEVEL SAME AS COUNTER.		11W1		Q002R400
		J121		Q002R410
		30H0		Q002R420
SET H5+.	9-103	J4		Q002R430
ERASE COUNTER OF THIS LEVEL.	9-101	11W0		Q002R440
RESTORE PREVIOUS COUNTER, QUIT.		30W0	J9	Q002R450
				R
Q3 FIND NO. OF DISTINCT VARIABLES IN TOTAL EXPRESSION (0).	G3	40H0		Q003R000
		10Q3		Q003R010
		J10		Q003R020
FIND AS VALUE IN DESC. LIST.		709-0		Q003R030
IF THERE, CLEAN UP, QUIT +.	J6	J8		Q003R040
IF NONE, COUNT VARIABLES.	9-0	J42		Q003R050
		60WC		1W0=TEX
		P30		Q003R060
CREATE FREE LIST, SAVE IT,		60W1		1W1=FREE
COUNT IT,		J126		Q003R080
MARK COUNT LOCAL,		J136		Q003R090
AND SAVE FOR OUTPUT.		60W2		1W2=OUTPUT
		51W1		Q003R110
ERASE FREE LIST.		J71		Q003R120
		11W0		Q003R130
CREATE BOUND LIST, SAVE IT,		P31		Q003R140
COUNT IT,		60W1		1W1=BOUND
		J126		Q003R160
		40H0		Q003R170
		11W2		Q003R180
		11W2		Q003R190
ADD IT TO OUTPUT DATA TERM.		J110		Q003R200
		51W1		Q003R210
ERASE BOUND LIST.		J71		Q003R220
ERASE BOUND COUNT.		J9		Q003R230
		11W0		Q003R240
		11W2		Q003R250
		10Q3		Q003R260
				Q003R270

ASSIGN AS VALUE OF Q3.		J11		Q003R280	
CLEAN UP AND QUIT.		11W2		Q003R290	
		J32	J4	Q003R300	
				R	
Q4 FIND NO. OF VARIABLE PLACES IN TEX (0).	1	Q4	40H0	Q004R000	
FIND AS VALUE OF DESC. LIST. IF NONE, GO COUNT PLACES.			10Q4	Q004R010	
IF THERE, CLEAN UP, QUIT, H5+.			J10	Q004R020	
			709-0	Q004R030	
			J6	Q004R040	
SET UP THREE COPIES OF TEX NAME.	9-0	40H0	J8	Q004R050	
		40H0		Q004R060	
		J90		Q004R070	
		J136		Q004R080	
CREATE D.T. WITH VALUE = 0.		J124		Q004R090	
		J6		Q004R100	
		109-100		Q004R110	
GENERATE FREE LOCATIONS FOR TALLY.		P28		Q004R120	
		J6		Q004R130	
GENERATE BOUND LOCATIONS FOR TALLY.		109-100		Q004R140	
		P29		Q004R150	
		40WC		Q004R160	
SAVE OUTPUT D.T.		60W0		1W0=OUTPUTQ004R170	
		10Q4		Q004R180	
ASSIGN AS VALUE OF Q4 OF TEX.		J11		Q004R190	
DISCARD (0), TALLY (1).	9-100	11W0	J30	Q004R200	
		30H0	J125	Q004R210	
				R	
ATTRIBUTE--VARIABLE	1	Q5	10Q5	J10	Q005R000
				R	
ATTRIBUTE--FREE VARIABLE	1	Q6	10Q6	J10	Q006R000
				R	
ATTRIBUTE--EXTERNAL NAME	1	Q7	10Q7	J10	Q007R000
				R	
FIND PROBLEM NUMBER OF (0)	1	Q8	10Q8	J10	Q008R000
				R	
Q9 ATTRIBUTE--BOUND VARIABLE.	1	Q9	10Q9	J10	Q009R000
				R	
FIND PROBLEM (0) DERIVED FROM	1	Q10	10Q10	J10	Q010R000
				R	
FIND METHOD OF DERIVATION FOR (0)	1	Q11	10Q11	J10	Q011R000
				R	
FIND THEOREM USED FOR (0)	1	Q12	10Q12	J10	Q012R000
				R	
FIND PROVING THEOREM FOR (0)	1	Q13	10Q13	J10	Q013R000
				R	
Q14 FIND TYPE OF CONNECTIVE (0).	1	Q14	10Q14	J10	Q014RC00
				R	
Q15 ATTRIBUTE -- INTERNAL FORM.	1	Q15	1CQ15	J10	Q015R000
				R	
Q16 FIND EXTERNAL NAME OF (0) IN TABLE T10.	1	Q16	1CT10		Q016R000
		J6	J10	Q016R010	

		1		R
Q17 FIND LEVEL OF SUBSEGMENT REPLACEMENT IN TEX (0).	Q17	40W0		Q017R000
		60W0		Q017R010
		10Q17		Q017R020
FIND CURRENT LEVEL.		J10		Q017R030
IF NONE,		70	J30	Q017R040
FIND NUMBER OF LEVELS, IF NONE, QUIT -.		11WC		Q017R050
COPY,		Q2		Q017R060
SAVE ONE FOR OUTPUT,		70J30		Q017R065
		J120		Q017R070
AND ASSIGN AS CURRENT LEVEL.		40HC		Q017R080
		11WC		Q017R090
		J6		Q017R100
		10Q17		Q017R110
		30W0	J11	Q017R120
	1			R
Q18 FIND SUFFIX OF EXPRESSION (0).	Q18	10Q18	J10	Q018R000
	1			R
Q19 FIND CHARACTER SYMBOL FOR '0'.	Q19	10L9		Q019R000
		J6	J10	Q019R010

DATA HEADER	5	1	D -
FREE VARIABLE A	A	0	A000D000
	0		A000D010
	Q5		A000D020
	Q5		A000D030
	Q6		A000D040
	Q6		A000D050
	Q7		A000D060
		0	A000D070
	+21A		A000D080
FREE VARIABLE B	B	0	B000D000
	0		B000D010
	Q5		B000D020
	Q5		B000D030
	Q6		B000D040
	Q6		B000D050
	Q7		B000D060
		0	B000D070
	+21B		B000D080
FREE VARIABLE C	C	0	C000D000
	0		C000D010
	Q5		C000D020
	Q5		C000D030
	Q6		C000D040
	Q6		C000D050
	Q7		C000D060
		0	C000D070
	+21C		C000D080
FREE VARIABLE D	D	0	D000D000
	0		D000D010
	Q5		D000D020
	Q5		D000D030
	Q6		D000D040
	Q6		D000D050
	Q7		D000D060
		0	D000D070
	+21D		D000D080
FREE VARIABLE E	E	0	E000D000
	0		E000D010
	Q5		E000D020
	Q5		E000D030
	Q6		E000D040
	Q6		E000D050
	Q7		E000D060
		0	E000D070
	+21E		E000D080
FREE VARIABLE F	F	0	F000D000
	0		F000D010
	Q5		F000D020
	Q5		F000D030

		Q6	F000DC40
		Q6	F000D050
		Q7	F000D060
		0	F000D070
	+21F		F000D080
FREE VARIABLE G	G	0	G000D000
		0	G000D010
		Q5	G000D020
		Q5	G000D030
		Q6	G000D040
		Q6	G000D050
		Q7	G000D060
		0	G000D070
	+21G		G000D080
HC LUBRICATION.	H0	*1	H000D000
		*2	H000D010
		*3	H000D020
		*4	H000D030
IMPLIES	I	0	I000D000
		0	I000D010
		Q14	I000D020
		J4	I000D030
		Q7	I000D040
		0	I000D050
	+21I		I000D060
K0 SYMBOL FOR CHARACTER K.	K	0	K000D000
		0	K000D010
		Q7	K000D020
		0	K000D030
	+21K		K000D040
HOLDS 'OR'	K1	V0	K001D000
HOLDS 'NOT'	K2	-0	K002D000
HOLDS 'AND'	K3	*0	K003D000
HOLDS 'PROVEN EQUIVALENCE'	K4	=0	K004D000
HOLDS 'DEFINITIONAL EQUIVALENCE'	K5	=1	K005D000
HOLDS 'IMPLIES'	K6	I0	K006D000
K7 HOLDS CONNECTIVE DELIMITER.	K7	.	K007D000
K10 PREVIOUS PROBLEM NUMBER.	K10	01	K010D000
SUBSTITUTION COUNT	K11	+ 1	K011D000
EFFORT BASE (AND TOTAL).	K12	+ 1	K012D000
LIMIT ON NO. OF SUBPROBLEMS	K20	+ 1	K020D000
LIMIT ON NO. OF SUBSTITUTIONS	K21	+ 1	K021D000
LIMIT ON EFFORT	K22	+ 1	K022D000
	K30	F	K030D000
K31 DON'T PRINT REJECTS IF HOLDS NO	K31	NO	K031D000
K41 VALUE = METHOD COLUMN.	K41	01	K041D000
K42 VALUE = NAME COLUMN.	K42	01	K042D000
K43 VALUE = EXPRESSION COLUMN.	K43	01	K043D000
K44 VALUE = 'LIMIT' COLUMN.	K44	01	K044D000
K45 VALUE = 'ACTUAL' COLUMN.	K45	01	K045D000
K46 VALUE = 'REJECTED' COLUMN.	K46	01	K046D000

K47 VALUE = NAME OF NEW SUBPROBLEM	K47	01	11 COLUMN.	K047D000	
K48 VALUE = THM, METHOD COLUMN.	K48	01	43	K048D000	
K51 DATA TERM '('	K51	21(K051D000	
K52 DATA TERM ') ')	K52	21)		K052D000	
K53 DATA TERM '..'	K53	21..		K053D000	
K54 DATA TERM ',','	K54	21,		K054D000	
LO SYMBOL FOR CHARACTER L.	L	0		L000D000	
		0		L000D010	
		Q7		L000D020	
		0		L000D030	
		+21L		L000D040	
L1 TRUE THEOREMS AXIOMS DEFINITIONS	L1	0	0	L001D000	
LIST OF FREE VARIABLES	L2	0		L002D000	
		A0		L002DC10	
		B0		L002D020	
		C0		L002D030	
		D0		L002D040	
		E0		L002D050	
		F0		L002D060	
		G0	0	L002D080	
L3 PROBLEM LIST FOR MULTI PROB EXEC	L3	0	0	(M2)	L003D000
TRUE EXPRESSIONS MAPS	L4	0	0		L004D000
LIST DESCRIBED BY L4	L5	L4	0		L005D000
L6 LIST OF METHODS FOR ORIG PROBS	L6	0			L006D000
		M16		L006D010	
		M17		L006D020	
L7 LIST OF METHODS FOR PROBLEMS.	L7	0			L007D000
DETACHMENT.		M11		L007D010	
REPLACEMENT.		M13		L007D010	
FORWARD CHAINING.		M14		L007D010	
BACKWARD CHAINING.		M15		L007D010	
L8 DESCRIPTION LIST TABLE OF	L8		0		L008D000
DELIMITABLE EXTERNL CONNECTIVES		0		L008D010	
		=		L008D020	
		=1		L008D030	
L9 DESCRIPTION LIST TABLE OF	L9		0		L009D000
CHARACTER SYMBOLS FOR		0		L009D010	
READING TEXT.		0		L009D020	
		/10		L009D025	
		1		L009D030	
		/1		L009D040	
		2		L009D050	
		/2		L009D060	
		3		L009D070	
		/3		L009D080	
		4		L009D090	
		/4		L009D100	
		5		L009D110	
		/5		L009D120	
		6		L009D130	
		/6		L009D140	

	7	L009D150
	/7	L009D160
	8	L009D170
	/8	L009D180
	9	L009D190
	/9	L009D200
	H	L009D210
	/11	L009D220
	J	L009D230
	/12	L009D240
	W	L009D250
	/13	L009D260
UNTRIED PROBLEMS LIST	L10	0 0 L010D000
L11 FOUND PROBLEMS LIST.	L11	0 0 L011D000
MC SYMBOL FOR CHARACTER M.	M	0 M000D000
	0	M000D010
	Q7	M000D020
		M000D030
	+21M	M000D040
NC SYMBOL FOR CHARACTER N.	N	0 N000D000
	0	N000D010
	Q7	N000D020
		N000D030
	+21N	N000D040
1 INTEGER CONSTANTS.	N1	+01 1 N001D000
	N2	+01 2 N002D000
	N3	+01 3 N003D000
	N4	+01 4 N004D000
	N5	+01 5 N005D000
	N6	+01 6 N006D000
	N7	+01 7 N007D000
	N8	+01 8 N008D000
	N9	+01 9 N009D000
	N10	+01 0 N010D000
0	0	0000D000
OO SYMBOL FOR CHARACTER O.	O	0 0000D010
	0	0 0000D020
	Q7	0 0000D030
		0 0000D040
VARIABLE P	P	0 P000D000
	0	P000D010
	Q5	P000D020
	Q5	P000D030
	Q9	P000D033
	Q9	P000D037
	Q7	P000D040
		0 P000D050
	+21P	P000D060
VARIABLE Q	Q	0 Q000D000
	0	Q000D010
	Q5	Q000D020

		Q5	Q000D030
		Q9	Q000D033
		Q9	Q000D037
		Q7	Q000D040
		0	Q000D050
	+21Q		Q000D060
VARIABLE R	R	0	R000D000
		0	R000D010
		Q5	R000D020
		Q5	R000D030
		Q9	R000D033
		Q9	R000D037
		Q7	R000D040
		0	R000D050
	+21R		R000D060
VARIABLE S	S	0	S000D000
		0	S000D010
		Q5	S000D020
		Q5	S000D030
		Q9	S000D033
		Q9	S000D037
		Q7	S000D040
		0	S000D050
	+21S		S000D060
VARIABLE T	T	0	T000D000
		0	T000D010
		Q5	T000D020
		Q5	T000D030
		Q9	T000D033
		Q9	T000D037
		Q7	T000D040
		0	T000D050
	+21T		T000D060
T1 'GIVEN'	T1	0	T001D000
		0	T001D010
		21GIVEN	T001D020
T2 'PROOF FOUND.'	T2	0	T002D000
		9-1	T002D010
		9-2	T002D020
		9-3 0	T002D030
	9-1	21 PROO	T002D040
	9-2	21F FOU	T002D050
	9-3	21ND.	T002D060
T3 'SUBSTITUTION'	T3	0	T003D000
		9-1	T003D010
		9-2	T003D020
		9-3 0	T003D030
	9-1	21SUBST	T003D040
	9-2	21ITUTI	T003D050
	9-3	21ON	T003D060
T4 'Q.E.D.'	T4	0	T004D000

		9-1	T004D010
		9-2 0	T004D020
	9-1	21Q•E•D	T004D030
	9-2	21.	T004D040
T5 LIST OF ONE BLANK D•T•	T5	0	T005D000
		0	T005D010
		21	T005D020
T6 'NO PROOF FOUND'	T6	0	T006D000
		9-1	T006D010
		9-2	T006D020
		9-3 0	T006D030
	9-1	21NO PR	T006DC40
	9-2	21OOF F	T006D050
	9-3	21OUND	T006D060
T7 'EFFORT'	T7	0	T007D000
		9-1	T007D010
		9-2 0	T007D020
	9-1	21EFFOR	T007D030
	9-2	21T	T007D040
T8 'SUBPROBLEMS'	T8	0	T008D000
		9-1	T008DC10
		9-2	T008D020
		9-3 0	T008DC30
	9-1	21SUBPR	T008D040
	9-2	21OBLEM	T008D050
	9-3	21S	T008D060
T9 'SUBSTITUTIONS'	T9	C	T009D000
		9-1	T009D010
		9-2	T009D020
		9-3 0	T009D030
	9-1	21SUBST	T009D040
	9-2	21ITUTI	T009D050
	9-3	21ONS	T009D060
T10 DESC. LIST TABLE OF NAMES.	T10	9-0 0	T010D000
	9-0	0	T010D010
		M11	T010D020
		T12	T010D030
		M12	T010D040
		T3	T010D050
		M13	T010D060
		T13	T010D070
		M14	T010D080
		T14	T010D090
		M15	T010D100
		T15	T010D110
		M16	T010D120
		T16 0	T010D130
T12 'DETACHMENT'	T12	0	T012D000
		9-1	T012D010
		9-2 0	T012D020
	9-1	21DETAC	T012D030

T13 'REPLACEMENT'	9-2	21HMENT	T012D040
	T13	0	T013D000
		9-1	T013D010
		9-2	T013D020
		9-3	T013D030
	9-1	21REPLA	T013D040
	9-2	21CEMEN	T013D050
	9-3	21T	T013D060
T14 'FORWARD CHAINING'	T14	0	T014D000
		9-1	T014D010
		9-2	T014D020
		9-3	T014D030
		9-4	T014D040
	9-1	21FORWA	T014D050
	9-2	21RD CH	T014D060
	9-3	21AININ	T014D070
	9-4	21G	T014D080
T15 'BACKWARD CHAINING'	T15	0	T015D000
		9-1	T015D010
		9-2	T015D020
		9-3	T015D030
		9-4	T015D040
	9-1	21BACKW	T015D050
	9-2	21ARD C	T015D060
	9-3	21HAINI	T015D070
	9-4	21NG	T015D080
T16 'SUBLEVEL REPLACEMENT'	T16	0	T016D000
		9-1	T016D010
		9-2	T016D020
		9-3	T016D030
		9-4	T016D040
	9-1	21SUBLE	T016D050
	9-2	21VEL R	T016D060
	9-3	21EPLAC	T016D070
	9-4	21EMENT	T016D080
T19 'REJECTED PROBLEM'	T19	0	T019D000
		9-1	T019D010
		9-2	T019D020
		9-3	T019D030
		9-4	T019D040
	9-1	21REJEC	T019D050
	9-2	21TED P	T019D060
	9-3	21ROBLE	T019D070
	9-4	21M	T019D080
T20 'ACTUAL'	T20	0	T020DC00
		9-1	T020D010
		9-2	T020D020
	9-1	21ACTUA	T020D030
	9-2	21L	T020D040
T21 'LIMIT'	T21	0	T021D000
			T021DC10

		21LIMIT	T021D020
T22 'TO PROVE'	T22	0	T022D000
		9-1	T022D010
		9-2 0	T022D020
	9-1	21TO PR	T022D030
	9-2	21OVE	T022D040
T23 'REMEMBER PROVED THEOREM'	T23	0	T023D000
		9-1	T023D010
		9-2	T023D020
		9-3	T023D030
		9-4	T023D040
		9-5	T023D050
		9-6 0	T023D060
	9-1	21 RE	T023D070
	9-2	21MEMBE	T023D080
	9-3	21R PRO	T023D090
	9-4	21VED T	T023D100
	9-5	21HEORE	T023D110
	9-6	21M	T023D120
T24 'BAD EXPRESSION'	T24	0	T024D000
		9-1	T024D010
		9-2	T024D020
		9-3 0	T024D030
	9-1	21BAD E	T024D040
	9-2	21XPRES	T024D050
	9-3	21SION	T024D060
U0 SYMBOL FOR CHARACTER U.	U	0	U000D000
		0	U000D010
		Q7	U000D020
			U000D030
		+21U	U000D040
OR	V	0	V000D000
		0	V000D010
		Q14	V000D020
		J4	V000D030
		Q7	V000D040
			V000D050
		+21V	V000D060
X0 SYMBOL FOR CHARACTER X.	X	0	X000D000
		0	X000D010
		Q7	X000D020
			X000D030
		+21X	X000D040
Y0 SYMBOL FOR CHARACTER Y.	Y	0	Y000D000
		0	Y000D010
		Q7	Y000D020
			Y000D030
		+21Y	Y000D040
Z0 SYMBOL FOR CHARACTER Z.	Z	0	Z000D000
		0	Z000D010
		Q7	Z000D020

			0	Z000D030
		+21Z	0	Z000D040
NOT	-		0	-000D000
		0		-000D010
		Q14		-000D020
		J3		-000D030
		Q7		-000D040
			0	-000D050
		+21-	0	-000D060
AND	*		0	*000D000
		0		*000D010
		Q14		*000D020
		J4		*000D030
		Q7		*000D050
			0	*000D050
		+21*	0	*000D060
PROVEN EQUIVALENCE	=		0	=000D000
		0		=000D010
		Q14		=000D020
		J4		=000D030
		Q7		=000D040
			0	=000D050
		+21=	0	=000D060
=1 DEFINITIONAL EQUIVALENCE	=1		0	=001D000
		0		=001D010
		Q14		=001D020
		J4		=001D030
		Q7		=001D040
			0	=001D050
		+21•=•	0	=001D060
+0 SYMBOL FOR PLUS SIGN.	+		0	+000D000
		0		+000D010
		Q7		+000D020
			0	+000D030
		+21+	0	+000D040
/0 SYMBOL FOR SLASH•	/		0	/000DC00
		0		/000D010
		Q7		/000D020
			0	/000D030
		+21/	0	/000D040
/1 SYMBOL FOR DIGIT 1•	/1		0	/001D000
		0		/001D010
		Q7		/001D020
			0	/001D030
		+211	0	/001D040
/2 SYMBOL FOR DIGIT 2•	/2		0	/002D000
		0		/002D010
		Q7		/002D020
			0	/002D030
		+212	0	/002D040
/3 SYMBOL FOR DIGIT 3•	/3		0	/003D000

		C	/003D010
		Q7	/003D020
		0	/003D030
		+213	/003D040
/4 SYMBOL FOR DIGIT 4.	/4	0	/004D000
		C	/004D010
		Q7	/004D020
		0	/004D030
		+214	/004D040
/5 SYMBOL FOR DIGIT 5.	/5	0	/005D000
		0	/005D010
		Q7	/005D020
		0	/005D030
		+215	/005D040
/6 SYMBOL FOR DIGIT 6.	/6	0	/006D000
		0	/006D010
		Q7	/006D020
		0	/006D030
		+216	/006D040
/7 SYMBOL FOR DIGIT 7.	/7	0	/007D000
		C	/007D010
		Q7	/007D020
		0	/007D030
		+217	/007D040
/8 SYMBOL FOR DIGIT 8.	/8	0	/008D000
		C	/008D010
		Q7	/008D020
		0	/008D030
		+218	/008D040
/9 SYMBOL FOR DIGIT 9.	/9	0	/009D000
		0	/009D010
		Q7	/009D020
		0	/009D030
		+219	/009D040
/10 SYMBOL FOR DIGIT 0.	/10	0	/010D000
		0	/010D010
		Q7	/010D020
		0	/010D030
		+210	/010D040
/11 SYMBOL FOR CHARACTER J.	/11	0	/011D000
		0	/011D010
		Q7	/011D020
		0	/011D030
		+21H	/011D040
/12 SYMBOL FOR CHARACTER W.	/12	0	/012D000
		0	/012D010
		Q7	/012D020
		0	/012D030
		+21J	/012D040
/13 SYMBOL FOR CHARACTER H.	/13	0	/013D000
		0	/013D010

		Q7	/013D020	
		0	/013D030	
		+21W	/013D040	
/14 DUMMY CHARACTER SYMBOL	/14	0	/014D000	
		0	/014D010	
		Q7	/014D020	
		0	/014D030	
		+21/UGH/	/014D040	
/16 DUMMY EXPRESSION -- 'DEFINITIONS'.	/16	9-1	/016D000	
		9-2	0	/016D010
	9-1	0	/016D020	
		Q15	/016D030	
		Q15	/016D040	
		Q7	/016D050	
		0	/016D060	
EXTERNAL NAME CONNECTIVE 'I'.		21	/016D070	
	9-2	I0	/016D080	
		9-10	/016D090	
		9-20	0	/016D100
DUMMY VARIABLE 'DEFIN'.	9-10	0	/016D110	
		0	/016D120	
		Q5	/016D130	
		Q5	/016D140	
		Q9	/016D150	
		Q9	/016D160	
		Q7	/016D170	
		0	/016D180	
EXTERNAL NAME • DUMMY VARIABLE 'TIONS'.	9-20	21DEFIN	/016D190	
		0	/016D200	
		0	/016D210	
		Q5	/016D220	
		Q5	/016D230	
		Q9	/016D240	
		Q9	/016D250	
		Q7	/016D260	
		0	/016D270	
EXTERNAL NAME • (SYMBOL FOR LEFT PAREN.	(21TIONS	/016D280	
		0	(000D000	
		0	(000D010	
		Q7	(000D020	
		K51	(000D030	
' SYMBOL FOR QUOTE MARK.	'	0	'000D000	
		0	'000D010	
		Q7	'000D020	
		0	'000D030	
) SYMBOL FOR RIGHT PAREN.)	+21'	'000D040	
		0)000D000	
		0)000D010	
		Q7)000D020	
		K52)000D030	
, SYMBOL FOR COMMA •	,	0	,000D000	

•	0		,000D010
	Q7		,000D020
	K54	0	,000D030
•0 SYMBOL FOR PERIOD		0	,000D000
	0		,000D010
	Q7		,000D020
	K53	0	,000D030
\$0 SYMBOL FOR DOLLAR SIGN.	\$	0	\$000D000
	0		\$000D010
	Q7		\$000D020
		0	\$000D030
		+21\$	\$000D040

EXECUTIVE HEADER	5	R	-
	1	R	
	X1	11W26	X001R000
		10X23	X001R010
		J73	X001R020
SET UP TRAPS.		J76	X001R030
		50X21	X001R040
MARK TO TRACE.		10J147	X001R050
		J100	X001R060
		10X22	X001R070
MARK TO PROPAGATE TRACE.		10J148	X001R080
GET NEXT TRUE TEX FROM INPUT UNIT IF ANY LEFT.	9-1	J100	X001R090
		M89	X001R100
CONVERT TO TREE FORM.		709-10	X001R110
		40HC	X001R120
ADD TO SET OF TRUE EXPRESSION.		P50	X001R130
TAKE ACTION, TRY FOR ANOTHER	9-2	709-2	X001R140
		M50 9-1	X001R150
SKIP TWO LINES.	9-10	9-100 9-1	X001R160
		J154	X001R170
GET NEXT PROBLEM TEX.	9-11	J155	X001R180
IF NO MORE, GO TRY PROOFS.		J155	X001R190
CONVERT TO INTERNAL.		M89	X001R200
		70M2	X001R210
PRINT EXPRESSION.		40HC	X001R215
ADD TO LIST OF PROBLEMS.		P50	X001R220
BAD INPUT ACTION.		709-12	X001R230
		40HC	X001R240
		M70	X001R250
		10L3	X001R260
X10 INVOKE FULL TRACE.		J6	X001R270
		J65 9-11	X001R280
	9-12	9-100 9-11	X001R290
	9-100	40HC	X001R300
		M88	X001R310
		40HC	X001R320
		J15	X001R330
		J75 J72	X001R340
	1	R	
	X9	10N2	X009R000
		J166 J165	X009R010
	1	R	
X11 REVOKE CURRENT TRACE MODE IF ANOTHER EXISTS.	X10	10N1	X010R000
		J120	X010R010
		40W31	X010R020
		20W31	X010R030
		X19 0	X010R040
	1	R	
	X11	40H5	X011R000
		10W31	X011R010

X14 SAVE FOR RESTART ON INTERRUPT USING UNIT 3.	J78		X011R020	
	709-0		X011R030	
	11W31		X011R040	
	30W31		X011R050	
	X19		X011R060	
	30H5	J9	X011R070	
9-0	30H5	0	X011R080	
1	X13	40H0	J150	R
1	X14	40H5		X014R000
	10N3		X014R010	
	J3		X014R020	
	J166		X014R020	
	70	J7	X014R030	
	30H5	0	X014R040	
1	X15	10L4	J150	R
1	X19	03J0	0	X019R000
5		X9	R	
X19 MONITOR POINT FORCER. SAVE FOR RESTART				

REFERENCES

1. Newell, Allen, ed., Information Processing Language-V Manual, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1961.
2. Newell, A., and H. A. Simon, The Logic Theory Machine: A Complex Information Processing System, The RAND Corporation, P-868, July 12, 1956. Published in the IRE Transactions on Information Theory, Vol. IF-2, September 1956.
3. Newell, A., J. C. Shaw, and H. A. Simon, Empirical Explorations of the Logic Theory Machine: A Case Study in Heuristics, The RAND Corporation, P-951, March 14, 1957. Published in the Proceedings of the Western Joint Computer Conference, IRE, February 1957.
4. Newell, A., and J. C. Shaw, Programming the Logic Theory Machine, The RAND Corporation, P-954, February 28, 1957. Published in the Proceedings of the Western Joint Computer Conference, IRE, February 1957.
5. Whitehead, Alfred North, and Bertrand Russell, Principia Mathematica, Vol. 1, 2nd ed., Part I, "Mathematical Logic," University Press, Cambridge, England, 1927, Sec. A., "The Theory of Deduction," pp. 90-126. Also published in soft-cover edition (\$1.95) by University Press, Cambridge, England (to *56).
6. Wang, H., "Toward Mechanical Mathematics," IBM J. Res. & Develop., Vol. 4, No. 1, January 1960, pp. 2-22.
7. Wang, H., "Proving Theorems by Pattern Recognition, I," Communications of the ACM, Vol. 3, No. 4, April 1960.
8. Minsky, M., "Steps Toward Artificial Intelligence," Proceedings of the IRE, January 1961, pp. 21-23.
9. Newell, A., J. C. Shaw, and H. A. Simon, The Process of Creative Thinking, The RAND Corporation, P-1320, January 28, 1959, pp. 21-49.

10. Minsky, op. cit., pp. 9-10.
11. Newell, Allen, Some Problems of Basic Organization In Problem-Solving Programs, The RAND Corporation, RM-3283-PR, December 1962.
12. Newell, Allen, Learning, Generality, and Problem-Solving, The RAND Corporation, RM-3285-1-PR, February 1963.

IPL-V CODING SHEET

Problem No.

Programmer

Date _____

Page _____ of _____