A PROCEDURAL APPROACH TO CONSTRUCTIONS IN EUCLIDEAN GEOMETRY

by

BRIAN V. FUNT

B.Sc., University of British Columbia, 1971

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF

THE REQUIREMENTS FOR THE DEGREE OF

MASTER OF SCIENCE

in the Department

of

Computer Science

We accept this thesis as conforming

to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA

October, 1973

Abstract

A problem solving program capable of handling high school level Euclidean geometry straight-edge and compass constructions has been written. Figures are constructed by discovering, for the points composing them, the loci which satisfy the given sets of constraints. The representation of geometric knowledge is procedural. The relation to theorem proving in geometry, and aspects of the language PLANNER, which was used in the implementation of the program, are discussed.

Table Of Contents

Table Of Figures

Acknowledgement

I wish particularily to thank Dr. Raymond Reiter, my thesis supervisor, for his wealth of ideas and illuminating criticsm; Dr. Richard Rosenberg who originally interested me in Artificial Intelligence; and Karen for her love and inspiration.

Introduction

One of the ancient and traditional realms of problem solving is plane geometry. The continued role of geometry in the high school curriculum is justified not because of its theoretical importance, but because it contains a wealth of problems for the teaching and exercise of rigorous thought. Plane geometry and diagrams are inseparable, and this fact lends a concreteness to geometry which is absent in many of the other domains of rigorous thinking. Geometry problems are not 'toy problems'; they are problems which have been studied for millennia.

The characteristics which make geometry such an excellent area for the teaching of problem solving also make it good for the study of problem solving. Systems for proving theorems in geometry have been developed by Gelernter,[1,2] and Goldstein.[3]

---

[1] H. Gelernter et al, "Empirical Explorations of the Geometry-Theorem Proving Machine," Computers and Thought, ed. E.A. Feigenbaum and J. Feldman, (New York: McGraw Hill, 1963) pp.153-163.

[2] H. Gelernter, "Realization of a Geometry-Theorem Proving Machine," Computers and Thought, pp.134-152.

[3] Ira Goldstein, Elementary Geometry Theorem Proving, (Cambridge, Mass.: Massachusetts Institute of Technology, 1973) AI Memo no. 280.

Work has also been done on related problems: Wong[4] has devised a set of heuristics for constructions in a proof; Price[5] has examined the problem of drawing a diagram to satisfy the constraints specified in the hypothesis of a theorem.

What is considered here is the straight-edge and compass construction problem. Given a complete and consistent set of geometric constraints, the problem is to construct the figure which satisfies them. This problem relates to the other problems mentioned. Performing a construction is a form of theorem proving; the theorem being proven is that there exists a solution figure satisfying the constraints. In addition, some theorem proving ability is required for the proof of lemmas extending what is given to other facts concerning the solution figure or other auxiliary figures used in its construction.

A procedural representation is used for the required geometric knowledge. Each definition or theorem is coded as a procedure which determines whether its conditions are met and its result applicable. This procedural knowledge is expressed

---

[4] Richard Wong, Construction Heuristics for Geometry and a Vector Algebra Representation of Geometry, (Cambridge, Mass.: Massachusetts Institute of Technology, 1972) Technical Memo no.28.

[5] Keith Price, "Satisfying Geometric Constraints," (Cambridge, Mass.: Massachusetts Institute of Technology, 1971) unpublished Bachelor's Paper.

in such a way as to aid in the search for the loci of points which satisfy the constraints of the problem. When appropriate loci are found the problem is solved.

The author has written a program, called the CONSTRUCTOR, in MICRO-PLANNER[6] (the subset of Hewitt's language PLANNER[7] which has thus far been implemented) for the solution of straight-edge and compass construction problems. The program accepts the problem description in restricted English, and produces an algorithm described in English for the creation of the solution figure.

Polya,[8] who has studied problem solving in order to teach it, uses the construction problem as a medium for the communication of some 'patterns' of reasoning. By pointing out the common thread running through the solutions of many problems he hopes that the reader will, after working many exercise

---

[6] G.J. Sussman et al, MICRO-PLANNER Reference Manual, (Cambridge, Mass.: Massachusetts Institute of Technology, 1971) AI Memo no. 203A.

[7] Carl Hewitt, Description and Theoretical Analysis (Using Schemata) of PLANNER: A Language for Proving Theorems and Manipulating Models in a Robot, (Cambridge, Mass.: Massachusetts Institute of Technology, 1972) Revised Ph.D. Dissertation, AI Technical Report no. 258.

[8] George Polya, Mathematical Discovery: On Understanding, Learning, and Teaching Problem Solving, (New York: John Wiley and Sons, 1962) Vol. 1.

problems, internalize the underlying pattern of reasoning. This work is an attempt to incorporate such patterns as an integral part of a problem solving program.

I Background And Preliminary Remarks

I.1 Related Work

Although work has been done on other aspects of plane geometry by researchers in Artifical Intelligence, no exploration has been made of the straight-edge and compass construction problem considered here. The main emphasis has been on machine proof of geometry theorems, the earliest and most familiar work in this regard being that of Gelernter and his co-workers.[9] The most interesting feature of the Geometry Machine implemented by Gelernter, and the chief reason for its success in proving moderately difficult theorems, is the introduction of semantics into the proof process by the use of a diagram as a model.

A MICRO-PLANNER program called BTP (Basic Theorem Prover) has been written by Goldstein.[10] A procedural approach is used to incorporate and extend the ideas embedded in the Geometry Machine. The established geometry theorems and definitions of which BTP's knowledge is composed, are expressed as PLANNER

[9] Gelernter et al, Computers and Thought, pp.134-163.

[10] Goldstein, Elementary Geometry Theorem Proving.

procedures which Goldstein calls 'experts'. It appears that the effort involved in the implementation of BTP was substantially less than that of the Geometry Machine. This saving is attributable to the procedural approach which allowed Goldstein to concentrate more on the mathematics and less on the programming. The procedural approach has been adopted in this thesis. Goldstein also discusses a 'plausible move generator' which would decide the order in which the 'experts' should be tried rather than simply letting the right one be found by PLANNER failure and backup. BTP, however, is all that Goldstein has implemented thus far. Even without the 'plausible move generator', BTP has been able to prove all the theorems which the Geometry Machine proved.

Neither the Geometry Machine nor BTP create the diagram which they use to filter invalid subgoals. A diagram must be presented to both programs by the user of the system. Price has explored the problem of creating a diagram from the constraints implicit in the hypothesis of a theorem, and has outlined but not implemented, an algorithm for solving the problem. This problem may at first appear to be the same or very similar to the straight-edge and compass construction problem considered in this work; however, there are several crucial differences. The constraints contained in the hypothesis of a theorem in general do not totally constrain the figure. It is possible that one or more lengths or angles in the figure may be chosen freely.

Price thinks of the number of such choices as the degree of freedom of the diagram. In a straight-edge and compass construction problem the degree of freedom of the solution figure is zero.

Whereas the method of construction in a straight-edge and compass problem is restricted as implied by the name, this is not the case when drawing a diagram for a theorem prover; any technique, strategy, or tool can be used. Price suggests calling a construction routine based upon the heuristics Polya discusses in relation to the straight-edge and compass construction problem.

> "In the actual construction of a figure, given some element to use, I depend on the work of G. Polya in his chapter on solving construction problems."[11]

Although these heuristics would undoubtedly be useful in the problem Price considers, they should be generalized to eliminate the restriction of loci to straight lines and circles. The main thrust of Price's algorithm is to analyse the degree of freedom of the figure to be drawn; choose values for as many unconstrained elements of the figure as there are degrees of freedom, thereby fully constraining the figure; and then to pass this new problem to a construction routine. It is to this last

---

[11] Price, "Satisfying Geometric Constraints," p.11.

step that work on the straight-edge and compass construction problem is most directly related.

I.2 The Domain

The problem domain is a subset of the class of straight-edge and compass constructions. Under the restriction that only a straight-edge and compass be used, the problem is to construct a figure which satisfies a list of constraints. The type of figure dealt with here has been limited to that of triangles and circles; however, the majority of construction problems concern them. Although construction problems range from the very difficult (or even the impossible trisection of an angle) to the quite simple , the problems used as examples are taken from high school geometry texts. The problems may specify angles, sides, radii, medians, altitudes, tangential circles, tangential lines, and points on the circumference of circles. All problems are assumed not to be contradictory, and to have at least one solution.

I.3 The Pattern Of Two Loci

There is a schema in which many straight-edge and compass construction problem solutions may be classified. Geometric objects are specifiable in terms of key points in them: a triangle by its three vertex points, a line by two points through which it passes, and a circle by its center and a point of its circumference. The problem of constructing a figure can thus be considered to be that of locating its essential points. With the tools at hand, straight lines and circles may be drawn; sets or loci of points are thereby created. The intersection set of two such straight-edge or compass loci is easily determined diagramatically, and if the loci are distinct, has at most two members. Thus, through the construction of loci of the two available types, points can be located.

The statement of a straight-edge and compass construction problem is, in its essential nature, a list of conditions or restrictions which must be met by the figure. These conditions can be reformulated into constraints on the points of the figure. The set of points satisfying a constraint can be used in the construction of the solution figure if that set forms either a circular or rectilinear locus. This approach to the straight-edge and compass construction problem is summarized by Polya:

> "First, reduce the problem to the construction of ONE point. Then, split the condition into TWO parts so that each part yields a locus for the unknown point; each locus must be either a straight line or a circle."[12]

He refers to it as the 'pattern of two loci'.

Consider as an example the problem, "Construct a triangle

---

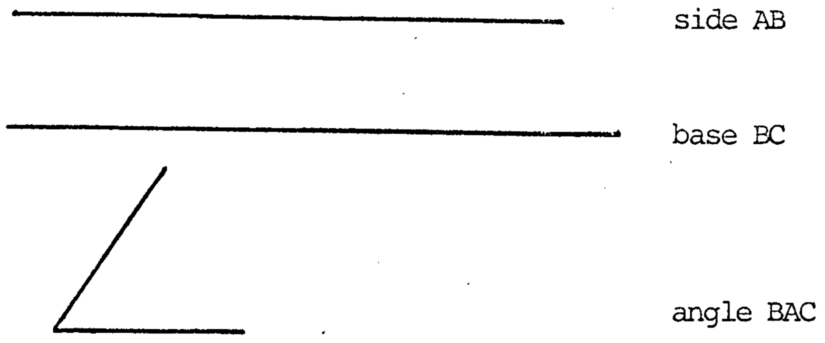
[12] Polya, Mathematical Discovery, Vol. 1 p.5.

given the base, the vertex angle and a side." If we name the triangle ABC, the base BC, the angle BAC, and the side AB, after placing the side AB, point C can be determined using the 'pattern of two loci'. Point C is the intersection of the circle center B radius BC, and the line through point A at angle BAC to BC. This is sketched in figure I. The solution obtained by the CONSTRUCTOR is given below. Included is a dump of the assertions in the database as they were generated from the English statement of the problem.[13] The solution algorithm is the same as the one described above, except it is more explicit about placing the first side.

---

[13] The assertions are discussed in detail in sections II.3 and II.4.

Given:



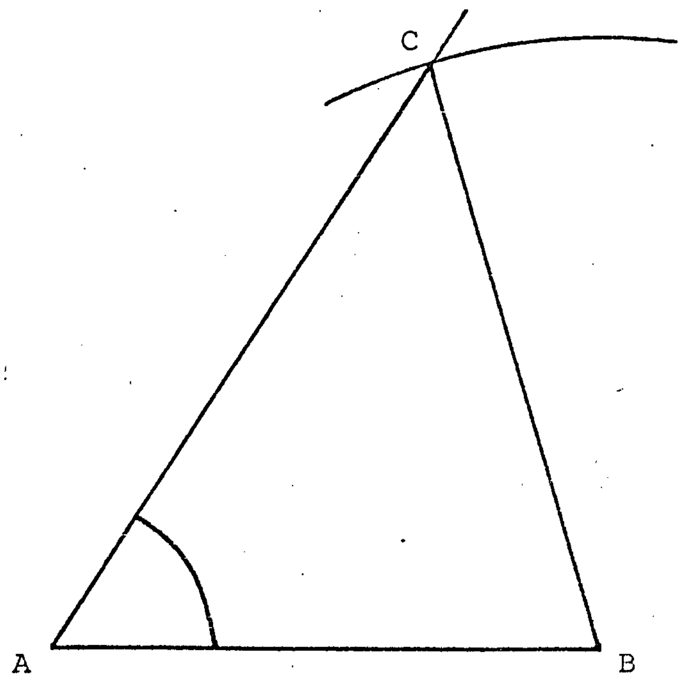


FIGURE I

<u>Example:</u>
(construct a triangle given the base ,
the vertex angle and a side)

```
(CONSTRUCTING TRIANGLE A B C)
(AB IS A SEGMENT OF LINE LINE1 , AC OF LINE2 , AND BC OF LINE3)
(BC IS THE BASE)
(ANGLE BAC IS A GIVEN ANGLE)
(AB IS A GIVEN SIDE)
((LINE LINE3))
((LINE LINE2))
((LINE LINE1))
((TRIANGLE A B C))
((POINT C))
((POINT B))
((POINT A))
((POINT C (CONDITION-ON C ST TRIANGLE A B C)) . 20)
((POINT B (CONDITION-ON B ST TRIANGLE A B C)) . 20)
((POINT A (CONDITION-ON A ST TRIANGLE A B C)) . 20)
((C ON LINE3))
((C ON LINE2))
((B ON LINE3))
((B ON LINE1))
((A ON LINE2))
((A ON LINE1))
((KNOWN LENGTH AB))
((KNOWN ANGLE BAC))
((KNOWN LENGTH BC))

(TRYING CONDITION: (CONDITION-ON A ST TRIANGLE A B C))
(  UNKNOWN POINTS (A B C) NO. OF LOCI FOUND: 0)
(TRYING CONDITION: (CONDITION-ON A ST TRIANGLE A B C))
(  UNKNOWN POINTS (A B C) NO. OF LOCI FOUND: 1)
(TRYING CONDITION: (CONDITION-ON C ST TRIANGLE A B C))
(  UNKNOWN POINTS (C) NO. OF LOCI FOUND: 0)
(TRYING CONDITION: (CONDITION-ON C ST TRIANGLE A B C))
(  UNKNOWN POINTS (C) NO. OF LOCI FOUND: 1)

****beginning of solution algorithm****
((DRAW LINE LINE1 ANYWHERE) (A IS ON LINE LINE1) (PLACE POINT B
ANYWHERE ON LINE1) (POINT A IS ON THE CIRCLE WITH CENTER B RADIUS
AB)(CONSTRUCT LINE LINE2 THRU POINT A (AT ANGLE BAC TO LINE LINE1
 )(C IS ON LINE LINE2) (POINT C IS ON THE CIRCLE WITH CENTER B
RADIUS BC))
```

Although the 'pattern of two loci' is only used here in the solution of construction problems it has a more general nature. Polya points out that if the notion of locus is broadened to a set of arbitrary items satisfying a restriction and the number of sets is two or more, the pattern of finding the solution element in the intersection still proves useful.[14] The generalized pattern can be seen: in the interplay of syntax and semantics in natural language; in the choice of a theorem or axiom for application in a geometry proof from the intersection of those having applicable results with those whose result is true in the diagram; and, of course, in the solution of a set of simultaneous algebraic equations.

[14] Polya, Mathematical Discovery, Vol. 1, pp.133-138.

II The CONSTRUCTOR

II.1 Overview

A framework or theme is established by the 'pattern of two loci' which has been incorporated into a program for the solution of geometry construction problems. The pattern is not in itself a solution to the construction problem, but rather an approach. The word 'pattern' is used by Polya to connote the fact that the solutions to various problems have a similar structure, as do the reasoning processes which lead to them. The 'pattern of two loci' is a skeleton which must be filled out with methods for the discovery of the loci.[15]

One formalism in which the pattern can be expanded upon is that of first-order logic. The pattern is expressed by the existential statement/question

```
(E x y) (C1(x) & C2(x) &  . . . Cn(x) &
         C1(y) & C2(y) &  . . . Cn(y) &
        (circle(x) v line(x)) & (circle(y) v line(y)) &

        intersecting(x,y) & distinct(x,y))
                    where x,y are loci of a point
                    and C1, . . ,Cn are the conditions
                    of the problem relating to that point
```

A proof of the preceding statement can be attempted within a

[15] Polya, Mathematical Discovery, Vol. 1, pp.3-21.

formal system containing axioms expressing the primitive constructions (a line given two points on it, and a circle given its center and radius), and theorems. Included must be theorems about the construction of basic geometric objects (discussed in the section "Basic Constructions"), as well as theorems concerning loci and the conditions they satisfy. When a proof is obtained, a method for the construction of a point in the required figure can be derived from it.

The existential statement/question can be treated directly in PLANNER as a PROG of two variables.

```
(prog (x y)
      (goal (C1 ?x))
      (goal (C2 ?x))
         . . .
      (goal (Cn ?x))
      (goal (C1 ?y))
      (goal (C2 ?y))
         . . .
      (goal (Cn ?y))
      (or (goal (Circle ?x))
          (goal (Line ?x)))
      (or (goal (Circle ?y))
          (goal (Line ?y)))
      (goal (Intersecting ?x ?y))
      (goal (Unequal ?x ?y))     )
```

This expression of the 'pattern of two loci' embodies its essential nature. The pattern is further augmented to make a full system for the solution of construction problems by the establishment of a set of procedures, PLANNER theorems, representing the geometric knowledge of basic constructions and locus theorems. These procedures are activated by the goals in the PROG. The instantiations of the two PROG variables are loci whose intersection is the location of a required point.

The CONSTRUCTOR, a MICRO-PLANNER program, employs procedures for the representation of the geometric knowledge it requires. The utilization of this knowledge is directed by the conditions of the problem linked together by the 'pattern of two loci' in its PROG form. The input to the program is a single English sentence which expresses the problem's conditions. Output is a list of instructions constituting an algorithm for the construction of the figure satisfying the conditions. Since the conditions are first specified in English they must be transformed into a more usable form. This transformation could be made directly into PLANNER goals, which when amalgamated into a PROG, could be passed to the MICRO-PLANNER interpreter for evaluation. However rather than actually creating a PROG it is somewhat easier and more efficient to write a control function which evaluates the goals for the same net effect as a PROG would have, but maintains greater control over the order in which they are evaluated and the effects of backtracking when failure occurs.

The 'pattern of two loci' and the PROG formalization of it are the motivating force behind the formulation of the procedures described in the immediately succeeding sections.

II.2 Consequent Knowledge

In construction problems, lengths, angles, and relations between parts of the figure to be constructed are given. Using this information to immediately begin the construction of the figure is an impossibility. A relation between various parts of the figure cannot be directly represented on a piece of paper. The solution is to use the well established analytic method, that of working backwards from the conclusion of the problem, in this case the desired figure, to its hypothesis, the information given about the figure. The figure is determined when its points have been determined, so we have the following 'backward chaining' series of steps or goals as the basic mode of operation of the constuctor.

(1) Construct the figure

(2) Construct its points

(3) Construct two loci on which a required point lies.

(4) Construct other lines, points, angles etc. Which are needed for the construction of the loci.

. . .

(n) Eventually in the construction of subgoal figures, information given as part of the problem statement will be required.

The language, PLANNER, is particularly effectual in the description of backwards chaining processes, because of its featured pattern directed procedure invocation. Each procedure proclaims its abilities through the use of a pattern; when a subgoal must be satisfied, the listings of procedure patterns are culled for the procedure most appropriate to the purpose. The consequent knowledge of the CONSTRUCTOR is the reformulation of geometry theorems into procedures, called specialists,[16] (THCONSE theorems) able to fulfill goals and their subgoals.

II.2.A The Loci Specialists

The loci specialists constitute a group of procedures each expert in extracting a locus from a condition the diagram must fulfill when complete. Each specialist is adept at testing the database to determine if the locus to which it pertains is usable in satisfying the current condition. Associated with a specialist is a pattern which must match the current condition in order for it to be invoked, implying that only specialists

---

[16] This term is adapted from T. Winograd who discusses semantic specialists.

relevant to the condition expend any effort.

There is a body of theorems concerning loci proven for the student in most geometry textbooks. These theorems relate a locus to a set of conditions as in:

> The locus of the vertex of a right triangle, with a given hypotenuse as the base, is a circle upon the hypotenuse as a diameter.[17]

These theorems are incarnated in the loci specialists.

From some conditions it may be possible to pry more than one locus. Such conditions are not primitive, and must be divided into their component conditions before even a single locus can be obtained. In many cases a condition which the figure must meet, as expressed in the input, will naturally result in a single locus of points. As an example, the condition, point C is equidistant from points A and B, is one which results in point C being on the perpendicular bisector of the line segment AB. Other conditions, point A is a vertex of the triangle ABC, for instance, are compound in nature and can be subdivided. In this case the subdivision is into the restriction: A is distance AB from point B, A is distance AC from C, A is on the line of which AB is a segment, and A is on the line of which AC is a segment. A compound condition is

---

[17] A.M. Welchons, W.R Krickenberger and H.R. Pearson, Plane Geometry, (Boston, Mass.: Ginn, 1958) p. 353.

considered to be satisfied if the conjunction of its component conditions is satisfied. The division of a compound condition into its component conditions is easily implemented in PLANNER by creating one procedure for each component condition. Each procedure is given a pattern matching the original compound condition.

Below is a description of each of the loci specialists in terms of the condition to which it responds, the subgoals it initiates, and the locus it returns if the sub-condtions are met.

<u>C-ALT1</u>

Condition:	the altitude (distance) from point P to line L is D
Subgoals:	is L known or can it be constructed? is D known or can it be constructed?
Locus:	P is on a line parallel to L which is distance D from L

<u>C-DIST1</u>

Condition:	the distance of point P from Q is D
Subgoals:	is the position of Q known or can it be determined? is D known or can it be determined?
Locus:	P is on the circle of radius D center Q

<u>C-EQUIDIST</u>

Condition:	the point P is equidistant from points Q and R
Subgoals:	is the line segment QR known or

equivalently, are points Q and R known or can they be determined?

Locus: P is on the perpendicular bisector of QR

C-MEDIAN1

Condition: point P is the vertex of a triangle from which the median of length M is dropped to the base B

Subgoals: is the length M known or can it be determined?
is the midpoint of B known or can it be determined?

Locus: P is on the circle of radius M, center the midpoint of B

C-TRI2

Condition: point P is the vertex of a triangle PQR (a compound condition)

Subgoals: is the line through P and Q known or can it be constructed?

Locus: P is on that line.

C-TRI3

Condition: point P is the vertex of a triangle PQR

Subgoals: is the line through P and R known or can it be constructed?

Locus: P is on that line.

C-TRI4

Condition: point P is the vertex of a triangle PQR

Subgoal1: what is the locus of points such that the distance of P to Q is PQ

Locus1: the same as the result of subgoal1

Subgoal2:	what is the locus of points such that the distance from P to R is PR?
Locus2:	subgoal2 locus

<u>C-ISOS1</u>

Condition:	point P is the vertex of an isosceles triangle PQR
Subgoal:	what is the locus of points equidistant from Q and R?
Locus:	the result of the subgoal.

<u>C-RTTRI1</u>

Condition:	point P is the vertex of a right triangle PQR.
Subgoal:	is QR a known line segment?
Locus:	P is on the circle having segment QR as a diameter.

<u>C-CIRCLE1</u>

Conditon:	point P is the center of a circle C which is tangent to a curve V.
Subgoals:	is the radius R of C known or can it be determined? is the curve V known or can it be constructed?
Locus1:	if V is a circle, then P is on a circle whose center is that of V and whose radius is that of V plus that of C.
Locus2:	if V is a line, then P is on a line parallel to V the radius of C from V.

<u>C-CIRCLE2</u>

Condition:	point P is the center of a circle C which is tangent to a curve V at a point Q.

Subgoals: is V known or can it be constructed?

Locus1: if V is a circle, then P is on the line which passes through Q and the center of V.

Locus2: if V is a line, then P is on the perpendicular to V at Q.

<u>C-CIRCLE3</u>

Condition: point P is the center of a circle having Q and R as points on its circumference.

Subgoal: what is the locus of points equidistant from Q and R?

Locus: the result of the subgoal.

<u>C-CIRCLE5</u>

Condition: point P is the center of a circle C which is tangent to two lines.

Subgoal: are the two lines known or can they be constructed?

Locus1: if the lines intersect, then P is on the bisector of the angle they form.

Locus2: if the two lines are parallel, then P is on the line parallel to both of them and midway between them.

One thing every locus specialist must check before succeeding, is whether or not exactly the same locus has been produced at some previous time in the determination of a point. The two loci for a point must obviously be distinct.

II.2.B The Line And Slope Specialists

The line specialists are procedures for the construction of (infinite) straight lines. Subgoals set up by the loci specialists often involve the determination of a line, and it is to these requests that the line specialists respond. The slope specialists are expert in finding the slopes of lines, and are called when a line specialist must know a line's slope in order to construct it.

C-LINE2
: a line is known if it passes through a known point and its slope is known or can be determined.

C-LINE1
: a line is known if it passes through two known points

C-SLOPE1
: the slope of a line is known if it is parallel to another line whose slope is known.

C-SLOPE2
: the slope of a line is known if it intersects another known line at a known angle.

C-SEG1
: a line segment is known if its two endpoints are known.

The line and slope specialists use the THUNIQUE feature of MICRO-PLANNER to avoid infinite recursive loops. This means, for example, that before each line specialist begins, it first checks to see if it or any other line specialist is also in the process of determining the same line. The time and space overhead involved is neither unreasonable in relation to other factors, nor is there any other available solution.

Related to the line specialists is an economizing specialist C-LINETEST. Time costs are high for determining lines because, with slopes involved, other line determinations may be required. To avoid fruitless expenditure of effort in the line and slope specialists, C-LINETEST checks the database (no other specialists are invoked) for the presence of a known line or point. If no line or points are known, the line specialists cannot possibly succeed and are not invoked.

II.2.C Angle Specialists

The construction of angles is done by the angle specialists. Usually the requests for the construction of angles originate in the slope specialists. The angle specialists make use of THUNIQUE for the same reasons as the line and slope specialists do.

C-ANGLE1
: an angle is constructable if it is the angle of intersection of two lines which are known or can be constructed.

C-ANGLE2
: an angle is constructable if it is equal to another angle which is known or can be constructed.

C-ANGLE3
an angle is constructable if it is a multiple of another angle which is known or constructable.

C-ANGLE4
an angle is constructable if it is one angle of a triangle and the other two angles of the triangle are known or can be constructed.

II.2.D Assumption Specialists

When a point or a line is not known and is not constructable from the hypothesis of the problem, it may be expedient and permissible to locate it arbitrarily. Two such assumptions, that of one line and one point, are admissible in an individual problem. The rationale for this flexibility lies in the freedom of choice of an origin and axis in a coordinate system.

The decision as to which point and line should be assumed is crucial. Choosing a point which might be determined by other means results in a reduction in the number of conditions contributing toward the problem solution. In order to give as much direction as possible to this decision, it is postponed until the choice which is made will be certain to be of use. When a specialist requires a line or a point in order to succeed, but it is neither known nor constructable in the current context, its position is assumed whenever allowable.

The assumption specialists test the permissiblity of the requisite assumption, and if it passes, assert that the point or line is known.

There are two assumption specialists, one for lines and one for points, embodying the restraints on the situations in which arbitrary determination of positions can be made.

C-DTRMNPOINT

a point P has the greatest liberty whenever no other points or lines are known. It may under those conditions be placed freely without constraint. When P lies on a line and that line is known or constructable, P may be placed anywhere along it. P has no freedom and must not be assumed whenever another point is already known.

C-ASSUMELINE

whenever no lines are known or constructable and no points on the line in question are known, the line may be positioned arbitrarily. If a point on the line is known, then the slope of the line may be chosen; however, the line must pass through the known point.

II.3 Input And Phrase Specialists

A single sentence consisting of several conjoined phrases is accepted by the CONSTRUCTOR. The understanding of English input is not a goal of this work, and thus the structure and vocabulary of sentences understood by the CONSTRUCTOR has remained very restricted. Statements of straight-edge and compass construction problems have many features in common, the essential framework being:

Construct an OBJECT given RESTRICTION1, RESTRICTION2 . . . RESTRICTIONn.

Since most problems concerning triangles and circles in geometry textbooks are stated in this format or are easily amenable to it, the convenience of having a sentence as the medium of communication outweighs the overhead of its interpretation. Example: Construct a right triangle given a leg and the altitude on the hypotenuse.

Translation of the problem statement must result in a database of information for reference by the consequent knowledge specialists. This transformation of the input is accomplished by PLANNER antecedent theorems called phrase specialists containing definitional geometric knowledge. A LISP function operating under the assumption that 'given', 'and', 'construct', and ',' act as phrase delimiters, splits the input sentence into phrases which are then asserted (using the MICRO-PLANNER primitive THASSERT) triggering the execution of appropriate phrase specialists. The interpretation of the input is therefore effectively a pattern matching process. The pattern matching is done when a phrase is matched to the calling pattern of an antecedent specialist, although the facilities available in MICRO-PLANNER for pattern matching purposes are very rudimentary.

To construct a figure both the knowns and the unknowns of the problem must be specified, and this information must be represented in the database. The knowns are the angles, lengths, and relationships introduced as 'givens' in the problem. When an altitude is a 'given', for example, the information conveyed is that a length is known and that a vertex of a triangle is related to its opposite side by that length. It is important to note that when an angle or length is stated as 'given', this is merely a declaration of it as a known in the problem; lengths and angles are rarely actually specified in geoemtry texts by accompanying diagrams. The unknowns are the points, line segments, angles, lines and circles from which the figure is to be composed, but about which nothing has been specified.

The knowledge used to develop the knowns and unknowns in the database is contained in the phrase specialists. A description of some of the key phrase specialists and an example is given below.

<u>A-CTRI</u>

this specialist describes a triangle. It asserts the existence of three points and three lines, and that each point lies on two of the lines. Three compound locus restrictions stating that each of the three points is part of a triangle are also asserted.

<u>A-ALTMED1</u>

this specialist handles both altitudes and medians which are given. Altitudes and medians are specified with respect to a side of a triangle, however, it is the point opposite that side which is the one affected by the restriction. The appropriate point is determined by first finding the side to which it refers in the database. The point and its restriction are then added to the database. A name for the length of the altitude or median is generated and using this name, the length is asserted to be known.

<u>A-SIDE</u>

when a side is given this specialist is invoked. It simply checks the database to find a side of a triangle which is not known and asserts that it is known.

<u>A-CIRCLE</u>

a circle has a center and radius for which names are generated. A name for the circle is also generated. The circle is then asserted to exist, its center to be its center and its radius its radius. The condition that the center point must be the center of the circle is added as a restriction on that point.

<u>A-TAN1</u>

circles are frequently restricted to be tangent to other curves (lines or circles) at given points. This requires the generation of names for a new curve and point, and the assertion of their existence. The point is asserted to be on both the circle and the curve, as well as being known.

Although there are other phrase specialists the above are indicative of their operation. As an example of the result of the phrase specialists consider the construction problem:

Example

```
(construct a triangle given a side , the altitude
to that side and the median to that side)

****beginning of output from input and phrase specialists****


 (CONSTRUCTING TRIANGLE A B C)
 (AB IS A SEGMENT OF LINE LINE1 , AC OF LINE2 , AND BC OF LINE3)
 (AB IS A GIVEN SIDE)
 (LET THE GIVEN ALTITUDE BE OF LENGTH LEN4)
 (LET THE GIVEN MEDIAN BE OF LENGTH LEN5)

****end of output from input and phrase specialists****

****begin assertion base listing ****

 ((MIDPT6 ON LINE1))
 ((MEDIAN C TO AB))
 ((LINE LINE3))
 ((LINE LINE2))
 ((LINE LINE1))
 ((TRIANGLE A B C))
 ((MIDPOINT MIDPT6 OF AB))
 ((POINT C))
 ((POINT B))
 ((POINT A))
 ((POINT C (CONDITION-ON C ST MEDIAN C TO AB IS LEN5)) . -110)
 ((POINT C (CONDITION-ON C ST ALTITUDE C TO LINE1 IS LEN4))
 . 100)
 ((POINT C (CONDITION-ON C ST TRIANGLE A B C)) . 20)
 ((POINT B (CONDITION-ON B ST TRIANGLE A B C)) . 20)
 ((POINT A (CONDITION-ON A ST TRIANGLE A B C)) . 20)
 ((C ON LINE3))
 ((C ON LINE2))
 ((B ON LINE3))
 ((B ON LINE1))
 ((A ON LINE2))
 ((A ON LINE1))
 ((KNOWN LENGTH LEN5))
 ((KNOWN LENGTH LEN4))

****end of assertion base listing****


 ((KNOWN LENGTH AB))
 (TRYING CONDITION: (CONDITION-ON A ST TRIANGLE A B C))
 (  UNKNOWN POINTS (A B C) NO. OF LOCI FOUND: 0)
 (TRYING CONDITION: (CONDITION-ON A ST TRIANGLE A B C))
 (  UNKNOWN POINTS (A B C) NO. OF LOCI FOUND: 1)
 (TRYING CONDITION: (CONDITION-ON C ST MEDIAN C TO AB IS LEN5))
 (  UNKNOWN POINTS (C) NO. OF LOCI FOUND: 0)
 (TRYING CONDITION: (CONDITION-ON C ST MEDIAN C TO AB IS LEN5))
 (  UNKNOWN POINTS (C) NO. OF LOCI FOUND: 1)
 (TRYING CONDITION: (CONDITION-ON C ST ALTITUDE C TO LINE1 IS
 LEN4))
 (  UNKNOWN POINTS (C) NO. OF LOCI FOUND: 1)
```

```
 ((DRAW LINE LINE1 ANYWHERE) (A IS ON LINE LINE1) (PLACE POINT B
ANYWHERE ON LINE1) (POINT A IS ON THE CIRCLE WITH CENTER B RADIUS
AB) (POINT C IS ON CIRCLE OF RADIUS LEN5 WITH CENTER MIDPT6 , THE
MIDPOINT OF AB)(POINT C IS ON A LINE PARALLEL TO LINE1 AT DISTANC
LEN4))
```

II.4 Point Ordering And Assumption

In most construction problems there is more than one point to be located in the determination of the final figure. The question arises as to the order in which the points should be attempted. Certainly the order is important because the loci of some points can only be found with reference to other points. A simple ordering scheme based upon the idea that those points having the least freedom should be looked for first, has been implemented. This leaves the points with the most freedom available for assumption, if necessary. The more restrictions placed upon a point, the higher is the likelihood that there will be at least two loci from which to determine it. The ordering strategy is: first, begin with the search for the loci of the most restricted point; then, if a subgoal of a loci specialist requires a point to be known, it is simply assumed to be known. That is as long as the conditions of the point assumption specialist are met. Any point assumed in this manner will be one with more freedom than the point whose loci are being sought. Never is an effort made to determine a point which is required by a loci specialist, if it is not simply stated to be known in the database, since this would mean invoking the loci specialists for it, thereby negating the top level ordering of points. Allowing the assumption of lines and points on the demand of a loci specialist that will make immediate use of the assumption, gives a natural priority to the choice of the single point and line available in any one construction problem.

Some conditions on points are more restrictive than others, with the effect that a simple count of the number of conditions relevant to the required points will not necessarily reveal the most restricted point. To facilitate the ordering of points according to their degree of restriction a numeric coefficient is assigned to each condition which is asserted (using the THPROP MICRO-PLANNER feature) in conjunction with the condition by the phrase specialists. These coefficients are totalled and the results sorted into decreasing order by a supervisory LISP function that initiates the point determinations.

The restriction coefficients of the various conditions are given below.

triangle	20
equidistant	40
altitude	100
median	-110
right triangle	50
circle	100

The coefficient for the median condition is an exception because the locus of points satisfying it is defined in terms of the midpoint of a line segment. Since a line segment (two points) cannot be assumed , the midpoint of the segment would itself have to be assumed in order to proceed. However, the midpoint is not a good point on which to exercise the freedom of arbitrary location, because other points in the figure are unlikely to be specified with respect to it. The specialist requiring the midpoint to be known is thus a poor one to invoke before the line segment is likely to have been constructed. The low median coefficient ensures that it will not be.

II.5 Solution Abstraction

In PLANNER there is no explicit method or feature to be used in collecting the solution as it is developed during the execution of PLANNER procedures. The onus of providing a suitable formalism for abstracting the solution from the execution sequence rests with the user. In general, this formalism embodies a generalized trace of the goals and procedures involved in forming the successful branch of the search tree. The evolving 'state of the world' as reflected in changes to the database, and a list of comments or instructions interpretable by either man or machine regarding this evolution, together, form the basis of this trace. For example, a change in the state of the 'blocks' world[18] was represented by the erasure of an old location and the assertion of a new location for an object. This was coupled with the concatenation of a new command (chosen from the small set(movehand, grasp, ungrasp)) to a list of commands each describing changes in the 'blocks' world. Interpretation of this list by graphics routines resulted in a pictorial trace of the solution branch.

A similar type of formalization is used in the CONSTRUCTOR. Whenever an angle, line or locus, for example, is determined, instructions for its construction are added to an expanding list. The instructions and the list they form become an algorithm for the construction of the final figure. Since the algorithm is intended for human interpretation the instructions it entails are communicated in English; however, they could just as simply be coded as the names of graphics routines to be called, or in any other machine interpretable form.

---

[18] Terry Winograd, Procedures as a Representation for Data in a Computer Program for Understanding Natural Language, (Cambridge, Mass: Massachusetts Institute of Technology, 1971) Revised Ph.D. Dissertation, MAC TR-84.

Two different types of traces result depending upon the method used to concatenate new instructions to the list. Permanent concatenation, that is concatenation which is not undoable by PLANNER backtracking, results in a trace of all the subresults obtained en route to the solution. Concatenation which is undoable under PLANNER failure returns a list of only those results forming an integral part of the solution.

II.6 Basic Constructions

The output from the CONSTRUCTOR is a list of English constructions representing an algorithm for the construction of the actual figure described in the problem statement. These instructions are given in terms of constructions which are considered to be basic although not primitive. The primitive constructions are simply those of drawing a line given two points on it and, of drawing a circle given its center and radius. A basic construction is either a primitive construction or one such as: draw a line parallel to a given line at a given distance. This is a simple construction, although it does require definition in terms of the two primitive constructions. The justification for the use of a set of basic constructions is that algorithmic directions are given for their construction in most geometry texts. A list of the basic constructions is given in Appendix I.

II.7 Canonical Naming

As in geometry theorem proving systems, the problem of uniquely identifying geometric objects arises here. Without a naming system, redundancy in the definition and execution of the specialists would be inevitable. In the BTP of Goldstein,[19] names were assigned on the basis of the coordinates of the points involved in an object, as they were determined from the diagram the theorem prover used. Since the CONSTRUCTOR does not use or ever actually draw a diagram, a canonical naming based upon lexicographic ordering is employed. Each point is given a name from the letters of the alphabet and these are used to form the names of other objects.

Letting x,y,z be variables over the set of point names {A,B,C...} and letting >, < be relations of alphabetical order, the naming system is as follows:

(a) xy is a canonical line segment name if x and y are the endpoints of the segment, and x < y

(b) xyz is a canonical angle name if y is the vertex point; x and z are points on the two

[19] Goldstein, Elementary Geometry Theorem Proving, pp. 13-14.

rays which form the angle; and $x < y$.

(c) xyz is a canonical triangle name if $x < y < z$. If a triangle is right angled or isosceles, the distinguished vertex is always chosen to be the first character of the triangle name.

(d) the naming of lines, as distinct from line segments, presents a somewhat different problem. Line names LINE1, LINE2, etc. are generated for lines as they arise.

(e) circle names CIRCLE1, CIRCLE2, etc. are generated as circle references occur in the input text.

II.8 FLOW OF CONTROL

The linkage between the segments of the CONSTRUCTOR is accomplished by a LISP-PLANNER hybrid function. It is based in LISP, but makes explicit calls to the MICRO-PLANNER interpreter for the evaluation of some PLANNER code. The first steps in the execution sequence are those of initialization, and reading the problem statement. This statement is then divided into phrases, as described previously, at which time the MICRO-PLANNER interpreter is called to assert each phrase in turn. The assertion of the phrases results in the invocation of antecedent phrase specialists, fabricators of a model delineating all the knowns and unknowns of the problem. When all the phrases have been asserted, those points essential to the construction of the figure and their associated conditions are looked up and sorted into order in accordance with the restriction count. The MICRO-PLANNER interpreter is then commanded into action again for evaluation of the conditions relevant to the first point. Conditions are expressed as PLANNER goals. When two of these goals have been satisfied the point is asserted to be known, and attention is shifted onto the next point of the ordered sequence. This process continues until all the points are determined. If a situation arises in which no two goals for a point are satisfiable, any assumptions and changes to the database are undone (by MICRO-PLANNER backup) and the loci possibilities of the other points are explored. Return is made to the point in question only after other points have been established. When all points are realized the list of instructions comprising the solution algorithm is printed.

The whole process is explicated by the adjoining flowchart.

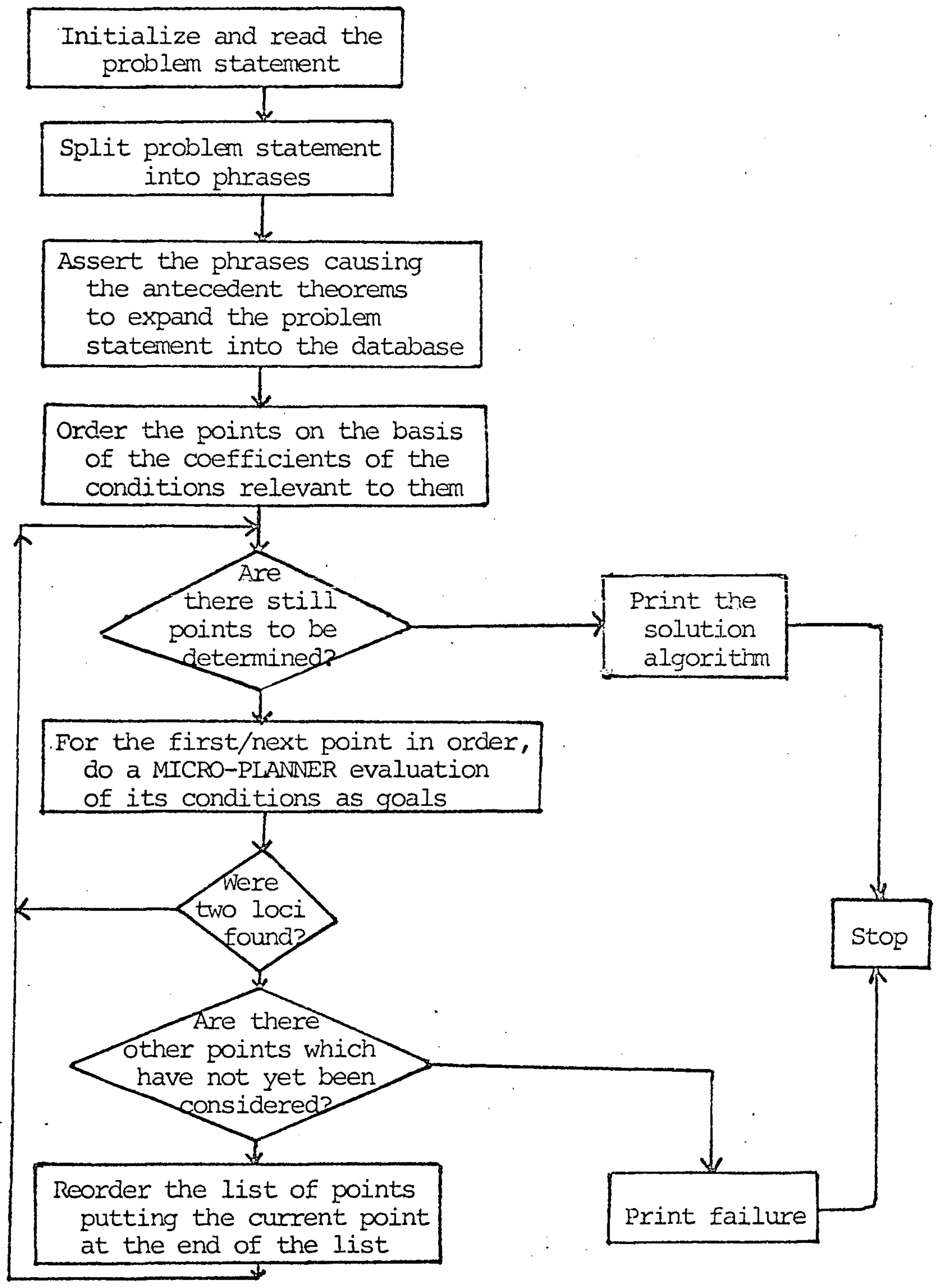

Initialize and read the problem statement
Split problem statement into phrases
Assert the phrases causing the antecedent theorems to expand the problem statement into the database
Order the points on the basis of the coefficients of the conditions relevant to them
Are there still points to be determined?
Print the solution algorithm
For the first/next point in order, do a MICRO-PLANNER evaluation of its conditions as goals
Were two loci found?
Stop
Are there other points which have not yet been considered?
Reorder the list of points putting the current point at the end of the list
Print failure

FIGURE II

III Comments On PLANNER

PLANNER has received simultaneous praise and criticism. Of its three distinguishing features, an associative database, pattern directed procedure invocation, and an automatic backtracking control structure, the latter has been singled out for acute criticism by G. Sussman and D. McDermott.[20] The laudability of PLANNER is illustrated by its instrumental role in the formulation of the 'blocks' world,[21] and the geometry theorem prover, BTP.[22]

Backtracking is a single term for a process which can be seen to consist of two distinct operations. The first operation in the backtrack process is the return to a preceding state in the computation, a decision point, an earlier node in the search tree. The effect of all computation occurring after that node was passed, is undone; the branch of the search tree developed from that node is pruned from the tree. This operation is the heart of backtracking; however, the initiation of computation on

---

[20] G.J. Sussman and D.V. McDermott, "From PLANNER to CONNIVER--A Genetic Approach," Fall Joint Computer Conference 1972, pp. 1171-1179.

[21] Winograd, Procedures as a Representation for Data in a Computer Program for Understanding Natural Language.

[22] Goldstein, Elementary Geometry Theorem Proving.

a new branch emanating from the decision point is also often considered to be implied by the term backtracking. This is most certainly the case in reference to the PLANNER automatic backtracking control structure.

Criticism of automatic backtracking is similarily divisible into two components; what it does and what it does not do. What is <u>not</u> done is the saving of any results from the abandoned branch. Not even a record of the hypothesis which led to the branch's exploration is kept. The answer to this criticism seems to lie in the establishment of contexts and operations for their manipulation, as has been done in QA4[23] and CONNIVER.[24]

What is done under the auspices of PLANNER backtracking is the automatic examination of the next possibility at a decision point. There is no control given to the programmer over this restarting of computation at a point where it may very well be unexpected. According to Sussman and McDermott the problem that

> "an unexpected failure will propagate back . . . and compute without our explicit programming of this activity . . . is observed by <u>real</u> users of

---

[23] J.F. Rulifson, J.A. Derksen and R.J. Waldinger, <u>QA4: A Procedural Calculus for Intuitive Reasoning</u>, (Menlo Park, Calif.: Stanford Reasearch Institute, 1973) Artificial Intelligence Center Technical Note 73.

[24] G.J. Sussman and D.V. McDermott, <u>The CONNIVER Reference Manual</u>, (Cambridge, Mass.: Massachusetts Institute of Technology, 1972) AI Memo no. 259.

> MICRO-PLANNER who complain that they cannot understand the behavior of their programs because the flow of control is not explicit in the code."[25]

An essential difficulty lies with the GOAL primitive, the most common and complex form of decision point. It has too much power vested in it for the frequency with which it must be used. The author suggests syphoning some of this power off into a new PLANNER primitive, to be called IS. This new primitive is a natural contraction of the power of GOAL. Much fruitless exploration and backtracking effort can be eliminated by replacing GOAL with the less powerful IS whenever possible.

The problem with the GOAL function is effectively illustrated by an example which Sussman and McDermott use in their criticism of backtracking. The example is:

```
(CONSEQUENT (X Y Z) (?X IN ?Y)
                    (GOAL (?X IN ?Z))
                    (GOAL (?Z IN ?Y)) )
```
[26]

They comment that this theorem

> ". . . if called by,e. g. (GOAL (BLOCK2 IN BOX1)), its only possible actions are either to find a Z between BLOCK2 and BOX1, or to fail. Although which Z is found cannot possibly affect

---

[25] Sussman and McDermott, "From PLANNER to CONNIVER--A Genetic Approach," p.1173.

[26] Sussman and McDermott, "From PLANNER to CONNIVER--A Genetic Approach," p.1171.

> subsequent events, a failure back to the theorem will cause it to look up another Z, succeed, and allow its caller to fail again in exactly the same way!"[27]

The difficulty arises in that the GOAL function operates under the assumption that every distinct path used in satisfying the goal could result in a different instantiation of the GOAL pattern or in a different modification of the database. Thus all possible methods of satisfying the goal must be tried. Often, however, this is not the case; different theorems, or even the same theorem with different data, as in the above example, may satisfy the goal several times without re-instantiation of the pattern or relevant change to the database. Restarting computation a second time, after re-satisfying the goal in such an inconsequential manner, is pointless. The IS function behaves more as a <u>single</u> test of the environment. A query is made as to whether or not the statement represented by the pattern occurs in the database, theorem base, environment. Equivalently, is there a matching datum in the database, or a matching theorem which, when invoked, succeeds? The way in which the pattern statement is found is taken to be of no consequence or significance. The problem in the example is overcome by using (IS (BLOCK2 IN BOX1)) to call the consequent theorem.

---

[27] Sussman and McDermott, "From PLANNER to CONNIVER--A Genetic Approach," p.1173.

The IS function is easily understood when contrasted with the GOAL function. The GOAL function operates in the following way with respect to success and failure. Initially an attempt is made to satisfy the goal by finding a matching datum or invoking matching theorems. If it is satisfied, the goal succeeds and the next statement after the GOAL is evaluated. The important aspect, in relation to backtracking, is that if a failure propagates back from the statement following the GOAL, to the GOAL, the search for a matching datum or successful theorem, which was suspended when the GOAL was previously satisfied, is restarted. In contrast, the IS function passes the failure backwards rather than attempting to succeed a second time. In all other aspects the operation of IS is the same as GOAL.

The use of the IS function solves the problem, arising in the example, of another Z being found even though it could have no possible effect upon the subsequent computation. Rather than restarting the computation on the statement (GOAL (?X IN ?Z)), the failure is passed beyond the (IS (BLOCK2 IN BOX1)) statement. Examination of MICRO-PLANNER traces of the goals and theorems tried by the CONSTRUCTOR revealed the occurrence of excessive backtracking which could be eliminated by using the IS primitive in place of GOAL. The IS function, added by the author as THIS to MICRO-PLANNER, was used slightly more often than THGOAL in the implementation of the CONSTRUCTOR.

The above discussion is not intended as a justification of the imposition of an automatic backtrack control structure upon the user of a language, as is the case in PLANNER. Backtracking has not turned out to be as useful a feature as it might at first have appeared. Rulifson, Derksen, and Waldinger have observed a shift, which they consider to be a "most surprising trend in QA4 programming", away from backtracking.[28] Nonetheless, backtracking may not be an entirely useless tool, provided that sufficient control over it is available. It can be used during the initial formulation of solutions in order to establish the shape of a better formulation, and for the handling of the element of search which exists in all non-ideal algorithms. The distinction must be made between poorly formulated solutions to well understood problems, and well understood attempts to solve obscure and perplexing problems. Backtracking in PLANNER can be accused of encouraging the former; controllable backtracking may enable the latter.

There is no question that the features of an associative database, and pattern directed procedure invocation are invaluable. They are the factors which have contributed to the success PLANNER has enjoyed despite the difficulties encountered

[28] Rulifson, Derksen and Waldinger, QA4: A Procedural Calculus for Intuitive Reasoning, p. 289.

in controlling its automatic backtracking control structure.

IV CONSTRUCTOR Output

The solutions to the examples in previous sections, and those given here were produced by the CONSTRUCTOR. The CONSTRUCTOR code was interpreted by the MICRO-PLANNER interpreter, modified to include the THIS function, which in turn was operating under the LISP interpreter (no LISP compiler is presently available) in the environment of the Michigan Terminal System on an IBM 360/67. For the triangle problems timings ranged from 20 to 35 CPU seconds, while for the circle problems the range was from 10 to 20 CPU seconds. A representative selection of the solutions found by the CONSTRUCTOR is given below.

The CONSTRUCTOR, as well as not solving problems concerning figures which are different than those specified in the section "The Domain", has as its most serious limitation, the fact that it cannot solve any problem requiring the introduction of a new point. Examples of such problems are: construct a triangle given two sides and the median to the third side; and construct a triangle given its three medians. The latter problem is discussed in more detail in the section "Directions for Further Research". Approximately five CPU minutes are required in order for the CONSTRUCTOR to abandon such problems and admit failure.

The following comments may help in the understanding of the output from the CONSTRUCTOR. The output begins with a statement of the names the CONSTRUCTOR has generated for the geometric elements refered to in the problem. Following this there is a list of all the assertions in the database generated by the input and phrase specialists. After the database dump, all conditions which were set up as goals are printed. In addition, the points in the figure which were unknown, and the number of loci which had been found for the first point on the list of unknown points, at the time the goal was attempted, is given. The final part of the output is the algorithm for the construction of the solution figure. It is a list of instructions, each instruction enclosed in a separate set of parentheses.

```
(construct a right triangle given a leg
 and the altitude to the hypotenuse)

 (CONSTRUCTING TRIANGLE A B C)
 (AB IS A SEGMENT OF LINE LINE1 , AC OF LINE2 , AND BC OF LINE3)
 (AB IS A GIVEN SIDE)
 (LET THE GIVEN ALTITUDE BE OF LENGTH LEN4)
 ((ANGLE BAC EQUAL ANGLE NINETY))
 ((LINE LINE3))
 ((LINE LINE2))
 ((LINE LINE1))
 ((TRIANGLE A B C))
 ((RIGHT TRIANGLE A B C))
 ((POINT C))
 ((POINT B))
 ((POINT A))
 ((POINT A (CONDITION-ON A ST ALTITUDE A TO LINE3 IS LEN4))
. 100)
 ((POINT A (CONDITION-ON A ST RIGHT TRIANGLE A B C)) . 50)
 ((POINT C (CONDITION-ON C ST TRIANGLE A B C)) . 20)
 ((POINT B (CONDITION-ON B ST TRIANGLE A B C)) . 20)
 ((POINT A (CONDITION-ON A ST TRIANGLE A B C)) . 20)
 ((C ON LINE3))
 ((C ON LINE2))
 ((B ON LINE3))
 ((B ON LINE1))
 ((A ON LINE2))
 ((A ON LINE1))
 ((KNOWN LENGTH LEN4))
 ((KNOWN LENGTH AB))
 ((KNOWN ANGLE NINETY))
 (TRYING CONDITION: (CONDITION-ON A ST ALTITUDE A TO LINE3 IS
  LEN4))
   (  UNKNOWN POINTS (A B C) NO. OF LOCI FOUND: 0)
 (TRYING CONDITION: (CONDITION-ON A ST ALTITUDE A TO LINE3 IS
  LEN4))
 (  UNKNOWN POINTS (A B C) NO. OF LOCI FOUND: 1)
 (TRYING CONDITION: (CONDITION-ON A ST RIGHT TRIANGLE A B C))
 (  UNKNOWN POINTS (A B C) NO. OF LOCI FOUND: 1)
 (TRYING CONDITION: (CONDITION-ON A ST TRIANGLE A B C))
 (  UNKNOWN POINTS (A B C) NO. OF LOCI FOUND: 1)
 (TRYING CONDITION: (CONDITION-ON C ST TRIANGLE A B C))
 (  UNKNOWN POINTS (C) NO. OF LOCI FOUND: 0)
 (TRYING CONDITION: (CONDITION-ON C ST TRIANGLE A B C))
 (  UNKNOWN POINTS (C) NO. OF LOCI FOUND: 1)
 ((DRAW LINE LINE3 ANYWHERE) (POINT A IS ON A LINE PARALLEL TO
LINE3 AT DISTANCE LEN4) (PLACE POINT B ANYWHERE ON LINE3) (POINT A
IS ON THE CIRCLE WITH CENTER B RADIUS AB) (ANGLE BAC IS EQUAL TO
ANGLE NINETY) (CONSTRUCT LINE LINE2 THRU POINT A (AT ANGLE BAC TO
LINE LINE1)) (C IS ON LINE LINE2) (C IS ON LINE LINE3))
```

(construct a circle tangent to a given line
and tangent to a second intersecting
line at a given point)

```
 (CIRCLE CIRCLE4 IS TANGENT TO LINE LINE6)
 (THE CIRCLE IS TANGENT TO THE LINE LINE7 AT POINT C)

 ((INTERSECTING LINE7 LINE6))
 ((RADIUS CIRCLE4 RADIUS5))
 ((CIRCLE CIRCLE4))
 ((CIRCLE4 TANGENT LINE6))
 ((CIRCLE4 TANGENT LINE7 AT C))
 ((LINE LINE7))
 ((LINE LINE6))
 ((POINT C))
 ((POINT A))
 ((POINT A (CONDITION-ON A CENTER CIRCLE CIRCLE4)) . 100)
 ((C ON CIRCLE4))
 ((C ON LINE7))
 ((B ON CIRCLE4))
 ((B ON LINE6))
 ((A CENTER CIRCLE4))
 ((KNOWN LINE LINE7))
 ((KNOWN POINT C))
 ((KNOWN LINE LINE6))

 (TRYING CONDITION: (CONDITION-ON A CENTER CIRCLE CIRCLE4))
 (  UNKNOWN POINTS (A) NO. OF LOCI FOUND: 0)
 (TRYING CONDITION: (CONDITION-ON A CENTER CIRCLE CIRCLE4))
 (  UNKNOWN POINTS (A) NO. OF LOCI FOUND: 1)


 ((CENTER POINT A IS ON THE BISECTOR OF THE ANGLE FORMED BY LINES
LINE6 AND LINE7) (CENTER POINT A IS ON A LINE PERPENDICULAR TO
LINE7 AT POINT C))
```

(construct an isosceles triangle given a
base angle and the altitude to a leg)

```
 (CONSTRUCTING TRIANGLE A B C)
```

```
(AB IS A SEGMENT OF LINE LINE8 , AC OF LINE9 , AND BC OF LINE10)
(ANGLE ABC IS A GIVEN ANGLE)
(LET THE GIVEN ALTITUDE BE OF LENGTH LEN11)

((ANGLE ABC EQUAL ANGLE ACB))
((ISOSCELES TRIANGLE A B C))
((LENGTH AB EQUAL LENGTH AC))
((LINE LINE10))
((LINE LINE9))
((LINE LINE8))
((TRIANGLE A B C))
((POINT C))
((POINT B))
((POINT A))
((POINT C (CONDITION-ON C ST ALTITUDE C TO LINE8 IS LEN11))
. 100)
((POINT A (CONDITION-ON A ST A EQUIDISTANT B AND C)) . 40)
((POINT C (CONDITION-ON C ST TRIANGLE A B C)) . 20)
((POINT B (CONDITION-ON B ST TRIANGLE A B C)) . 20)
((POINT A (CONDITION-ON A ST TRIANGLE A B C)) . 20)
((C ON LINE10))
((C ON LINE9))
((B ON LINE10))
((B ON LINE8))
((A ON LINE9))
((A ON LINE8))
((KNOWN LENGTH LEN11))
((KNOWN ANGLE ABC))

(TRYING CONDITION: (CONDITION-ON C ST ALTITUDE C TO LINE8 IS
LEN11))
(  UNKNOWN POINTS (C A B) NO. OF LOCI FOUND: 0)
(TRYING CONDITION: (CONDITION-ON C ST ALTITUDE C TO LINE8 IS
LEN11))
(  UNKNOWN POINTS (C A B) NO. OF LOCI FOUND: 1)
(TRYING CONDITION: (CONDITION-ON C ST TRIANGLE A B C))
(  UNKNOWN POINTS (C A B) NO. OF LOCI FOUND: 1)
(TRYING CONDITION: (CONDITION-ON A ST A EQUIDISTANT B AND C))
(  UNKNOWN POINTS (A) NO. OF LOCI FOUND: 0)
(TRYING CONDITION: (CONDITION-ON A ST A EQUIDISTANT B AND C))
(  UNKNOWN POINTS (A) NO. OF LOCI FOUND: 1)
(TRYING CONDITION: (CONDITION-ON A ST TRIANGLE A B C))
(  UNKNOWN POINTS (A) NO. OF LOCI FOUND: 1)


 ((DRAW LINE LINE8 ANYWHERE) (POINT C IS ON A LINE PARALLEL TO
LINE8 AT DISTANCE LEN11) (PLACE POINT B ANYWHERE ON LINE8)
(CONSTRUCT LINE LINE10 THRU POINT B (AT ANGLE ABC TO LINE LINE8))
(C IS ON LINE LINE10) (POINT A IS ON THE PERPENDICULAR BISECTOR
OF BC) (A IS ON LINE LINE8))
```

```
(construct a circle of given radius which
is tangent to a given circle at a given point)

 (LET THE RADIUS BE OF LENGTH RADIUS13)
 (THE CIRCLE IS TANGENT TO THE CIRCLE CIRCLE14 AT POINT B)

 ((RADIUS CIRCLE12 RADIUS13))
 ((CIRCLE CIRCLE14))
 ((CIRCLE CIRCLE12))
 ((CIRCLE12 TANGENT CIRCLE14 AT B))
 ((POINT B))
 ((POINT A))
 ((POINT A (CONDITION-ON A CENTER CIRCLE CIRCLE12)) . 100)
 ((B ON CIRCLE12))
 ((B ON CIRCLE14))
 ((A CENTER CIRCLE12))
 ((KNOWN CIRCLE CIRCLE14))
 ((KNOWN POINT B))
 ((KNOWN LENGTH RADIUS13))

 (TRYING CONDITION: (CONDITION-ON A CENTER CIRCLE CIRCLE12))
 (  UNKNOWN POINTS (A) NO. OF LOCI FOUND: 0)
 (TRYING CONDITION: (CONDITION-ON A CENTER CIRCLE CIRCLE12))
 (  UNKNOWN POINTS (A) NO. OF LOCI FOUND: 1)
 ((CENTER POINT A IS ON A CIRCLE CENTER B WITH RADIUS RADIUS13) (
CENTER POINT A IS ON A LINE THRU POINT B AND THE CENTER OF
 CIRCLE14))



(construct a right triangle given a base angle and the hypotenuse

   (CONSTRUCTING TRIANGLE A B C)
   (AB IS A SEGMENT OF LINE LINE1 , AC OF LINE2 , AND BC OF LINE3
   (ANGLE ABC IS A GIVEN ANGLE)
   (BC IS THE HYPOTENUSE)
   ((ANGLE BAC EQUAL ANGLE NINETY))

   ((LINE LINE3))
   ((LINE LINE2))
   ((LINE LINE1))
   ((TRIANGLE A B C))
   ((RIGHT TRIANGLE A B C))
   ((POINT C))
   ((POINT B))
```

```
((POINT A))
((POINT A (CONDITION-ON A ST RIGHT TRIANGLE A B C)) . 50)
((POINT C (CONDITION-ON C ST TRIANGLE A B C)) . 20)
((POINT B (CONDITION-ON B ST TRIANGLE A B C)) . 20)
((POINT A (CONDITION-ON A ST TRIANGLE A B C)) . 20)
((C ON LINE3))
((C ON LINE2))
((B ON LINE3))
((B ON LINE1))
((A ON LINE2))
((A ON LINE1))
((KNOWN LENGTH BC))
((KNOWN ANGLE ABC))
((KNOWN ANGLE NINETY))

(TRYING CONDITION: (CONDITION-ON A ST RIGHT TRIANGLE A B C))
(  UNKNOWN POINTS (A B C) NO. OF LOCI FOUND: 0)
(TRYING CONDITION: (CONDITION-ON A ST TRIANGLE A B C))
(  UNKNOWN POINTS (A B C) NO. OF LOCI FOUND: 0)
(TRYING CONDITION: (CONDITION-ON A ST RIGHT TRIANGLE A B C))
(  UNKNOWN POINTS (A B C) NO. OF LOCI FOUND: 1)
(TRYING CONDITION: (CONDITION-ON A ST TRIANGLE A B C))
(  UNKNOWN POINTS (A B C) NO. OF LOCI FOUND: 1)
(TRYING CONDITION: (CONDITION-ON A ST RIGHT TRIANGLE A B C))
(  UNKNOWN POINTS (A B C) NO. OF LOCI FOUND: 1)
(TRYING CONDITION: (CONDITION-ON A ST TRIANGLE A B C))
(  UNKNOWN POINTS (A B C) NO. OF LOCI FOUND: 1)
(TRYING CONDITION: (CONDITION-ON B ST TRIANGLE A B C))
(  UNKNOWN POINTS (A B C) NO. OF LOCI FOUND: 0)
(TRYING CONDITION: (CONDITION-ON B ST TRIANGLE A B C))
(  UNKNOWN POINTS (A B C) NO. OF LOCI FOUND: 1)
(TRYING CONDITION: (CONDITION-ON A ST RIGHT TRIANGLE A B C))
(  UNKNOWN POINTS (A) NO. OF LOCI FOUND: 0)
(TRYING CONDITION: (CONDITION-ON A ST RIGHT TRIANGLE A B C))
(  UNKNOWN POINTS (A) NO. OF LOCI FOUND: 1)
(TRYING CONDITION: (CONDITION-ON A ST TRIANGLE A B C))
(  UNKNOWN POINTS (A) NO. OF LOCI FOUND: 1)


((DRAW LINE LINE3 ANYWHERE) (B IS ON LINE LINE3) (PLACE POINT C
ANYWHERE ON LINE3) (POINT B IS ON THE CIRCLE WITH CENTER C RADIUS
BC) (DRAW CIRCLE WITH CENTER M AT THE MIDPOINT OF BC RADIUS BM)
(CONSTRUCT LINE LINE1 THRU POINT B (AT ANGLE ABC TO LINE LINE3))
(A IS ON LINE LINE1))
```

V Directions For Further Research

There are several areas in which further development could proceed. The CONSTRUCTOR could easily be expanded to handle a broader class of figures than circles and triangles. Other possible candidates for construction might include: quadrilaterals such as parallelograms, squares, and trapezoids; as well as lines. Other constructions in which the solution figure is composed of several subfigures could also be solved, although a difficulty arises in communicating the problem specification. Some form of input other than a single English sentence would have to be employed.

The repetoire of the current system could be expanded to include problems in three dimensions. Just as two intersecting loci determine solution points in two dimensions so does the intersection of three loci in three dimensions. The loci in three dimensions are surfaces rather than curves, but this would not result in any complication.

The solution figure could be actually drawn if graphics routines were developed for the interpretation of the output algorithm. The specification of different values for the given lengths and angles of the problem would result in different solution figures.

One of the stipulations limiting the type of problem under consideration is that they must not be contradictory and hence unsolvable. The CONSTRUCTOR does not attempt to prove a construction impossible; it simply succeeds with an algorithm or reports failure. One technique which could be used in extending the CONSTRUCTOR to prove the impossibility of a construction is analagous to the 'pattern of two loci'. If a point is found to be on two loci, and the loci can be proved to be non-intersecting, then the construction is impossible. Examples of loci which do not intersect are: two parallel lines and also, two concentric circles.

Not all construction problems are solvable using only the 'pattern of two loci'. It is, however, a basic technique, and is required in forming at least part of the solution in all straight-edge and compass constructions. The 'pattern of two loci' can be used in constructing the 'stepping stone' which is refered to in the 'pattern of auxiliary figures'.

> "try to discover some part of the figure or some closely related figure which you can construct and which you can use as a stepping stone in constructing the original figure."[29]

Discovering which auxiliary figure will be of use is in itself a difficult problem for which the author is currently developing techniques. The employment of a diagram would, of course, be imperative when looking for auxiliary figures. An auxiliary

---

[29] Polya, Mathematical Discovery, Vol. 1, p. 15.

figure need not necessarily occur as a part of the diagram of the solution figure, but may be formed by the introduction of new points and lines, in other words a construction.[30] A geometry theorem prover such as that of Goldstein would also have a place as a subpart of a more powerful straight-edge and compass construction program. A proof that an auxiliary figure has particular properties may be required before it can be constructed.

As an example consider the problem, "Construct a triangle given the three medians". Construct a triangle given the three medians An auxiliary figure, constructable by the CONSTRUCTOR, is triangle MGD. A new point D, the midpoint of AM, is introduced into the diagram. It is necessary to prove that GD = MF, and to recall the established result that the medians of a triangle intersect at a point 2/3 the distance from the vertices along the medians. Once a triangle MGD has been constructed, angle GMD is known. With this new information triangle GAM can be constructed by the CONSTRUCTOR, and hence triangle ABC. Both the CONSTRUCTOR and Goldstein's theorem prover could be used as subsystems.

---

[30] The word 'construction' is used here in a theorem proving context and means the introduction of a new element into the problem, not the discovery of a figure.

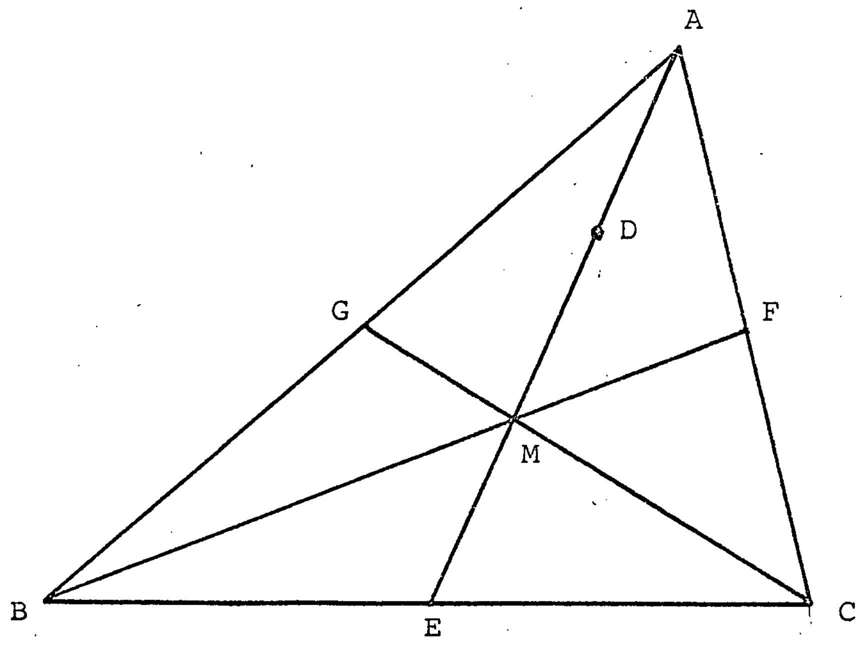


FIGURE III

VI Conclusion

This study has begun to explore the embodiment, manipulation, and extent of the knowledge required for the discovery of figures satisfying a complete and consistent set of constraints. For the delineation and management of the knowledge, procedures have been experimented with as a representation, and the pattern of two loci has been tested as a framework.

A base of knowledge consisting of established results, some of which are common to the theorem provers of Gelernter and Goldstein, was required. There is a difference in tone between the majority of results used in the two tasks however. The theorem provers use results stating that if certain conditions are met then a particular conclusion is justified and valid. The CONSTRUCTOR uses results stating that if certain conditions exist in a partially constructed figure then it is possible to continue the construction by following specific instructions and drawing another locus. The theorem provers use a result to make another step in a proof; the CONSTRUCTOR uses a result to make another step in the construction of a figure.

Encoding the relevant knowledge as programs proved to be both a flexible and adequate approach. It is possible to maintain a clarity of mind when expressing theorems as procedures because they are written in the first person. The programmer dons the cloak of a theorem and then describes its behaviour as his own. This does not imply that all such procedures must be considered to be independent. PLANNER encourages this assumption, but it is neither necessary nor desirable. A procedure should and could be able to behave in accordance with the failures and accomplishments of other procedures. It is natural to first make the independence assumption, write procedures, and observe their performance; then discarding the assumption, interactions between procedures can be added on the basis of their observed behaviour. Regardless of the shortcomings of PLANNER in this respect, the ability to formulate geometric concepts as procedures proved invaluable.

It would be reasonable to ask why a diagram was never used as an aid in solving construction problems as it was by Gelernter and Goldstein in proving theorems. Originally it was thought that a diagram would be essential; however, in the application of the 'pattern of two loci' a diagram simply is not necessary. All the information contained in the problem statement can be represented directly in the database. There is definitely a place for diagrams in the solution of construction problems, but it is associated more with the theorem proving aspects of the problem. Before a locus which might satisfy the restrictions on a point can be drawn, other points, lengths or slopes may have to be found. Such facts may possibly be determined by proving theorems relating them to other known facts. For example, an unknown length may be that of a side of a quadrilateral whose opposite side is known. The unknown length is determined if the quadrilateral can be proven to be a parallelogram. Thus, the mechanism for the determination of these other facts is theorem proving. The sophistication of the machinery available in the CONSTRUCTOR for this is not extensive, and it is in extending it that diagrams would have to be introduced.

An excellent framework is provided by the 'pattern of two loci' for the solution of a large class of straight-edge and compass construction problems of a high school level. In addition, a mastery of the 'pattern of two loci' is prerequisite to solving more difficult problems.

Bibliography

1. Baumgart, Bruce G. Micro-planner Alternate Reference Manual. Palo Alto, Calif.: Stanford University, SAILON no. 67, 1972.

2. Gelernter, H., J.R. Hansen, and D.W. Loveland. "Empirical Explorations of the Geometry-Theorem Proving Machine." Computers and Thought . Ed. Edward A. Feigenbaum and Julian Feldman. New York: McGraw Hill, 1963, pp.153-163.

3. Gelernter, H. "Realization of a Geometry-Theorem Proving Machine." Computers and Thought. Ed. E.A. Feigenbaum and J. Feldman. New York: McGraw Hill, 1963, pp.134-152.

4. Goldstein, Ira. Elementary Geometry Theorem Proving. Cambridge, Mass.: Massachusetts Institute of Technology, AI Memo no. 280, 1973.

5. Hewitt, Carl. Description and Theoretical Analysis (Using Schemata) of Planner: A Language for Proving Theorems and Manipulating Models in a Robot. Cambridge, Mass.: Massachusetts Institute of Technology, Revised Ph.D. Dissertation, AI Technical Report no. 258, 1972.

6. Morgan, F.M. And W.E. Breckenridge. Plane Geometry. Toronto, Ont.: Thomas Nelson & Sons, Rev. Ed., 1954.

7. Polya, George. Mathematical Discovery: On Understanding, Learning and Teaching Problem Solving. New York: John Wiley & Sons, Vol.1, 1962.

8. Price, Keith. "Satisfying Geometric Constraints." Cambridge, Mass.: Massachusetts Institute of Technology, unpublished Bachelor's Paper, 1971.

9. Rulifson, J.F., J.A. Derksen and R.J. Waldinger. QA4: A Procedural Calculus for Intuitive Reasoning. Menlo

Park, Calif.: Stanford Research Institute, Artificial Intelligence Center Technical Note 73, 1972.

10. Sussman, Gerald J. And Drew V. McDermott. The CONNIVER Reference Manual. Cambridge, Mass.: Massachusetts Institute of Technology, AI Memo no. 259, 1972.

11. Sussman, Gerald J. And Drew V. McDermott. "From PLANNER to CONNIVER - A Genetic Approach." Fall Joint Computer Conference 1972, pp. 1171-1179.

12. Sussman, G.J., T. Winograd and E. Charniak. MICRO-PLANNER Reference Manual. Cambridge, Mass.: Massachusetts Institute of Technology, AI Memo no. 203A, 1971.

13. Sussman, G.J. And D.V. McDermott. Why Conniving is Better than Planning. Cambridge, Mass.: Massachusetts Institute of Technology, AI Memo no. 255A, 1972.

14. Welchons, A.M., W.R. Krickenberger and H.R. Pearson. Plane Geometry. Boston, Mass.: Ginn, 1958.

15. Winograd, Terry. Procedures as a Representation for Data in a Computer Program for Understanding Natural Language. Cambridge, Mass.: Massachusetts Institute of Technology, Revised Ph.D. Dissertation, MAC TR - 84, 1971.

16. Wilcox, B. And C. Hafner. LISP/MTS User's Guide. Ann Arbor, Mich.: Mental Health Research Institute, 1973.

17. Wong, Richard. Construction Heuristics for Geometry and a Vector Algebra Representation of Geometry. Cambridge, Mass.: Massachusetts Institute of Technology, Technical Memo no. 28, 1972.

Appendix I

(a) Draw a circle given its center and radius.

(b) Draw a line given two points on it.

(c) Draw an angle equal to another given angle.

(d) Draw a line at a given distance, and parallel to a given line.

(e) Draw the perpendicular to a given line at a given point.

(f) Find the midpoint of a given line segment.

(g) Draw the bisector of a given angle.

(h) Draw the perpendicular bisector to a given line segment.

(i) Draw a line at a given angle to a given line, and at a given point on the it.

Appendix II

Sample Consequent Knowledge Specialists

```
     (thconse c-alt1
; the locus of a point a known distance
;from a known line is a line parallel
; to that line
   ((threstrict a '(lambda (x) (eq $?p x))) p line h)
   (condition-on $?p st altitude $?a to $?line is $?h)
   (this (known length $?h) (thuse c-length2 c-length1))
   (this (determine line $?line) (thuse c-dtrmnline))
   (construction point
     $?a
     is
     on
     a
     line
     parallel
     to
     $?line
     at
     distance
     $?h))


(thconse c-angle1
; an angle is known if it is
;is the angle of intersection of 2 known lines.
   (a e v f)
   (known angle $?a)
   (thunique 'angle $?a)
   (thgoal (explode $?a $?e $?v $?f)
     (thnodb)
     (thuse c-explode))
   (this (known line (thev (line $?e $?v))) (thuse c-linetest))
   (this (known line (thev (line $?f $?v))) (thuse c-linetest))
   (thassert (known angle $?a)))

(thconse c-assumeline
; assuming a line means to determine its position arbitrarily
; a line may be assumed to be known
;if no other line has been assumed

   (l point)
   (assume line $?l)
   (thnot (thgoal (assumed line "?")))
   (thassert (known line $?l))
   (thassert (assumed line $?l))
   (thcond
      ((thand (thgoal ($?point on $?l))
; if a point on the line is known then
 (thgoal (known point $?point)))
  (construction draw line $?l anywhere thru point $?point))
;the line must pass thru it
      ((thnot (thgoal (known point "?")))
  (construction draw line $?l anywhere))
;otherwise it may go anywhere
      (t (thfail theorem))))

(thconse c-dtrmnline
; to determine a line means to first
;see if it is known (in the data base or by deduction)
; and if it is not known to assume that it is known
   (l point)
   (determine line $?l)
   (thor (this (known line $?l) (thuse c-linetest))
   (this (assume line $?l) (thuse c-assumeline)))
   (thfail? t '(thfail theorem)))
;if failure backs up to this point the whole
    ;theorem should fail


(thconse c-line2
; a line is known if it passes thru
;a known point and has known slope.
   (l a s)
   (find line $?l)
   (thcond
      ((thand (thgoal ($?a on $?l))
 (thgoal (known point $?a))))
      ((thand (thgoal ($?a on $?l))
 (this (known point $?a) (thnodb) (thuse c-dtrmnpoint)))))
   (thor (this (known slope $?l $?s) (thuse c-slope2 c-slope1))
   (thfail theorem))
   (thassert (known line $?l))
   (construction construct line $?l thru point $?a $?s))


(thconse c-slope2
; the slope of a line is known if it
; intersects another known line at a known angle.
   (l m a s)
   (known slope $?l $?s)
   (thunique 'slope $?l)
   (thgoal ($?l intersects $?m at angle $?a)
     (thuse c-intersect1))
   (this (known angle $?a) (thuse c-angle2 c-angle3))
   (this (known line $?m) (thuse c-linetest))
   (thsetq $?s ("list" 'at 'angle $?a 'to 'line $?m)))
```

```
(thconse c-circle5
   (c ded
circle
l1
rad
(threstrict l2 '(lambda (teq) (not (eq (thv l1) teq)))))
   (condition-on (thv c) center circle (thv circle))
   (thgoal ((thv circle) tangent (thv l1)))
   (thcond ((thgoal ((thv circle) tangent (thv l2))))
     ((thgoal ((thv circle) tangent (thv l2) at "?"))))
   (thcond
     ((thgoal (intersecting (thv l1) (thv l2))
  (thuse c-intersecting1 c-intersecting2))
  (this (determine line (thv l1)) (thtbf thtrue))
  (this (determine line (thv l2)) (thtbf thtrue))
  (construction center
    point
    (thv c)
    is
    on
    the
    bisector
    of
    the
    angle
    formed
    by
    lines
    (thv l1)
    and
    (thv l2)))
     ((thgoal (parallel (thv l1) (thv l2)) (thtbf thtrue))
  (this (determine line (thv l1)) (thtbf thtrue))
  (this (determine line (thv l2)) (thtbf thtrue))
  (thgoal (radius (thv circle) (thv rad)))
  (thdo
    (thassert
(deduced length
   (thv rad)
   (is one half the separation of the parallel lines)))
    (thassert (known length (thv rad))))
  (construction center
    point
    (thv c)
    is
    on
    a
    line
    parallel
    to
    lines
    (thv l1)
    and
    (thv l2)
    midway
    between
    them))))
```

<u>Sample Input And Phrase Specialists</u>

```
      (thante a-altmed1
   (side len
   segment
   a
   b
   c
   (threstrict ot onto)
   (threstrict factor altmedp)
   (threstrict th '(lambda (th) (memq th '(the that one)))))
   (given the $?factor $?ot $?th $?side)
   (thcond
     ((memq $?side '(base hypotenuse))
  (thgoal (triangle $?a $?b $?c))
  (thsetq $?segment (seg $?b $?c) $?point $?a))
     ((eq $?side 'side)
  (thgoal (triangle $?a $?b $?c))
  (thcond
    ((thgoal (known length (thev (seg $?b $?a))))
 (thsetq $?segment
   (seg $?b $?a)
   $?point
   $?c))
    ((thgoal (known length (thev (seg $?a $?c))))
 (thsetq $?segment
   (thev (seg $?a $?c))
   $?point
   $?b))
    (t
      (thsetq $?segment
  (thev (seg $?b $?c))
  $?point
  $?a))))
     (t (thert bad side - a-altmed1)))
   (thsetq $?len (gensym1 'len))
   (thassert (known length $?len))
   (tprint let the given $?factor be of length $?len)
   (or (eq $?factor 'altitude)
(thassert (median $?point to $?segment) $t))
  (thassert
    (point $?point
     (thev
 (tlist condition-on
  $?point
  st
  $?factor
  $?point
  to
  (cond
    ((eq $?factor 'altitude) (line $?segment))
    (t $?segment))
  is
  $?len)))
    (thprop (cond ((eq $?factor 'altitude) 100) (t -110)))))


(thante a-anyangle
  ((threstrict art articlep) a b c)
  (given $?art angle)
  (thgoal (triangle $?a $?b $?c))
  (thor
    (thand (thassert (known angle (thev (ang $?a $?b $?c))))
     (tprint angle (ang $?a $?b $?c) is a given angle))
    (thand (thassert (known angle (thev (ang $?a $?c $?b))))
     (tprint angle (ang $?a $?c $?b) is a given angle))
    (thand (thassert (known angle (thev (ang $?b $?a $?c))))
     (tprint angle (ang $?b $?a $?c) is a given angle))))


(thante a-isos1
  (a b c)
  (isosceles triangle $?a $?b $?c)
  (thassert
    (point $?a
     (thev
 (tlist condition-on
  $?a
  st
  $?a
  equidistant
  $?b
  and
  $?c)))
    (thprop 40))
  (thassert
    (length (thev (seg $?a $?b))
equal
length
(thev (seg $?a $?c)))))
```

```
(thante a-isos2
  (a b c)
  (isosceles triangle $?a $?b $?c)
  (thassert
    (angle (thev (ang $?a $?b $?c))
     equal
     angle
      (thev (ang $?a $?c $?b)))))
```

Sample LiSP-PLANNER Hybrid Functions

```
(defun construction fexpr (cnarg)
;this function adds a new instruction to the
;solution algorithm after checking that the
;same instruction has not been added previously.
;the instructions may be removed by failure
;backup if necessary
    (prog nil
   (setq cntemp
   (mapcar '(lambda (a)
  (cond ((atom a) a)
 (t (eval a)))) cnarg))
    (cond ((member cntemp construction) (return nil))
   (t (setq thexp (list 'thand
   '(thsetq construction (cons cntemp construction))
   (list 'threturn (list 'quote cntemp))))) )
     (return t) ))
(setq cntemp nil)


(defun line expr cnsegs
     (prog (cnsega cnsegb cnexp)
    (cond ((eq cnsegs 2)
           (setq cnsega (arg 1) cnsegb (arg 2)))
   (t (setq cnexp (explode (arg 1)) cnsega (car cnexp)
     cnsegb (cadr cnexp))))
    (return
    (thval '(thprog (l)
   ($g ($e cnsega on $?l))
   ($g ($e cnsegb on $?l))
   (threturn $?l)) thalist))
 )))))))))
```