MASSACHUSETTS INSTITUTE OF TECHNOLOGY

ARTIFICIAL INTELLIGENCE LABORATORY

AI Working Paper 108 June 1976

Analysis by Propagation of Constraints in Elementary Geometry Problem Solving

by

Jon Doyle

ABSTRACT

This paper describes GEL, a new geometry theorem prover. GEL is the result of an attempt to transfer the problem solving abilities of the EL electronic circuit analysis program of Sussman and Stallman to the domain of geometric diagrams. Like its ancestor, GEL is based on the concepts of "one-step local deductions" and "macro-elements." The performance of this program raises a number of questions about the efficacy of the approach to geometry theorem proving embodied in GEL, and also illustrates problems relating to algebraic simplification in geometric reasoning.

This report describes research done at the Artificial Intelligence Laboratory of the Massachusetts Institute of Technology. Support for the Laboratory's artificial intelligence research is provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-75-C-0643.

Introduction

This paper describes GEL, a new geometry theorem prover. GEL is the result of an attempt to transfer the problem solving abilities of the EL electronic circuit analysis program of Sussman and Stallman [Sussman and Stallman 1975] to the domain of geometric diagrams. Like its ancestor, GEL is based on the concepts of "one-step local deductions" and "macro-elements." These methods, in the form of a modified version of EL, are used to implement a theorem prover with a mechanistic view of geometry. The performance of this program raises a number of questions about the efficacy of the approach to geometry theorem proving embodied in GEL, and also illustrates problems relating to algebraic simplification in geometric reasoning.

One reason elementary plane geometry has been a popular domain for illustrating problem-solving techniques is its semantic richness, its quality of permitting many points of view [Brown 1974, p. 7]. Geometry admits physical (mechanical), algebraic (analytic), Euclidean (synthetic) and turtle drawing (differential) representations. Of these, GEL illustrates the physical, mechanical view of geometry. GEL's analysis proceeds in a fashion analogous to the fixing of certain relationships in a jointed set of mechanical linkages. Given certain parameters of the diagram (certain angles, lengths, etc.), GEL propagates the constraints these parameters induce (the rigid

relationships they determine) throughout the diagram and determines all relationships in the diagram forced by the initial parameters.

Previous approaches to geometry theorem proving have been based on a Euclidean view of geometry, or on a combination of this view and the analytic view afforded by a metrically specified diagram. Gelernter [Gelernter 1963a, 1963b] devised perhaps the best known geometry theorem prover to illustrate the importance of "backward chaining" as a problem solving technique and to explore the detection and use of analogy between problems in solving them by detecting and using "syntactic symmetries" between problems. Goldstein [Goldstein 1973] used the domain of geometry theorem proving to demonstrate the power and simplicity of procedural representations of knowledge, experts, and the PLANNER problem solving formalism [Hewitt 1975]. Nevins [Nevins 1974] introduced a new geometry theorem prover to call attention to the somewhat neglected method of forward chaining, which was almost totally absent in Goldstein's theorem prover, and to observe that the diagram need not be used as a filter for pruning the search if the representation of the problem domain is sufficiently structured and search sufficiently controlled. Finally, Ullman [Ullman 1975] described a geometry theorem prover utilizing both forward and backward chaining. His system derived considerable power from its use of the diagram, not only as a filter, but as a means to structure the entire database in such a way as to automatically constrain and guide the search for a proof.

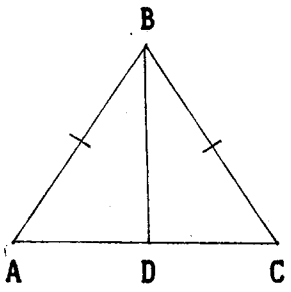
Much of the power of the geometry theorem provers of Nevins and Ullman stems from the use of concepts which are very similar to concepts embodied in EL. Nevins' system is essentially completely based on forward chaining, in that its backward chaining is limited to a form which appears identical to forward chaining. It seem likely that with a modification in the control of forward chaining, his program would not need backward chaining at all, except to handle constructions [de Kleer 1976, p. 11]. In Ullman's system, forward chaining is constrained to be of the form of the one-step local deductions of EL, with this mode of deduction made possible by modelling each element of the diagram by a "reference-frame" similar to the circuit-element structures utilized in EL, data structures with "slots" to represent each of the important aspects of the object being modelled. Ullman's system further employs a "middle-out" search strategy for investigating interesting figures in the diagram which seems analogous to the mechanism of macro-elements in EL.

These similarities in the operation of these programs and Sussman's observation that human geometry problem solving seems to make use of reasoning by constraint propagation to determine that structural relationships exist in the diagram provided the motivation for the current theorem prover. GEL was implemented by replacing EL's electronics knowledge with geometric knowledge. It is almost the case that GEL can do both electronic circuit analysis and geometry theorem

proving - only small technical details and concerns for efficiency of operation have dictated the expunging of electronics knowledge from GEL. This is possible as EL separates its knowledge of its problem solving domain from its problem solving mechanisms. GEL, however, does not make use at present of a number of the problem solving abilities of EL [Note 1].

An Example -- Pons Asinorum

To demonstrate the operation of GEL, we present the example of the Pons Asinorum, the theorem that the base angles of an isoceles triangle are equal. Like previous geometry theorem provers, GEL produces a proof of this theorem by observing that an isoceles triangle is congruent to itself. Unlike previous theorem provers, however, GEL requires the traditional angle bisector, although only as an attention-focusing device. If its attention is properly directed, GEL can prove the theorem without this additional line, but we will present the more complex diagram in order to illustrate some of the problems which can arise. The diagram of the Pons Asinorum is as follows:



Given: AB = BC

<DBA = <CBD

Show: <BAD = <DCB

To prove the theorem, GEL is given the following input:

```
;circuit for the pons asinorum theorem

(WIRE PONS-ASINORUM
      (VERTEX A B C D)
      (CONNECT
       (A (B D))
       (B (C D A))
       (C (D B))
       (D (A B C)))
      (LINE (A D C))
      (TRIANGLE (A B D) (B C D) (A B C))
      (GIVEN (= (SEG A B) (SEG B C))
             (= (ANGLE D B A) (ANGLE C B D)))
      (SHOW  (= (ANGLE B A D) (ANGLE D C B))))
```

Hopefully this method of specifying the diagram, hypotheses and theorem is fairly clear. The diagram is given the name PONS-ASINORUM. Following this is a list of all the vertices of the diagram, and a list of connections between vertices. The connections are specified by naming a vertex and then listing its immediately adjacent vertices in clockwise order. These connections are used to determine all angles and simple line-segments, including the neighbor-sum relationships between the angles at each vertex for use in performing angle additions. Following the connections are lists of all lines (three or

more collinear points, in linear order) and triangles in the diagram. The list of triangles (with vertices in clockwise order) is not really necessary, but was included to simplify the initial programming task.

At this point, the initial diagram wiring is complete. The list of hypotheses is now given, and using these, GEL asserts the initial constraints on the diagram. Each angle determined by a line is given the value of 180 degrees, and for each set of objects listed as equal in a given clause, a value of the form GIVEN82 is generated. This value is specified as a basic value, one which is used strictly as a symbolic value, and each item in the given clause is asserted [Note 2] to have this given value as its value, as in

(= (LENGTH S*AB) GIVEN82).

(All objects of the diagram are given canonical names prefixed with an identifying letter and asterisk. Thus S*AB is the name for the segment AB. For angles, an additional prefix of "*" denotes a reflex angle.)

Finally, the goals of the theorem are given as facts to show. Each such fact expands into a demon which monitors GEL's database for the appearance of the desired facts. In PONS-ASINORUM, the SHOW statement compiles into a demon watching for the assertion of facts of the form

(= (ANGLE A*BAC) value1)) and (= (ANGLE A*BAD) value2)).

Upon completing these steps, GEL begins analyzing the diagram.

The only deductive mechanism used at present in this analysis is the one-step local deduction or antecedent theorem, in which each fact asserted may fill a slot of an object and cause other slots of that object to be filled as a result. In this example, the analysis proceeds as follows until the goal is reached.

```
ANGLE-PROP: F90 (= (ANGLE (A#1 T*BCD)) GIVEN85)
ANGLE-PROP: F91 (= (ANGLE (A#2 T*ABD)) GIVEN85)
LENGTH-PROP: F92 (= (LENGTH (S#1 T*ABC)) GIVEN82)
LENGTH-PROP: F93 (= (LENGTH (S#3 T*BCD)) GIVEN82)
LENGTH-PROP: F94 (= (LENGTH (S#3 T*ABC)) GIVEN82)
LENGTH-PROP: F95 (= (LENGTH (S#3 T*ABD)) GIVEN82)
REFLEX-ANGLE: F96 (= (ANGLE *A*ADC) 180.0)
ADD-ANGLES: F97 (= (ANGLE A*ABC) (&* 2.0 GIVEN85))
ANGLE-PROP: F98 (= (ANGLE (A#2 T*ABC)) (&* 2.0 GIVEN85))
TRI-SAS-21: F99 (= (ANGLE (A#1 T*ABC))
                   (FSAS1 GIVEN82 (&* 2.0 GIVEN85) GIVEN82))
TRI-180: F100 (= (ANGLE (A#3 T*ABC))
                 (&+ 180.0
                     (&* -1.0 (FSAS1 GIVEN82 (&* 2.0 GIVEN85) GIVEN82))
                     (&* -2.0 GIVEN85)))
ANGLE-PROP: F101 (= (ANGLE (A#2 T*BCD))
                    (&+ 180.0
                        (&* -1.0 (FSAS1 GIVEN82 (&* 2.0 GIVEN85) GIVEN82))
                        (&* -2.0 GIVEN85)))
ANGLE-PROP: F102 (= (ANGLE A*BCD)
                    (&+ 180.0
                        (&* -1.0 (FSAS1 GIVEN82 (&* 2.0 GIVEN85) GIVEN82))
                        (&* -2.0 GIVEN85)))
TRI-180: F103 (= (ANGLE (A#3 T*BCD))
                 (&+ (FSAS1 GIVEN82 (&* 2.0 GIVEN85) GIVEN82) GIVEN85))
ANGLE-PROP: F104 (= (ANGLE A*BDC)
                    (&+ (FSAS1 GIVEN82 (&* 2.0 GIVEN85) GIVEN82) GIVEN85))
ANGLE-PROP: F105 (= (ANGLE A*BAD) (FSAS1 GIVEN82 (&* 2.0 GIVEN85) GIVEN82))
ANGLE-PROP: F106 (= (ANGLE (A#1 T*ABD))
                    (FSAS1 GIVEN82 (&* 2.0 GIVEN85) GIVEN82))
TRI-180: F107 (= (ANGLE (A#3 T*ABD))
                 (&+ 180.0
                     (&* -1.0 (FSAS1 GIVEN82 (&* 2.0 GIVEN85) GIVEN82))
                     (&* -1.0 GIVEN85)))
ANGLE-PROP: F108 (= (ANGLE A*ADB)
                    (&+ 180.0
                        (&* -1.0 (FSAS1 GIVEN82 (&* 2.0 GIVEN85) GIVEN82))
                        (&* -1.0 GIVEN85)))
```

```
GOAL FOUND:

F105: (= (ANGLE A*BAD) (FSAS1 GIVEN82 (&* 2.0 GIVEN85) GIVEN82))
F102: (= (ANGLE A*BCD)
         (&+ 180.0
             (&* -1.0 (FSAS1 GIVEN82 (&* 2.0 GIVEN85) GIVEN82))
             (&* -2.0 GIVEN85)))
```

But these expressions (which will be explained shortly) are different! We defer discussion of this anomaly for the moment and instead examine the reasoning behind these calculations by querying GEL for explanations of these facts.

```
(explain 'f105)

F105 (= (ANGLE A*BAD) (FSAS1 GIVEN82 (&* 2.0 GIVEN85) GIVEN82))
                (F99 ANGLE-PROP)
F99 (= (ANGLE (A#1 T*ABC)) (FSAS1 GIVEN82 (&* 2.0 GIVEN85) GIVEN82))
                (F98 F94 F92 TRI-SAS-21)
F98 (= (ANGLE (A#2 T*ABC)) (&* 2.0 GIVEN85)) (F97 ANGLE-PROP)
F97 (= (ANGLE A*ABC) (&* 2.0 GIVEN85)) (F87 F86 ADD-ANGLES)
F94 (= (LENGTH (S#3 T*ABC)) GIVEN82) (F83 LENGTH-PROP)
F92 (= (LENGTH (S#1 T*ABC)) GIVEN82) (F84 LENGTH-PROP)
F87 (= (ANGLE A*CBD) GIVEN85) (GIVEN)
F86 (= (ANGLE A*ABD) GIVEN85) (GIVEN)
F84 (= (LENGTH S*BC) GIVEN82) (GIVEN)
F83 (= (LENGTH S*AB) GIVEN82) (GIVEN)
QED

(explain 'f102)

F102 (= (ANGLE A*BCD)
        (&+ 180.0
            (&* -1.0 (FSAS1 GIVEN82 (&* 2.0 GIVEN85) GIVEN82))
            (&* -2.0 GIVEN85))) (F100 ANGLE-PROP)
F100 (= (ANGLE (A#3 T*ABC))
        (&+ 180.0
            (&* -1.0 (FSAS1 GIVEN82 (&* 2.0 GIVEN85) GIVEN82))
            (&* -2.0 GIVEN85))) (F99 F98 TRI-180)
F99 (= (ANGLE (A#1 T*ABC)) (FSAS1 GIVEN82 (&* 2.0 GIVEN85) GIVEN82))
                (F98 F94 F92 TRI-SAS-21)
F98 (= (ANGLE (A#2 T*ABC)) (&* 2.0 GIVEN85)) (F97 ANGLE-PROP)
F97 (= (ANGLE A*ABC) (&* 2.0 GIVEN85)) (F87 F86 ADD-ANGLES)
F94 (= (LENGTH (S#3 T*ABC)) GIVEN82) (F83 LENGTH-PROP)
F92 (= (LENGTH (S#1 T*ABC)) GIVEN82) (F84 LENGTH-PROP)
F87 (= (ANGLE A*CBD) GIVEN85) (GIVEN)
```

```
F86 (= (ANGLE A*ABD) GIVEN85) (GIVEN)
F84 (= (LENGTH S*BC) GIVEN82) (GIVEN)
F83 (= (LENGTH S*AB) GIVEN82) (GIVEN)
QED
```

The above proofs list each fact of the proof followed by its antecedents and the law by which it was derived. Some of these laws should be recognizable. TRI-180, for instance, is the law which asserts that the sum of the angles of a triangle is 180 degrees. TRI-SAS-21 is a law embodying part of the knowledge that any triangle is determined by two sides and their included angle. The -21 suffix is related to GEL's approach to the identification of parts of triangle and other geometric objects. Each triangle has as parts its first, second and third angles and sides, denoted A#1, A#2, A#3, S#1, S#2, and S#3, where S#i is the side opposite A#i. These parts of the triangle are identified with the appropriate actual angles and sides, so that many triangles may share the same angles and sides. Thus the law TRI-SAS-21 is applicable if A#2 and its adjacent sides S#1 and S#3 are known in a triangle, in which case the law determines the value of one of the remaining parts of the triangle, A#1. There are also the TRI-SAS-22 and TRI-SAS-23 laws. These three laws should really be only one law, but programming details have required their separation in the present implementation. ADD-ANGLES and REFLEX-ANGLE perform the functions of specifying angle sums around vertices and reflex angles if an angle's value is discovered. LENGTH-PROP and ANGLE-PROP are merely propagation laws which serve to spread values assigned to one segment or angle to all segments or angles identified with that segment or angle. Altogether, GEL at present uses a set of laws comprised of the

TRI-SAS, TRI-ASA and TRI-SSS groups, TRI-180, ADD-ANGLES and REFLEX-ANGLE.

The funny functions occurring in the above proofs describe the geometric relations between the measures of parts of triangles, and are really abbreviations for rather messy nonlinear, radical and transcendental formulae. There are four such functions:

(FSAS1 X Y Z) is the measure of the angle opposite side X in a triangle with side X, angle Y and side Z,

(FSAS2 X Y Z) is the measure of the side opposite angle Y in a triangle with side X, angle Y and side Z,

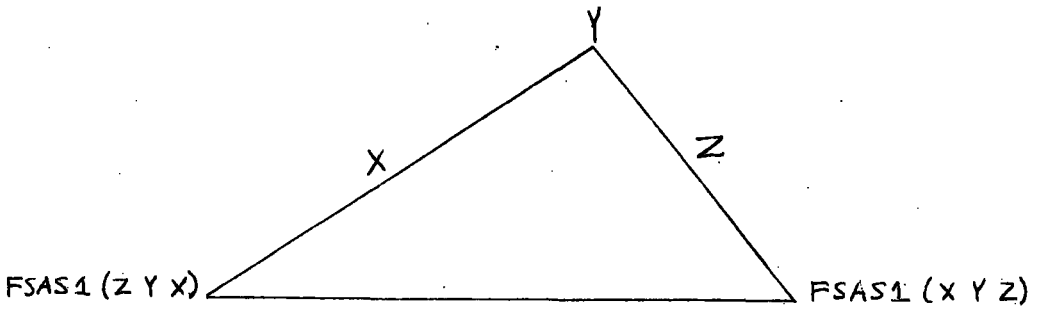
(FASA X Y Z) is the measure of the side opposite angle X in a triangle with angle X, side Y and angle Z, and

(FSSS X Y Z) is the measure of the angle opposite side X in a triangle with sides X, Y and Z.

These functions have peculiar features, such as commutativity in some (but not all) arguments, and many identities involving compositions of these functions. In particular, there are a number of identities relating simple compositions of these functions. For example, one such identity is

180 - (FSAS1 X Y Z) - Y = (FSAS1 Z Y X),

which can be seen to be universally valid by examining the diagram

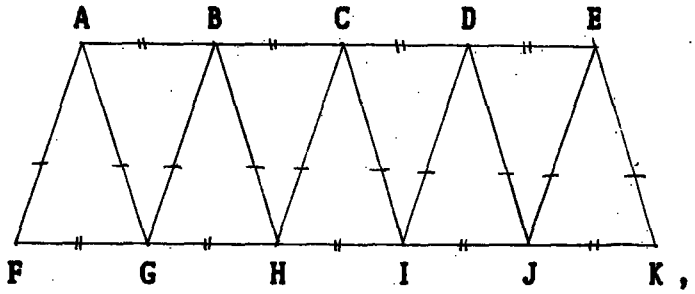


This simplification rule applied to F102 above proves the theorem, for the expression for that angle then simplifies to the same value as that in F105. GEL's inability to prove the theorem completely stems from its use of a fairly unsophisticated pattern-matching system for simplifying these geometric expressions.

Limitations of GEL

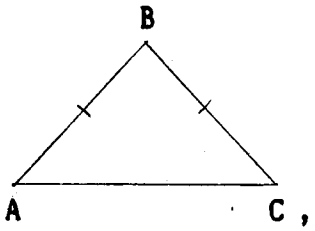
The first (and least important) limitation of this theorem proving system is its limited set of geometric laws. GEL knows nothing about quadrilaterals, parallelograms, or parallel lines. This limitation is not very fundamental, as these capabilities can be added to the program without too much difficulty. In fact, it appears that the function of concepts like parallel lines is simply to allow analysis without as many connecting lines in the diagram; that is, parallel lines seem to serve as macro-elements in the sense of EL.

An important deficiency of GEL is its lack of direction, its reliance on undirected law application for calculating its goals. In large diagrams, this lack allows the theorem prover to calculate almost all values before arriving at the goal values. GEL is easily forced into much useless computation by the introduction of superfluous points and lines into the diagram. Thus in the diagram



GEL would perform much wasted computation before it would determine that $\angle AFG = \angle JKE$, whereas a backward chaining theorem prover would prove this fact easily and quickly.

Another problem arising in GEL is attention focusing. To prove the Pons Asinorum above, GEL required the angle bisector to give it enough information about triangle ABC to be able to determine the values of the base angles. If it had been given the diagram



nothing would have occurred, since there would be only two values known in the triangle, the values of the sides. In this example, the remedy is simple; a value is postulated for one of the unknowns (either <ABC or AC will work) and everything is then solved for in term of this unknown, as illustrated in the following. Giving GEL the wiring diagram for the above diagram, with a value specified for <ABC, GEL produces the following analysis.

```
(wire pons-asinorum
      (vertex a b c)
      (connect
       (a (b c))
       (b (c a))
       (c (a b)))
      (triangle (a b c))
      (given (= (seg a b) (seg b c))
             (= (angle c b a)))
      (show (= (angle b a c) (angle a c b))))
```

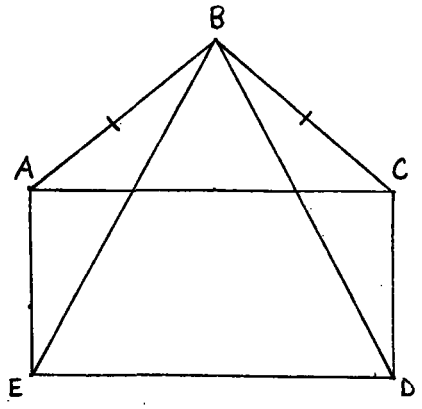
```
ANGLE-PROP: F42 (= (ANGLE (A#2 T*ABC)) GIVEN38)
LENGTH-PROP: F43 (= (LENGTH (S#1 T*ABC)) GIVEN35)
LENGTH-PROP: F44 (= (LENGTH (S#3 T*ABC)) GIVEN35)
TRI-SAS-21: F45 (= (ANGLE (A#1 T*ABC)) (FSAS1 GIVEN35 GIVEN38 GIVEN35))
TRI-180: F46 (= (ANGLE (A#3 T*ABC)) (FSAS1 GIVEN35 GIVEN38 GIVEN35))
ANGLE-PROP: F47 (= (ANGLE A*ACB) (FSAS1 GIVEN35 GIVEN38 GIVEN35))
ANGLE-PROP: F48 (= (ANGLE A*BAC) (FSAS1 GIVEN35 GIVEN38 GIVEN35))

GOAL FOUND:

F48: (= (ANGLE A*BAC) (FSAS1 GIVEN35 GIVEN38 GIVEN35))
```

```
F47: (= (ANGLE A*ACB) (FSAS1 GIVEN35 GIVEN38 GIVEN35))

(explain 'f48)

F48 (= (ANGLE A*BAC) (FSAS1 GIVEN35 GIVEN38 GIVEN35))
                (F45 ANGLE-PROP)
F45 (= (ANGLE (A#1 T*ABC)) (FSAS1 GIVEN35 GIVEN38 GIVEN35))
                (F44 F43 F42 TRI-SAS-21)
F44 (= (LENGTH (S#3 T*ABC)) GIVEN35) (F36 LENGTH-PROP)
F43 (= (LENGTH (S#1 T*ABC)) GIVEN35) (F37 LENGTH-PROP)
F42 (= (ANGLE (A#2 T*ABC)) GIVEN38) (F39 ANGLE-PROP)
F39 (= (ANGLE A*ABC) GIVEN38) (GIVEN)
F37 (= (LENGTH S*BC) GIVEN35) (GIVEN)
F36 (= (LENGTH S*AB) GIVEN35) (GIVEN)
QED

(explain 'f47)

F47 (= (ANGLE A*ACB) (FSAS1 GIVEN35 GIVEN38 GIVEN35))
                (F46 ANGLE-PROP)
F46 (= (ANGLE (A#3 T*ABC)) (FSAS1 GIVEN35 GIVEN38 GIVEN35))
                (F45 F42 TRI-180)
F45 (= (ANGLE (A#1 T*ABC)) (FSAS1 GIVEN35 GIVEN38 GIVEN35))
                (F44 F43 F42 TRI-SAS-21)
F44 (= (LENGTH (S#3 T*ABC)) GIVEN35) (F36 LENGTH-PROP)
F43 (= (LENGTH (S#1 T*ABC)) GIVEN35) (F37 LENGTH-PROP)
F42 (= (ANGLE (A#2 T*ABC)) GIVEN38) (F39 ANGLE-PROP)
F39 (= (ANGLE A*ABC) GIVEN38) (GIVEN)
F37 (= (LENGTH S*BC) GIVEN35) (GIVEN)
F36 (= (LENGTH S*AB) GIVEN35) (GIVEN)
QED
```

While this strategy allows GEL to prove this simple theorem completely, it is not really the best answer, as it seems to be merely a poor substitute for using knowledge of congruences to prove theorems.

The final problem arises in GEL's determining values as expressions. The effect of this method of analysis is one of transforming the problem from the domain of geometric diagrams to the domain of algebraic expressions, and replacing theorems of geometry by simplification rules. It appears that this transformation does not

allow a more succinct set of rules of deduction; evidently the simplifier must understand algebraic simplification (which is in general a notoriously difficult task [Note 3]) as well as the geometric function identities. Since the geometric function identities are really just small geometry theorems involving relations within one or between two triangles, transforming the geometric problem into the functional form seems to require an unnecessary duplication of the diagram analyzer's knowledge in the simplifier. Thus instead of transforming the problem into a form involving fewer and simpler rules of deduction, the value mechanisms of GEL make the problem harder by involving the theorem prover in the problems of algebraic simplification. As an example of the mess this puts GEL in, consider the following theorem:



Given: rectangle ACDE

AB = BC

Show: <BED = <EDB

When given this theorem to prove, GEL announces the following results after considerable effort. The expressions below are simplified to the limits of GEL's ability. Since GEL's simplifier is a crude pattern-matching simplifier, the result is not impressive.

GOAL FOUND:

F163:

```
(= (ANGLE A*BED)
   (&+ (FSSS GIVEN140 GIVEN137 GIVEN140)
       (FSAS1 GIVEN134
              (&+ 90.0 (FSSS GIVEN140 GIVEN137 GIVEN140))
              GIVEN140)))
```

F174:

```
(= (ANGLE A*BDE)
   (&+ 180.0
       (&* -1.0 (FSSS GIVEN140 GIVEN137 GIVEN140))
       (&* -1.0
           (FSAS1 GIVEN137
                  (&+ (FSSS GIVEN140 GIVEN137 GIVEN140)
                      (FSAS1 GIVEN134
                             (&+ 90.0
                                 (FSSS GIVEN140 GIVEN137 GIVEN140))
                             GIVEN140))
                  (FASA (&+ 90.0 (FSSS GIVEN140 GIVEN137 GIVEN140))
                        GIVEN134
                        (&+ 90.0
                            (&* -1.0
                                (FSSS GIVEN140 GIVEN137 GIVEN140))
                            (&* -1.0
                                (FSAS1 GIVEN134
                                       (&+ 90.0
                                           (FSSS GIVEN140
                                                 GIVEN137
                                                 GIVEN140))
                                       GIVEN140)))))))
       (&* -1.0
           (FSAS1 GIVEN134
                  (&+ 90.0 (FSSS GIVEN140 GIVEN137 GIVEN140))
                  GIVEN140))))
```

I will spare you the proofs of these facts.

Actually, the equivalence class mechanisms successfully employed by Nevins and Ullman avoid the problems introduced by GEL's value mechanisms. This method maintains equivalence classes for all

equal angles, line-segments, triangles, etc. When a new pair of objects are found to be equal, their equivalence classes are merged. This approach fits hand in hand with the use of congruences and parallelism in geometry theorem proving, for the primary function of congruences and parallel lines is to assert equality between values without ever having to specify what the values are.

Conclusion

Introspection suggests that human geometry problem solving makes use of constraints which are determined in a manner somewhat analogous to constraining mechanical linkages. While it appears possible to use such information in planning a proof, this method of analysis introduces more problems than it solves in a straightforward implementation of a geometry diagram analyzer.

Acknowledgements

I am indebted to Gerald Jay Sussman, Johan de Kleer, Tomas Lozano-Perez, Kent Stevens, Drew McDermott, Richard Stallman and Scott Fahlman for many helpful discussions, and to the Fannie and John Hertz Foundation for their support of this research.

Notes

1. EL has a number of facilities for using contradictions and coincidences [Sussman and Stallman 1976] which are not used by GEL.

2. GEL uses the EL-ARSE database [Sussman and Stallman 1976, Mason 1976] to store facts and implement laws. This database system is in some ways a descendant of the CONNIVER database system, but has many additional capabilities.

3. Algebraic simplification has long been a stumbling block in symbolic mathematical systems such as MACSYMA [Mathlab 1975, Moses 1971], and seems to arise in many problems in many areas [de Kleer 1975, pp. 78-81].

Bibliography

[Brown 1974]

Allen L. Brown, "Qualitative Knowledge, Casual Reasoning, and the Localization of Failures," MIT-AI Working Paper 61, March 1974.

[de Kleer 1975]

Johan de Kleer, "Qualitative and Quantitative Knowledge in Classical Mechanics," MIT-AI Technical Report 352, December 1975.

[de Kleer 1976]

Johan de Kleer, "Domain Specific Control of Search in a Predicate-Calculus Theorem Prover," informal draft, MIT-AI Lab, April 1976.

[Gelernter 1963a]

H. Gelernter, "Realization of a Geometry-Theorem Proving Machine," in Feigenbaum and Feldman, editors, Computers and Thought, 1963 pp. 134-152.

[Gelernter 1963b]

H. Gelernter, et al., "Empirical Explorations of the Geometry-Theorem Proving Machine," in Feigenbaum and Feldman, Computers and

Thought, 1963, pp. 153-163.

[Goldstein 1973]

Ira P. Goldstein, "Elementary Geometry Theorem Proving," MIT-AI Memo 280, April 1973.

[Hewitt 1975]

Carl E. Hewitt, "How to Use What You Know," in Proc. IJCAI-75, 1975, pp. 189-198.

[Mason 1976]

Matthew T. Mason, "Qualitative Simulation of Swine Production," Bachelors thesis, MIT, Electrical Engineering and Computer Science, May 1976

[Mathlab 1975]

The Mathlab Group, "The MACSYMA Reference Manual," MIT, 1975.

[Moses 1971]

Joel Moses, "Algebraic Simplification: A Guide for the Perplexed," CACM, 1971, pp. 527-537.

[Nevins 1974]

Arthur J. Nevins, "Plane Geometry Theorem Proving Using Forward Chaining," MIT-AI Memo 303, January 1974.

[Sussman and Stallman 1975]

Gerald Jay Sussman and Richard Matthew Stallman, "Heuristic Techniques in Computer Aided Circuit Analysis," MIT-AI Memo 328, March 1975.

[Sussman and Stallman 1976]

Gerald Jay Sussman and Richard Matthew Stallman, "Antecedent Reasoning and Dependency Directed Backtracking in a System for Computer Aided Circuit Analysis," forthcoming MIT-AI report, 1976.

[Ullman 1975]

Shimon Ullman, "A Model-Driven Geometry Theorem Prover," MIT-AI Memo 321, May 1975.

APPENDIX I: The Geometry Laws of GEL

```
;triangle angle sum

(law tri-180 hipri-asap ((t* triangle) a1 a2 a3)
     ()
     ((= (angle (a#1 !?t*)) !>a1)
      (= (angle (a#2 !?t*)) !>a2)
      (= (angle (a#3 !?t*)) !>a3))
     (equation '(&+ a1 a2 a3) 180.0 t*))


;triangle side-angle-side laws

(law tri-sas-11 asap ((t* triangle) s2 a1 s3 s1)
     ((= (length (s#2 !?t*)) !>s2)
      (= (angle  (a#1 !?t*)) !>a1)
      (= (length (s#3 !?t*)) !>s3))
     ((= (length (s#1 !?t*)) !>s1))
     (equation 's1 "(fsas2 ,s2 ,a1 ,s3) t*))

(law tri-sas-12 asap ((t* triangle) s2 a1 s3 a2)
     ((= (length (s#2 !?t*)) !>s2)
      (= (angle  (a#1 !?t*)) !>a1)
      (= (length (s#3 !?t*)) !>s3))
     ((= (angle  (a#2 !?t*)) !>a2))
     (equation 'a2 "(fsas1 ,s2 ,a1 ,s3) t*))

(law tri-sas-13 asap ((t* triangle) s2 a1 s3 a3)
     ((= (length (s#2 !?t*)) !>s2)
      (= (angle  (a#1 !?t*)) !>a1)
      (= (length (s#3 !?t*)) !>s3))
     ((= (angle  (a#3 !?t*)) !>a3))
     (equation 'a3 "(fsas1 ,s3 ,a1 ,s2) t*))

(law tri-sas-21 asap ((t* triangle) s1 a2 s3 a1)
     ((= (length (s#1 !?t*)) !>s1)
      (= (angle  (a#2 !?t*)) !>a2)
      (= (length (s#3 !?t*)) !>s3))
     ((= (angle  (a#1 !?t*)) !>a1))
     (equation 'a1 "(fsas1 ,s1 ,a2 ,s3) t*))
```

```
(law tri-sas-22 asap ((t* triangle) s1 a2 s3 s2)
     ((= (length (s#1 !?t*)) !>s1)
      (= (angle  (a#2 !?t*)) !>a2)
      (= (length (s#3 !?t*)) !>s3))
     ((= (length (s#2 !?t*)) !>s2))
     (equation 's2 "(fsas2 ,s1 ,a2 ,s3) t*))

(law tri-sas-23 asap ((t* triangle) s1 a2 s3 a3)
     ((= (length (s#1 !?t*)) !>s1)
      (= (angle  (a#2 !?t*)) !>a2)
      (= (length (s#3 !?t*)) !>s3))
     ((= (angle  (a#3 !?t*)) !>a3))
     (equation 'a3 "(fsas1 ,s3 ,a2 ,s1) t*))

(law tri-sas-31 asap ((t* triangle) s1 a3 s2 a1)
     ((= (length (s#1 !?t*)) !>s1)
      (= (angle  (a#3 !?t*)) !>a3)
      (= (length (s#2 !?t*)) !>s2))
     ((= (angle  (a#1 !?t*)) !>a1))
     (equation 'a1 "(fsas1 ,s1 ,a3 ,s2) t*))

(law tri-sas-32 asap ((t* triangle) s1 a3 s2 a2)
     ((= (length (s#1 !?t*)) !>s1)
      (= (angle  (a#3 !?t*)) !>a3)
      (= (length (s#2 !?t*)) !>s2))
     ((= (angle  (a#2 !?t*)) !>a2))
     (equation 'a2 "(fsas1 ,s2 ,a3 ,s1) t*))

(law tri-sas-33 asap ((t* triangle) s1 a3 s2 s3)
     ((= (length (s#1 !?t*)) !>s1)
      (= (angle  (a#3 !?t*)) !>a3)
      (= (length (s#2 !?t*)) !>s2))
     ((= (length (s#3 !?t*)) !>s3))
     (equation 's3 "(fsas2 ,s1 ,a3 ,s2) t*))


;triangle angle-side-angle laws

(law tri-asa-12 asap ((t* triangle) a2 s1 a3 s2)
     ((= (angle  (a#2 !?t*)) !>a2)
      (= (length (s#1 !?t*)) !>s1)
      (= (angle  (a#3 !?t*)) !>a3))
     ((= (length (s#2 !?t*)) !>s2))
     (equation 's2 "(fasa ,a2 ,s1 ,a3) t*))
```

```
(law tri-asa-13 asap ((t* triangle) a2 s1 a3 s3)
     ((= (angle  (a#2 !?t*)) !>a2)
      (= (length (s#1 !?t*)) !>s1)
      (= (angle  (a#3 !?t*)) !>a3))
     ((= (length (s#3 !?t*)) !>s3))
     (equation 's3 "(fasa ,a3 ,s1 ,a2) t*))

(law tri-asa-21 asap ((t* triangle) a1 s2 a3 s1)
     ((= (angle  (a#1 !?t*)) !>a1)
      (= (length (s#2 !?t*)) !>s2)
      (= (angle  (a#3 !?t*)) !>a3))
     ((= (length (s#1 !?t*)) !>s1))
     (equation 's1 "(fasa ,a1 ,s2 ,a3) t*))

(law tri-asa-23 asap ((t* triangle) a1 s2 a3 s3)
     ((= (angle  (a#1 !?t*)) !>a1)
      (= (length (s#2 !?t*)) !>s2)
      (= (angle  (a#3 !?t*)) !>a3))
     ((= (length (s#3 !?t*)) !>s3))
     (equation 's3 "(fasa ,a3 ,s2 ,a1) t*))

(law tri-asa-31 asap ((t* triangle) a1 s3 a2 s1)
     ((= (angle  (a#1 !?t*)) !>a1)
      (= (length (s#3 !?t*)) !>s3)
      (= (angle  (a#2 !?t*)) !>a2))
     ((= (length (s#1 !?t*)) !>s1))
     (equation 's1 "(fasa ,a1 ,s3 ,a2) t*))

(law tri-asa-32 asap ((t* triangle) a1 s3 a2 s2)
     ((= (angle  (a#1 !?t*)) !>a1)
      (= (length (s#3 !?t*)) !>s3)
      (= (angle  (a#2 !?t*)) !>a2))
     ((= (length (s#2 !?t*)) !>s2))
     (equation 's2 "(fasa ,a2 ,s3 ,a1) t*))


;triangle side-side-side laws

(law tri-sss-1 asap ((t* triangle) s1 s2 s3 a1)
     ((= (length (s#1 !?t*)) !>s1)
      (= (length (s#2 !?t*)) !>s2)
      (= (length (s#3 !?t*)) !>s3))
     ((= (angle  (a#1 !?t*)) !>a1))
     (equation 'a1 "(fsss ,s1 ,s2 ,s3) t*))
```

```
(law tri-sss-2 asap ((t* triangle) s1 s2 s3 a2)
     ((= (length (s#1 !?t*)) !>s1)
      (= (length (s#2 !?t*)) !>s2)
      (= (length (s#3 !?t*)) !>s3))
     ((= (angle  (a#2 !?t*)) !>a2))
     (equation 'a2 "(fsss ,s2 ,s1 ,s3) t*))

(law tri-sss-3 asap ((t* triangle) s1 s2 s3 a3)
     ((= (length (s#1 !?t*)) !>s1)
      (= (length (s#2 !?t*)) !>s2)
      (= (length (s#3 !?t*)) !>s3))
     ((= (angle  (a#3 !?t*)) !>a3))
     (equation 'a3 "(fsss ,s3 ,s2 ,s1) t*))


;angle addition at vertices

(law add-angles asap ((*a angle) val a1 a2)
     ((= (angle !?*a) !>val))
     ()
     (do ((nbrs (get *a 'neighbors) (cdr nbrs))
          (sums (get *a 'sumangles) (cdr sums))
          (ante antecedents)
          (unas unassigned)
          (antecedents antecedents ante)
          (unassigned unassigned unas)
          (ok1 nil nil)
          (ok2 nil nil))
         ((null nbrs))
         (cond ((getv "(= (angle ,(car nbrs)) !>a1))
                (setq ok1 t)))
         (cond ((getv "(= (angle ,(car sums)) !>a2))
                (setq ok2 t)))
         (cond ((or ok1 ok2)
                (equation-nocheck (cond (ok2 a2) (t (car sums)))
                          "(&+ ,val ,(cond (ok1 a1) (t (car nbrs))))
                          *a)))))

(law reflex-angle asap ((*a angle) val)
     ((= (angle !?*a) !>val))
     ()
     (equation-nocheck 360.0 "(&+ ,(get *a 'reflexangle) ,val) *a))
```

APPENDIX II: The GEL Simplifier Rules

Note: In addition to the following simplification rules (which should be interpreted as lhs → rhs,) GEL also has a few rules involving commutativity of function arguments embodied in the arithmetic package.

```
(simp-rules
( (fsas1 (fasa !#x !#y !#z) !,z !,y)                ,x)
( (fsas1 (fsas2 !#x !#y !#z) !>nil !,x)             ,y)
( (fsas1 (fsas2 !#x !#y !#z) !>nil !,z)             ,y)
( (fsas1 !#y !#z (fasa !#x !,y !,z))                (&+ 180.0 (&* -1 ,x) (&* -1 ,y)))
( (fsas1 !#x !>nil (fsas2 !,x !#y !#z))             (fsas1 ,x ,y ,z))
( (fsas1 !#z !>nil (fsas2 !#x !#y !,z))             (fsas1 ,z ,y ,z))
( (fsas1 !>nil (fsas1 !#x !#y !#z) !,z)             ,y)
( (fsas1 !#z (fsas1 !#x !#y !,z) !>nil)             (fsas1 ,z ,y ,x))
( (fsas1 !#y (fsss !#x !,y !#z) !,z)                (fsss ,y ,x ,z))
( (fsas1 !#z (fsss !#x !#y !,z) !,y)                (fsss ,z ,x ,y))
( (fsss !>nil (fasa !#x !#y !#z) !,y)               ,z)
( (fsss !>nil !#y (fasa !#x !,y !#z))               ,z)
( (fsss (fsas2 !#x !#y !#z) !,x !,z)                ,y)
( (fsss (fsas2 !#x !#y !#z) !,z !,x)                ,y)
( (fsss !#x (fsas2 !,x !#y !#z) !,z)                (fsas1 ,x ,y ,z))
( (fsss !#z (fsas2 !#x !#y !,z) !,x)                (fsas1 ,z ,y ,x))
( (fsss !#x !#z (fsas2 !,x !#y !,z))                (fsas1 ,x ,y ,z))
```

```
( (fsss !#z !#x (fsas2 !,z !#y !,x))                    (fsas1 ,z ,y ,x))
( (fsas2 (fasa !#x !#y !#z) !,z !,y)                    (fasa ,z ,y ,x))
( (fsas2 !#y !#z (fasa !#x !,y !,z))                    (fasa ,z ,y ,x))
( (fsas2 !#z (fsas1 !#x !#y !,z) !>nil)                 ,x)
( (fsas2 !>nil (fsas1 !#x !#y !#z) !,z)                 ,x)
( (fsas2 !#z (fsas1 !#x !#y !,z) !,y)                   ,x)
( (fsas2 !#y (fsas1 !#x !,y !#z) !,z)                   ,x)
( (fasa (fsas1 !#x !#y !#z) !,z !,y)                    ,x)
( (fasa !#y !#z (fsas1 !#x !,y !,z))                    (fsas2 ,x ,y ,z))
( (fasa (fsss !#x !#y !#z) !,z !>nil)                   ,x)
( (fasa (fsss !#x !#y !#z) !,y !>nil)                   ,x)
( (fasa !>nil !#z (fsss !#x !#y !,z))                   ,y)
( (fasa !>nil !#y (fsss !#x !,y !#z))                   ,z)
( (fasa !#z (fasa !#x !#y !,z) !>nil)                   (fasa ,z ,y ,x))
( (&+ 180.0 (&* -1.0 (fsas1 !#x !#y !#z)) (&* -1.0 !,y))  (fsas1 ,z ,y ,x))
( (&+ 180.0 (&* -1.0 !#y) (&* -1.0 (fsas1 !#x !,y !#z)))  (fsas1 ,z ,y ,x)))
```