EMPIRICAL EXPLORATIONS OF THE GEOMETRY THEOREM MACHINE *

H. Gelernter, J. R. Hansen, and D. W. Loveland
IBM Research Center
Yorktown Heights, New York

Introduction

In early spring, 1959, an IBM 704 computer, with the assistance of a program comprising some 20 000 individual instructions, proved its first theorem in elementary Euclidean plane geometry [1]. Since that time, the geometry theorem-proving machine (a particular state configuration of the IBM 704 specified by the afore-mentioned machine code) has found solutions to a large number of problems [2] taken from high school textbooks and final examinations in plane geometry. Some of these problems would be considered quite difficult by the average high school student. In fact, it is doubtful whether any but the brightest students could have produced a solution for any of the latter group when granted the same amount of prior "training" afforded the geometry machine (i.e., the same vocabulary of geometric concepts and the same stock of previously proved theorems).

The research project which had as its consequence the geometry theorem-proving machine was motivated by the desire to learn ways to use modern high-speed digital computers for the solution of a new and difficult class of problems; a class heretofore considered to be beyond the capabilities of a finite state automaton. In particular, we wished to make our computer perform tasks which are generally considered to require the intervention of human intelligence and ingenuity for their successful completion. The reasons behind our choice of theorem-proving in geometry as a representative task are set forth in detail in an earlier paper [3]. We only remark here that problem-solving in geometry satisfies our definition of an intellectual activity, while being at the same time especially well suited to the approach we wished to explore. The fact that geometry is decideable is irrelevant for the purpose of our investigation. The methods employed by the machine are suitable as well for the proof of theorems in systems for which no decision algorithm can exist.

*This report is a summary of the full paper to be submitted for publication in an appropriate journal.

We shall not labor the question as to whether our machine is indeed behaving intelligently in performing a task for which humans are credited with intelligence. The psychologists offer us neither aid nor comfort here; they have yet to satisfactorily characterize such behavior in humans, and have rarely considered the abstract concept of intelligence independent of its agent. In the final analysis, people are occasionally observed to do things that may best be described as intelligent, however vague the connotations of the word. These are, in general, tasks involving highly complex decision processes in a potentially infinite and uncontrollable environment. We should be most happy to have our machine duplicate this kind of behavior, whatever label is affixed to it.

Heuristic Programming and the Geometry Machine

The geometry machine is able to discover proofs for a significant number of interesting theorems within the domain of its ad hoc formal system (comprising theorems on parallel lines, congruence, and equality and inequality of segments and angles) without resorting to a decision algorithm or exhaustive enumeration of possible proof sequences. Instead, the theorem-proving program relies upon heuristic methods to restrain it from generating proof sequences that do not have a high a priori probability of leading to a proof for the theorem in question.

The general problem of heuristic programming has been discussed by Minsky [4] and Newell, Shaw, and Simon [5]. The particular approach pursued by the authors has been described at length in the papers to which we have already referred [1, 3]. We shall therefore defer to the presentation of the machine's detailed results in the full paper summarized here for a description of how these results were achieved. It should be recorded here, however, that the geometry machine operates principally in the analytic mode (reasoning backwards). At each stage of the search for a proof, a goal exists which must be "connected" with the premises for the problem by a bridge of axioms and previously established theorems or lemmas. If the connection cannot be made directly, then a set of "subgoals" is generated and the process is repeated

for one of the subgoals. Heuristic rules are used to reject subgoals that are not likely to prove useful, to select one from those remaining to work on, and to choose particular axioms and theorems to use in generating new subgoals. The machine does depart from this procedure in a number of circumstances (in setting up an indirect proof, for example), but these cases account for only a small fraction of the total search time.

The computer program itself was written within the framework of the so-called Newell-Shaw-Simon list memory[6]. In order to ease the task of writing so massive and complex a machine code, a convenient special-purpose list-processing language was designed to be compiled by the already available FORTRAN system for the IBM 704 computer[7]. The authors feel that had they not made free use of an intermediate programming language, it is likely that the geometry program could not have been completed.

Summary of Results

Since its initial solo performance, the geometry machine has existed in several different configurations. In its earliest and most primitive form, the system was equipped with a single major semantic heuristic[8]. That first system was, however, able to prove a large number of interesting, though admittedly simple theorems in elementary plane geometry[9]. The heuristic rule in question, which is independent of the particular formal system under consideration, may be described in the following way. All subgoal formulae that are generated at a given stage of the proof-search are interpreted in a model of the formal system; in our case, the model is a diagram, a formal semantic interpretation. If the interpreted subgoal is valid in the diagram, it is accepted as a possible step in the proof, provided that it is non-circular[10]. Otherwise, it is rejected.

As an experiment, a number of attempts were made to prove extremely simple theorems with the latter heuristic "disconnected" from the system (i.e., all non-circular subgoals generated were accepted). In each case, the computer's entire stock of available storage space was quickly exhausted by the initial several hundreds of first level subgoals generated, and, in fact, the machine never finished generating a complete set of first level subgoals. We estimate conservatively that on the average, a number of the order of 1 000 subgoals are generated per stage by the decoupled system. If one compares the latter figure with the average of 5 subgoals per stage accepted when the diagram is consulted by the machine, it is easy to see that the use of a diagram is crucial for our system. (Note that the total number of subgoals appearing on the problem-solving graph grows exponentially with the number accepted per stage.)

Since the procedure described above is a heuristic one, errors are occasionally made in the selection or rejection of formulae as subgoals. The diagram is made available to the machine in coordinate representation to finite precision. Formulae are interpreted by transforming them into an appropriate calculation on the numerical coordinates representing the point variables. For example, to check the validity of a statement concerning the equality of two segments, the length of each segment in the figure is calculated, and they are then compared to a certain preassigned number of decimal places. If, instead, the statement concerned parallel segments, the slopes would be calculated and compared. In a small number of cases, roundoff error has propagated beyond the allowed value, so that valid subgoals were rejected, or invalid ones accepted. It is important to point out, however, that in no case could this effect result in a false proof. Where valid subgoals were rejected, the machine found alternate paths to the solution. Where invalid ones were accepted, the machine failed, of course, to establish them within the formal system. In the worse possible case, the interpretation error could prevent the computer from finding any solution at all, but never could it lead to an invalid proof.

It should be clear at this point that the diagram is used only to guide the search for a proof by supplying yes or no answers to questions of the form: "Is segment AB equal to segment CD in the figure?", or "Is angle ABC a right angle in the figure?". There is no direct link between the diagram and the formal system in the geometry machine. The behavior of the machine would not be changed if the coordinate representation were replaced by a device that could draw figures on paper and scan them.

In the basic theorem-proving system described above, after a set of subgoals has been generated, each member of the set is explored in order. The next subgoal in line is not examined until the one preceding it has been followed down to a dead end. Too, in generating the next level for a given subgoal, every applicable theorem available is pressed into service.

This system was soon extended by the introduction of selection heuristics for both subgoals and subgoal-generating theorems. The subgoal selection heuristic assigns a "distance" between each subgoal string and the set of

premises in a vaguely-defined ad hoc formula space. At each stage, the next subgoal selected is that which is "closest" to the premises in formula space. The generator selection routine recognizes certain classes of subgoals that are usually established in one step. For such "urgent" subgoals, the appropriate generator is withdrawn immediately, and an attempt is made for a one-step proof (of that particular subgoal) before generating the full set for that formula.

The extended system is able to prove a number of somewhat more difficult theorems that are beyond the capacity of the basic machine[11]. For those problems within the range of both systems, the former is, on the average, about three times faster, and generates about two-thirds the total number of subgoals in half as many subgoal generation cycles as required by the basic system. The average depth of the problem-solving graph for the refined system, about seven to nine levels, is two-thirds the average depth for the basic system.

By the addition of a simple construction routine, the theorem-proving power of the machine is expanded to include an entirely new class of problem, hitherto logically unattainable. The routine, called upon only when all other attempts have failed, allows the machine to join two previously unconnected points in the diagram, and extends the newly-created segment to its intersections with all other segments in the figure. The new segment, when it intersects previously given ones, introduces new points into the problem which are named by the machine and become part of the problem system.

At this stage in its development, the geometry machine was capable of producing proofs that were quite impressive (Appendix I).* Its performance, however, fell off rapidly as the number of points in the diagram increased. This effect was due largely to the fact that unlike humans, who generally identify angles visually by their vertices and rays, the computer specifies an angle by a predicate on three variables, the vertex and a point on each ray. Consequently, the equality of angles 1 and 2 in figure 1 below may be represented in thirty-six different ways, since each angle has six different names. Formal rigor demands, too, that the equality of angles ADH and EDG, for example, be proved rather than taken for granted. It should be clear that where the condition above exists, the search for a proof quickly bogs down in a mass of uninteresting detail.

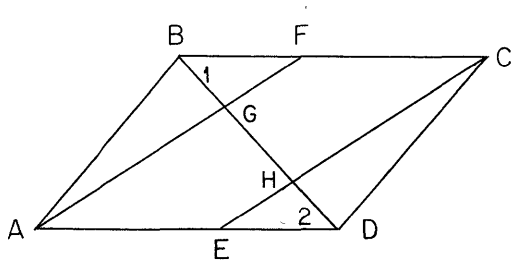


Figure 1

In the current system, the angle problem is solved by allowing the machine to use the diagram to identify a given angle with its full set of names, and to assume the equality relationship between different names for the same angle, as does its human counterpart. The geometry machine in its present configuration is able to find proofs for theorems of the order of difficulty represented by the following:

> THEOREM: If the segment joining the midpoints of the diagonals of a trapezoid is extended to intersect a side of the trapezoid, it bisects that side (Appendix II).

Limitations of the System

It will be immediately evident to those familiar with the properties of formal logistic systems that unless a construction which generates a new point is introduced by the machine, all problems are solved within the framework

---

*In the proofs appended to this paper, the non-obvious predicates have the following interpretations:

OPP-SIDE XYUV	Points X and Y are on opposite sides of the line through points U and V.
SAME-SIDE XYUV	Points X and Y are on the same side of the line through points U and V.
PRECEDES XYZ	Points, X, Y, and Z are collinear in that order
COLLINEAR XYZ	Points X, Y, and Z are collinear.

of a propositional calculus, however complex its structure. Although the machine's present construction routine can and does generate new points, we could not expect our results to be of great interest to logicians until a full set of possible constructions (corresponding to a complete set of existentially quantified axioms) is made available to the system to abet its search for a proof.

An equally serious limitation on the formal generality of the theorem-proving machine is imposed by our method for determining the well-formedness of strings within the logical system. In order to attain the necessary speed and efficiency in processing, well-formed formulae are defined by schema rather than recursively. The kind of statement that can be made in the system is then determined by the schema available to the machine. The practical effect of this loss in generality is to restrict rather severely the freedom with which algebraic statements in geometry may be manipulated.

In addition to the above, there are a number of non-essential bounds on the theorem-proving ability of the machine. These are a consequence of the limited speed and memory capacity of the computer for problems of such highly combinatorial character. Improvements in either of the above will be immediately effective in extending the class of machine-solvable problems in both quantity and difficulty.

Conclusion

The initial goal of our research program in machine intelligence has been attained. If the interrogator were to restrict his probing to the area of theorem-proving in elementary Euclidean plane geometry, our machine could be expected to give an excellent account of itself in competition with a human in Turing's well-known "imitation game"[12]. Of course there are many other problem areas (solving arithmetic problems, for example) where computers have always been able to compete successfully with humans. The significant point is that a knowledgeable interrogator would certainly avoid such areas in his questioning, while he might well (until now, at any rate) introduce a plane geometry problem in a calculated attempt to separate the men from the machines[13]. Although the stage is now set for the argument that any distinct area of human intellectual activity will in the same way succumb to the inexorable logic of electrons, switches, and gates, we defer to our philosopher colleagues for debate on the implications of that contention, at least until the time that computers have been programmed to consider such issues.

There are a number of consequences of our work that are, fortunately, more concrete than that alluded to above. Perhaps the most important are those relating to inferential analysis, a new branch of applied logic first characterized by Wang[14]. Inferential analysis "treats proofs as numerical analysis does calculations", and is expected to "lead to mechanical checks of new mathematical results" and, more important, "lead to proofs of difficult new theorems by machine". It is expected that our techniques for the manipulation and efficient search of problem solving trees and our results concerning syntactic symmetry[10] will prove to be useful tools in pursuing the goals of inferential analysis.

Contributions have been made, too, in the area of techniques for computer implementation of complex information processes. Results pertaining to the design and use of intermediate languages for the specification of list manipulation processes have been reported elsewhere[7]. The latter work indicates clearly the requirements of a digital computer system designed for optimum execution of such list processes. In brief, a list-processing computer should possess hardware facilities for:

1) Generalized indirect addressing; specified in the indirectly addressed instruction to arbitrary depth and in arbitrary order from either the left or the right field of a two-address register,

2) Effective address recovery; making available the terminal content of the address register (the final address in a long and complex indirect address chain, for example) as the address field for a subsequent operation,

3) Field logic; a greatly expanded set of interfield operations within a full register sectioned according to some previously established convention, and

4) List search operations; a list equivalent of the conventional table look-up instruction.

The bulk storage input-output requirements for a list-processing computer are severe, and are not included in the enumeration above. The system design of a digital computer for the manipulation of list structures will be described in detail in a subsequent paper.

Finally, we consider the implications of our work for the basic problem of machine intelligence. The geometry machine, we feel, offers convincing evidence of the power and fruitfulness of heuristic programming for the solution of problems of a certain class by computer. In our experience, the theorem-proving power of the machine has often been extended by the addition of a single heuristic to a degree equivalent to a three to fivefold increase in the speed or storage capacity of the computer.

Our program has proved to be disappointing as a tool for the study of the more elementary trial-and-error types of machine learning, largely because of the rather low rate at which it accumulates experience. It is reasonable to expect, however, that the geometry machine might yet be pressed into service in an investigation of the higher, conceptual types of machine learning, providing that one will someday know how to formulate the problem.

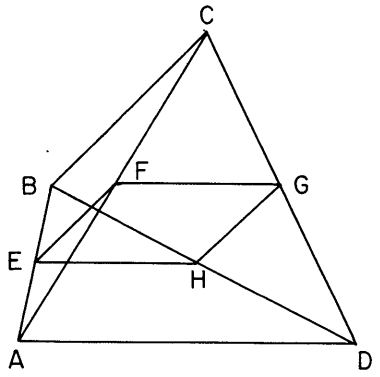
If nothing else, our work offers some qualitative indication of the order of magnitude of difficulty for problems that could be expected to yield to contemporary computer technology. Three years ago, the dominant opinion was that the geometry machine would not exist today. And today, hardly an expert will contest the assertion that machines will be proving interesting theorems in number theory three years hence.

Footnotes and References

1. H. Gelernter, "Realization of a Geometry Theorem-Proving Machine", Proc. of the International Conference on Information Processing, Paris, 1959
2. More than fifty proofs are on file at the present time.
3. H. Gelernter and N. Rochester, "Intelligent Behavior in Problem-Solving Machines", IBM Journal of Research and Development 2 (1958): 336-345
4. M. L. Minsky, "Some Methods of Artificial Intelligence and Heuristic Programming", Symposium on the Mechanization of Thought Processes, Teddington, 1958
5. A. Newell, J. C. Shaw, and H. A. Simon, "Report on a General Problem-Solving Program", Proc. of the International Conference on Information Processing, Paris, 1959
6. A. Newell and J. C. Shaw, "Programming the Logic Theory Machine", Proc. of the Western Joint Computer Conference, (1957): 230-240
7. H. Gelernter, J. R. Hansen, and C. L. Gerberich, "A FORTRAN-Compiled List Processing Language", Journal of the Association for Computing Machinery 7, April 1960
8. A semantic heuristic is one based on an interpretation of the formal system rather than on the structure of the strings within that system.
9. A number of these proofs are reproduced in reference 1 .
10. H. Gelernter, "A Note on Syntactic Symmetry and the Manipulation of Formal Systems by Machine", Information and Control 2 (1959): 80-89
11. See, for example, Appendix A of reference 1.
12. A. M. Turing, "Computing Machinery and Intelligence", Mind 59, (1950): 433
13. It may be argued (and undoubtedly, it will be argued) that the truly knowledgeable interrogator, cognizant of the decideability of geometry, would certainly avoid this area as well, perhaps preferring the manifestly undecideable parts of the predicate calculus or number theory to effect the distinction between man and machine. We recall here that our methods are independent of the decideability of the formal system, and, in fact, Wang[14] and Gilmore[15] have developed techniques that have produced proofs for theorems in the undecideable area of the predicate calculus.
14. H. Wang, "Toward Mechanical Mathematics", IBM Journal of Research and Development 4, January 1960 : 2-22
15. P. C. Gilmore, "A Proof Method for Quantification Theory", IBM Journal of Research and Development 4, January 1960: 28-35

Appendix I

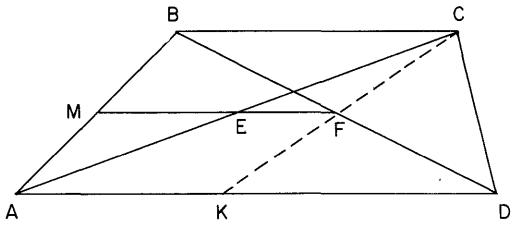
```
   PREMISES
**************
QUAD-LATERAL ABCD
POINT E MIDPOINT SEGMENT AB
POINT F MIDPOINT SEGMENT AC
POINT G MIDPOINT SEGMENT CD
POINT H MIDPOINT SEGMENT BD

   TO PROVE
**************
PARALELOGRAM EFGH

   SYNTACTIC SYMMETRIES
***************************
BA, AB, DC, CD, EE, HF, GG, FH,
CA, DB, AC, BD, GE, FF, EG, HH,
DA, CB, BC, AD, GE, HF, EG, FH,
```




```
   PROOF
***********
SEGMENT DG EQUALS SEGMENT GC
      DEFINITION OF MIDPOINT
SEGMENT CF EQUALS SEGMENT FA
      DEFINITION OF MIDPOINT
TRIANGLE DCA
      ASSUMPTION BASED ON DIAGRAM
PRECEDES DGC
      DEFINITION OF MIDPOINT
PRECEDES CFA
      DEFINITION OF MIDPOINT
SEGMENT GF PARALLEL SEGMENT AD
      SEGMENT JOINING MIDPOINTS OF SIDES OF TRIANGLE IS PARALLEL TO BASE
SEGMENT HE PARALLEL SEGMENT AD
      SYNTACTIC CONJUGATE
SEGMENT GF PARALLEL SEGMENT EH
      SEGMENTS PARALLEL TO THE SAME SEGMENT ARE PARALLEL
SEGMENT HG PARALLEL SEGMENT FE
      SYNTACTIC CONJUGATE
QUAD-LATERAL HGFE
      ASSUMPTION BASED ON DIAGRAM
PARALELOGRAM EFGH
      QUADRILATERAL WITH OPPOSITE SIDES PARALLEL IS A PARALLELOGRAM


   TOTAL ELAPSED TIME   1.03 MINUTES
```

Appendix II

```
   PREMISES
**************
QUAD-LATERAL ABCD
SEGMENT BC PARALLEL SEGMENT AD
POINT E MIDPOINT SEGMENT AC
POINT F MIDPOINT SEGMENT BD
PRECEDES MEF
PRECEDES AMB

   TO PROVE
**************
SEGMENT MB EQUALS SEGMENT MA

   NO SYNTACTIC SYMMETRIES
****************************
```




```
I AM STUCK, ELAPSED TIME   8.12 MINUTES
************************************
CONSTRUCT SEGMENT CF
EXTEND SEGMENT CF TO INTERSECT SEGMENT AD IN POINT K

ADD TO PREMISES THE FOLLOWING STATEMENTS
****************************************
PRECEDES CFK
COLLINEAR AKD


   PROOF
***********
SEGMENT BC PARALLEL SEGMENT AD
      PREMISE
COLLINEAR AKD
      PREMISE
SEGMENT KD PARALLEL SEGMENT BC
      SEGMENTS COLLINEAR WITH PARALLEL SEGMENTS ARE PARALLEL
OPP-SIDE KCDB
      ASSUMPTION BASED ON DIAGRAM
SEGMENT DB
      ASSUMPTION BASED ON DIAGRAM
ANGLE KDB EQUALS ANGLE CBD
      ALTERNATE-INTERIOR ANGLES OF PARALLEL LINES ARE EQUAL
PRECEDES CFK
      PREMISE
PRECEDES DFB
      DEFINITION OF MIDPOINT
ANGLE KFD EQUALS ANGLE CFB
      VERTICAL ANGLES ARE EQUAL
SEGMENT DF EQUALS SEGMENT FB
      DEFINITION OF MIDPOINT
TRIANGLE FDK
      ASSUMPTION BASED ON DIAGRAM
TRIANGLE FBC
      ASSUMPTION BASED ON DIAGRAM
TRIANGLE FDK CONGRUENT TRIANGLE FBC
      TWO TRIANGLES ARE CONGRUENT IF ANGLE-SIDE-ANGLE EQUALS ANGLE-SIDE-ANGLE
SEGMENT KF EQUALS SEGMENT CF
      CORRESPONDING SEGMENTS OF CONGRUENT TRIANGLES ARE EQUAL
SEGMENT CE EQUALS SEGMENT EA
      DEFINITION OF MIDPOINT
TRIANGLE AKC
      ASSUMPTION BASED ON DIAGRAM
PRECEDES CEA
      DEFINITION OF MIDPOINT
SEGMENT EF PARALLEL SEGMENT AK
      SEGMENT JOINING MIDPOINTS OF SIDES OF TRIANGLE IS PARALLEL TO BASE
SEGMENT EF PARALLEL SEGMENT KD
      SEGMENTS COLLINEAR WITH PARALLEL SEGMENTS ARE PARALLEL
SEGMENT FE PARALLEL SEGMENT BC
      SEGMENTS PARALLEL TO THE SAME SEGMENT ARE PARALLEL
PRECEDES MEF
      PREMISE
COLLINEAR MEF
      ORDERED COLLINEAR POINTS ARE COLLINEAR
SEGMENT FM PARALLEL SEGMENT BC
      SEGMENTS COLLINEAR WITH PARALLEL SEGMENTS ARE PARALLEL
SEGMENT FM PARALLEL SEGMENT DA
      SEGMENTS PARALLEL TO THE SAME SEGMENT ARE PARALLEL
TRIANGLE DBA
      ASSUMPTION BASED ON DIAGRAM
PRECEDES AMB
      PREMISE
SEGMENT MB EQUALS SEGMENT MA
      LINE PARALLEL TO BASE OF TRIANGLE BISECTING ONE SIDE BISECTS OTHER SIDE


   TOTAL ELAPSED TIME  30.68 MINUTES
```