Prof. Dr. H. Stoyan
Universität Erlangen-Nürnberg
Institut für Mathematische Maschinen
und Datenverarbeitung (Informatik VIII)
Am Weichselgarten 9
91058 Erlangen

NOTES ON THE GEOMETRY PROBLEM

M. L. Minsky

1. The language.

One of the reasons Plane Geometry might be a rewarding domain for artificial intelligence is that there is a good chance that we could find a language that was simultaneously suitable for machine use and human use. A rather small vocabulary is required, and very little grammar outside that of propositional calculus would be needed. A little bit of lower functional calculus and some special grammatical states or "moods" would be all that need be added. The "moods" are sentences which bear on the various parts of the master organization: they assign to a proposition the status of fact, assumed premise, question to be answered, order ("work on this subgoal: ___") etc. I think that incorporation in the object language of terms which can describe machine orders will cost little and lead to valuable observations, but I would concede that it might be overambitious on a first study. These terms will appear in some form in the pre-assembled program, in any case.

Nouns: "AB", "$\angle$ABC", "P", C(P,r) etc. represent line segments, angles, points, circles, etc. Other terms like "square ABCD" can be a priori or later defined.

Relation terms: Equality: AB = CD (length equality)

$\angle A = \angle B$

Congruence $\cong$

Similarity $\sim$

Parallelism $\|$

Perpendicularity $\perp$ etc.

2, Logic.

Most of elementary geometry can be handled through propositional calculus, or at least most of the formal proofs. The formalization of "construction" may be a little tricky.

There is one particularly interesting domain of problems, namely "locus problems", in which some functional calculus might be useful. The universe of loci could be practically reduced to a finite domain.

Another interesting aspect of the logic is the following. It seems to me that the propositional calculus that seems natural in Euclidean deduction is lopsided in an interesting manner. One uses substitution and detachment in a natural and uncomplicated manner. One uses the logical connectives $\equiv$, $.$, $\sim$, $\rightarrow$, freely. But one does not often use " ". For this reason, it might be rather easy to use a set of logic routines like that proposed by More, where just such an asymmetry between logical "and" and "or" is apparent, at least in the early stages of More's treatment. The same remark applies to the initial Newell-Simon program.

Much of geometry can already be expressed with the above terms, relations and logical connectives. A typical early theorem would be

$$(AB \perp PQ).(CD \perp PQ) \longrightarrow (AB \parallel CD).$$

One has to define "intersection", etc., as a ternary relation; X is the intersection of AB and CD. The property that all radii of a given circle are equal could be expressed without any great difficulty by defining the circle through A with center O by C(O,A) and defining a "radius" as any line OB with OB = OA.

3. Diagrams. I intend to continue development of the program for the diagram-building machine as outlined in my "Exploration Systems" paper. I think the suggestion of McCarthy has great merit: Represent the diagram by analytic devices with full machine precision. Let any uncertainty in the "empirical" measurement routine lie in the method of measurement rather than in the diagram. This has the merit that one can initially write the program without having to worry that a measured equality is due to a mere accident -- by choice of "general" or "arbitrary" parameters, so that there is no simple rational relation between the parameters and their square roots, such a contingency is very unlikely. A prepared table can be used for such decisions, further simplifying the program. This use of full machine precision would also make it

not too often necessary to redraw diagrams. I think that this would still arise often enough to make the diagram-drawing program interesting.

4. Characters and methods.

In this domain I think that it will not be difficult to do a great deal; in particular, it is easy enough to find good sets of characters and methods. Furthermore, their interconnections are sufficiently clear that it would not be surprising if it were feasible to construct the whole character-method machine outlined in my July paper. Even the abstract "character algebra" could be constructed. It would be very exciting if this model turned out to be powerful enough that the machine would no longer need to draw diagrams for simple theorems. It would have learned to do them in its head!

There are two kinds of characters: characters of the logical propositions and characters of the diagrams. To list a few, let a proposition followed by a "?" be a theorem or subgoal to be proven.

Characters:	Methods:
1. Proposition has the form:	
AB = CB?	(i) Prove they are in isosceles triangle. (ii) " " " " congruent triangles.
A = B?	(i) " " " " " " (ii) " " " " similar " (iii) " " " " isosceles " (iv) Opposite angles (v) Corresponding angles (vi) Alternate interior angles, etc.
$A = \frac{1}{2}$ B?	(i) Bisect angle B!

(NOTE: The "!" is an imperative which is an order to the diagram constructor. Naming the resulting angles B_1 and B_2, this method continues with the new subgoals " $A = B_1$?" or " $A = B_2$?")

2. Metric diagram characters.

(In proving two angles equal)

Does one angle measure 30°? ..... Prove. (Subgoal)
45°? ..... " "
90°? ..... Prove right triangle. (Subgoal)

Characters	Methods
Metric diagram characters (cont.)	
$\underline{AB} \overset{*}{=} 1/2\ \underline{BC}$	Prove ABC is 30° rt. triangle. (Subgoal)

(NOTE: The semantical bookkeeping might be simplified by use of special symbol "$\overset{*}{=}$" for empirical equality, to distinguish it from the logical "=".

3. Configurational diagram characters.

Is given angle an exterior angle of some triangle?
Is there another line parallel to given line?
Is there another angle equal to given angle?
Are two lines which we are trying to prove parallel crossed by some transversal?

It seems that there are so many useful configurational characters available that the subgoal must be given a coarse characterization first. Then this character is used to select which other more specific characters to evaluate. E.g., do not try to characterize the whole diagram. If subgoal is concerned with a given angle, only look at local part of figure, and only at characters grouped around angle properties.

4. Special characters and methods.

If problem is to prove a uniqueness, then use reductio ad absurdum logical method.

For locus problem, construct n (approximately five) points on the locus. Then try to group them into one or two second degree algebraic curves. (This could be an independent subroutine.) Find Geometric terms which describe these curves. Now prove that this is same figure described in locus problem.

Prof. Dr. H. Stoyan
Universität Erlangen-Nürnberg
Institut für Mathematische Maschinen
und Datenverarbeitung (Informatik VIII)
Am Weichselgarten 9
91058 Erlangen

NOTES ON THE GEOMETRY PROGRAM (II)

Marvin Minsky
August 16, 1956

Figure Construction.

When we are given a description of a figure in geometric language we try to draw an actual particular figure in such a way as to display the constraints specified in the description,but to make the figure as "general" as possible,otherwise. As humans,we do this by choosing,when choice is possible,parameters of the diagram in such a way that the resulting figure will suggest to us (as humans) as few falsehoods as possible. Thus we try to avoid a final figure in which any angle"looks like" a right angle,if in fact the constraints do not require that the angle be,in fact,a right angle. We do this because our choice of heuristic is suggested by our impression of the drawn figure,and the presence of angles that look to us like right angles have a large effect on the choice of heuristic.

In any case,in constructing a diagram from an abstract description,certain elements of the diagram have no constraints to speak of,others have partial constraints,and others are determined. This hierarchy of constraints does depend on the order in which parts of the diagram are added. In certain situations,the constraints have such a structure that it is quite difficult to draw the figure from the abstract description. I believe that when such situations arise,they reflect real logical aspects of the problem. An example of such a difficulty arises if one tries to draw a diagram of a triangle in which two angle bisectors are equal. McCarthy and I are investigating the question of whether problems in which this type of difficulty arises have a more complicated logical character than those in which it does not arise. It would appear that inequality arguments,monotonicity arguments,and uniqueness arguments are associated with the logical apparatus required to find proofs when

difficulty arises in diagram construction. Presumably, in such cases it will be necessary to extend the logical machinery beyond propositional calculus, probably not very far beyond.

I will herein describe an algorithm for diagram construction which seems to work for "routine" geometric descriptions. The conjecture is that the success of this algorithm in producing a figure consistent with the given description is a sufficient condition that the methods mentioned above will not be required in proof of the theorem. Of course, a great deal depends on what we allow for axioms: there will be some give and take between the logic required for proofs, and the kind of statements we require proofs for.

Consider the following construction: From any point of the base of a triangle, draw parallels to the sides.

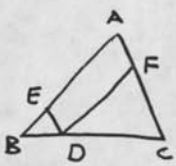
Now I am not contemplating writing a program that will accept the problem in this form, but the next version might be feasible. The required translation routine might not be too difficult to write.

Given: ∠(BAC), choose D on DC, find E on AB with DE||AC, find F on AC with DF||AB.

The following notation is even more machine-like, and the notation I intend to use for a while is introduced here.

Given T°ABC, , [B-D-C], [A-E-B], [DE||AC], [A-F-C], [DF||AB].

DEFINITIONS: "AB" is the line segment AB.



"T°ABC" is the triangle with vertices A, B, C.

"[AB=AC]" says that line segment AB has the same length as line segment AC.

"[B-C-D]" says that B, C, D are collinear in that order, i.e., that C is on the line segment between B and D.

"[DE||AC]" asserts that DE is parallel to AC.

"AB+CD" is the sum of the lengths of AB and CD.

I will define a few more expressions which will be useful in this exposition. No attempt is being made here to write down an exhaustive set of notations for geometry.

"C(P,a)" for "the circle with center P and radius a"
"C(P,AB)" for "the circle with center P and radius length of AB"
"||(A,BC)" for "the line parallel to BC through the point A"
"⊥(A,BC)" for "the perpendicular to BC through the point A"

Now I will describe how the algorithm is used to construct the diagram. First list all terms and then all constraints. Perhaps the simplest way is lexicographical order:

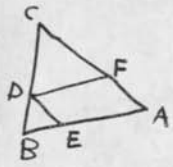
A, B, C, D, E, F, AB,AC,AE,AF,DE,DF, T°ABC,

,[A-E-B],[A-F-C],[B-D-C], [DE||AC], [DF||AB].
(1) (2) (3) (4) (5)

A further convenience will be the replacement of the constraints by the string made up of the letters in each constraint,with an index on each such string which will make it possible to refer back to find what constraint the string came from:

A,B,C,D,E,F,AB,AC,AE,AF,DE,DF,ABC,AEB_1,AFC_2,BDC_3,$DEAC_4$,$DFAB_5$.

-----Now choose the first symbol. This is the point A:



DRAW A

(At this point,the machine will get an instruction to choose a point,with no conditions. It will pick a random pair of coordinates,and assign them to be the position of the point A. We will have more to say about the kind of "random"-ness we want for this kind of unconstrained choice.)

-----Select the next symbol. This is the point B.

-----Look ahead to see if there are any terms which contain B and no letters which have not yet been drawn. (i.e.,contain only B and A.)

-----List these terms: . . . AB.

-----Are any of the above terms from constraints? NO.

DRAW B

(There are no constraints on B. The machine selects coordinates for B which are random except that they are "geometrically independent"of any previously chosen coordinates,namely those for A.)

NOTE: I am not including an explicit discussion of the "generality" routine". The term sequence will first be run through a clean-up routine which handles the non-degeneracy of figures in accord with the wording of the problem. The term ABC in the present problem is really a sort of constraint which should be interpreted as "A,B,and C are not collinear"(which the random coordinate program will take care of automatically,and also "the terms AB,AC,and BC" should appear in the sequence". The latter should be fixed,at this point.) See ** below.

----The non-constrained line segments listed under B(namely just AB) can now be drawn.

DRAW AB

(The machine doesn't have to make any choice here since once the endpoints are determined,so is the segment. It should,however,store under the address corresponding to AB,data about the line,e.g.,its slope,etc.,and its endpoints.)

----Choose the next symbol. This is C.

-----Look ahead for terms involving C and only A and/or B. These are: AC,ABC. ABC can be regarded as a constraint that C not lie on the line through A and B,and also as an instruction to add AB,AC,and BC to the list if they are not there already. **The clean-up routine mentioned above may not really be necessary if this interpretation of the names of figures is used,since the necessary additions to the list of term. occur just when those terms can be interpreted in the drawing.

DRAW C

(Coordinates random except that they are non-collinear with AB.)

DRAW AC, DRAW BC.

----Choose next symbol. This is D.

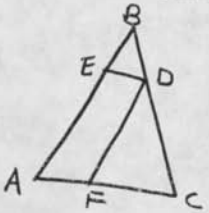
-----Look for all terms involving D and only A,B,and/or C:

BDC_3

-----Are any of these terms from constraints? YES:

BCD_3 represents the constraint [B-C-D]

DRAW D such that [P-D-C]



(The machine chooses a pair of coordinates for D which satisfy the equation for the line through B and C,is between those points,and is otherwise random.)

Remark: Whenever a point is determined consistent with the constraint [X-Y-Z],then the two new segments XY and YZ should be added to the lexicon of geometric objects.

DRAW BD
DRAW DC

-----Choose next symbol. This is E. List all terms which contain E and points already drawn.

$AE, DE, AEB_1, DEAC_4$

-----Are any of these terms from constraints? YES:

AEB_1 and $DEAC_4$

DRAW E such that [A-E-B] AND [DE||AC] (1)

(The machine has to find the coordinates of the point which satisfies the two constraints. The constraints can be converted into equations by the following definitions: Replace the instruction (1) just above by the expression

[A-E*-B] AND [DE*||AC].

The asterisk denotes the unknown point. By the nature of the algorithm we are describing, the constraint could not have come to the machine's attention unless there is just one unknown in it. The constraints with one asterisk are interpreted as follows:

[AB*||CD] means B ε ||(A,CD)

[A*B||CD] means A ε ||(B,CD) etc.

[AB*=CD] means B ε C(A,CD) etc.

[A-B*-C] means B ε line through A and C, and the inequalities necessary for the order.)

(Two constraints determine a point, hence we can

DRAW E)

Because one of the constraints is the [X-Y-Z] type, we

DRAW EA

DRAW EB

----- Choose next point. This is F. The same kind of analysis as for E yields

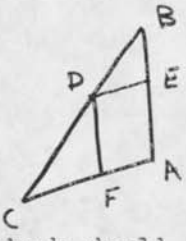
DRAW F such that [A-F-C] AND [DF||AB]

And then

DRAW AF

DRAW FC.

END



(A check reveals no letters to be determined. A check should be made to see that all expressions are now interpreted.)

The process can be abbreviated as follows

	I	II
°A		
°B	AB	
AB		
°C	AC,BC	
AC		
BC		
*D		[B-D*-C]
-BD		
-DC		
**E	AE,DE	[A-E*-B],[DE*||AC]
AE		
DE		
-EB		
**F	AF,DF	[A-F*-C],[DF*||AB]
.AF		
DF		
-FC		
END		

where °X is a point taken in "general position".

*X is a point with one constraint

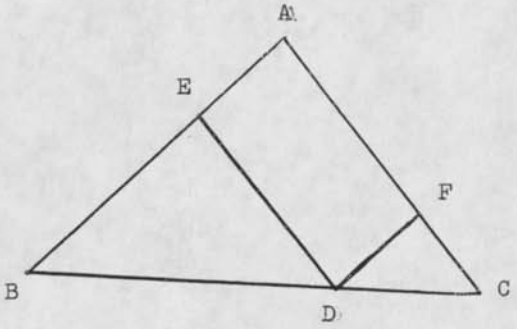
**X is a point determined by two constraints.

XY is a term now determined.

-XY is a new term defined by the construction.

Column I contains terms determined at each step.

Column II contains the constraints which become effective at each step.



The algorithm will work for this problem no matter what order in which the points may be taken. The machine never has to solve any problem more difficult than pairs of linear equations. For other problems, solution of a linear and a "circular" quadratic equation may be required, and in other constructions, pairs of quadratics may be required.

For these computations, I think that complex number representation would be useful, and the associated vector addition properties would simplify the problem of assigning coordinates to a point satisfying a single constraint: if the point lies on a circle, assign a random angle, if on a line, add a random multiple of the unit vector in that direction to some point on that line.

Heuristic note.

This algorithm works on some problems and not on others. If it works, then I am fairly sure that it is a consequence of this fact that certain METHODS can be ruled out in the proof exploration process, e.g., the methods mentioned on page 1.

This suggests that it might be profitable to list several such algorithms for figure construction. Then Each such algorithm can be regarded as a character for problems. The success or failure of a construction algorithm is surely a promising indication of the "character" of a problem. The failure of the algorithm described herein perhaps shows that certain methods are ruled out, and perhaps others are required.