# Case Analysis in Euclidean Geometry: An Overview

Nathaniel Miller

Cornell University Department of Mathematics
nat@math.cornell.edu

**Abstract.** This paper gives a brief overview of **FG**, a formal system for doing Euclidean geometry whose basic syntactic elements are geometric diagrams, and which has been implimentented as the computer system **CDEG**. The computational complexity of determining whether or not a given diagram is satisfiable is also briefly discussed.

## 1 A Diagrammatic Formal System

To begin with, consider Euclid's first proposition, which says that an equilateral triangle can be constructed on any given base. While Euclid wrote his proof in Greek with

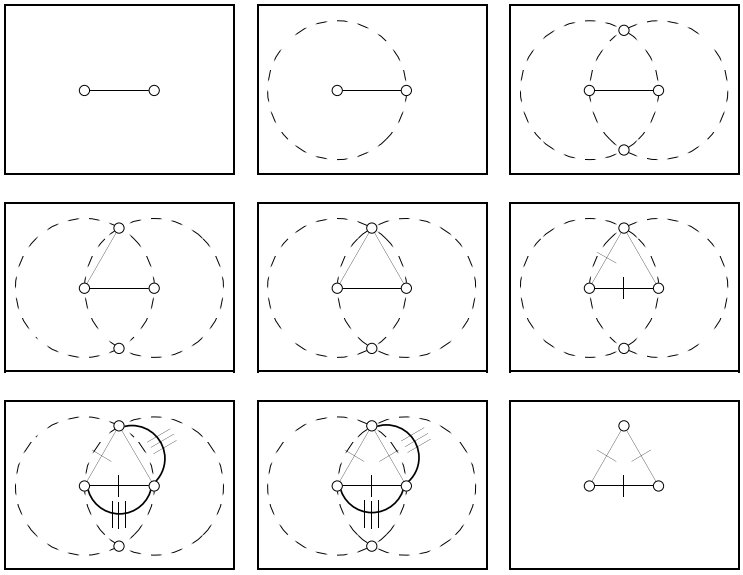

**Fig. 1.** Euclid's first proposition.

a single diagram, the proof that he gave is essentially diagrammatic, and is shown in Figure 1. Diagrammatic proofs like this are common in informal treatments of geometry, and the diagrams in Fig. 1 follow standard conventions: points, lines, and circles in a Euclidean plane are represented by drawings of dots and different kinds of line segments, which do not have to really be straight, and line segments and angles can be marked with different numbers of slash marks to indicate that they are congruent to one

another. In this case, the dotted segments in these diagrams are supposed to represent circles, while the solid segments represent pieces of straight lines.

It has often been asserted that proofs like this, which make crucial use of diagrams, are inherently informal. The comments made by Henry Forder in *The Foundations of Euclidean Geometry* are typical: "Theoretically, figures are unnecessary; actually they are needed as a prop to human infirmity. Their sole function is to help the reader to follow the reasoning; in the reasoning itself they must play no part." [2, p.42] Traditional formal proof systems are sentential—that is, they are made up of a sequence of sentences. Usually, however, these formal sentential proofs are very different from the informal diagrammatic proofs. A natural question, then, is whether or not diagrammatic proofs like the one in Fig. 1 can be formalized in a way that preserves their inherently diagrammatic nature.

The answer to this question is that they can. In fact, the derivation contained in Fig. 1 is itself a formal derivation in a formal system called **FG**, which has also been implemented in the computer system **CDEG**. These systems are based a precisely defined syntax and semantics of Euclidean diagrams. We can define a diagram to be a particular type of geometric object satisfying certain conditions; this is the syntax of our system. We can give a formal definition of which arrangements of lines, points, and circles in the plane are represented by a given diagram; this is the semantics. Finally, we can give precise rules for manipulating the diagrams—rules of construction and inference. All of these rules can be made entirely precise and implemented on a computer. The details are too long and technical to include here, but the interested reader can find them in [3]. A crucial idea, however, is that all of the meaningful information given by a diagram is contained in its topology, in the general arrangement of its points and lines in the plane. Another way of saying this is that if one diagram can be transformed into another by stretching, then the two diagrams are essentially the same. This is typical of diagrammatic reasoning in general.

## 2 Case Analysis

One of the most interesting aspects of these systems is how they treat case analysis. Any system for doing Euclidean geometry must deal somehow with geometric constructions. For example, Euclid's first postulate says that a line segment can be constructed joining any two given points. These formal systems have a corresponding rule that was used in the third and fourth steps of the derivation in Fig. 1. How does the computer know what diagrams could result from connecting the two dots in the third diagram in Fig. 1? The answer is that it returns all diagrams in which the two dots are connected by a new segment. In this particular case, there is only one such possible diagram, because of the restrictions placed on what a diagram can look like. Often, however, there will be more than one possibility, and in that case the computer will return them all.

For example, consider the situation shown in Fig. 2a. How many different ways can points $C$ and $D$ be connected? As it turns out, there are nine different topologically distinct possibilities, as **CDEG** confirms, which are shown in Fig. 2b. Thus, in both the formal system **FG** and in its computer implementation **CDEG**, when you apply the rule

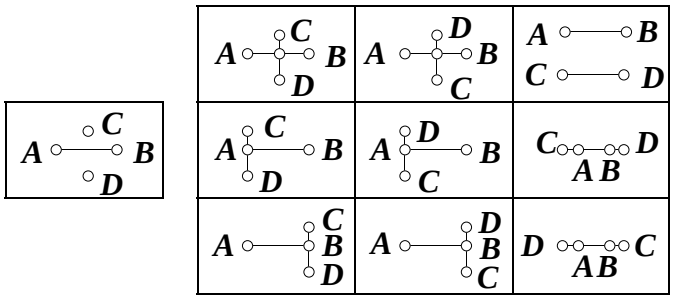


**Fig. 2.** (a and b) What can happen when points $C$ and $D$ are connected?

that says you can connect any two given points by a line segment to points $C$ and $D$ in the diagram in Fig. 2a, you get a disjunctive array of all nine of the diagrams in Fig. 2b.

## 3 Satisfiable and Unsatisfiable Diagrams

An important requirement for the construction rules in a formal system for doing geometry is that they be *sound*—that is, that they really do return every single physically realizable possibility. As one would expect, the construction rules used in **FG** and **CDEG** are indeed sound. We might also hope, however, that they would also be *frugal*—that they would never return any extra, unsatisfiable cases. This, unfortunately, is not the case. Our definition of what a diagram is allows us to draw diagrams that do not represent any physically realizable situation. For example, one could draw a diagram in which two different lines intersected in more than one place, since the lines in the diagram are not required to be straight. This particular example is eliminated by our definition of what constitutes a well-formed diagram, but there are other more complicated examples which are not, and sometimes some of these unsatisfiable diagrams may be returned by one of the construction rules.

Is it possible to eliminate these extra cases? Yes, but not efficiently. It is possible to determine whether or not a given diagram is satisfiable, by writing down a formula of first order logic that is satisfiable over the real numbers if and only if the given diagram is satisfiable; and we can determine if this formula is satisfiable by Tarski's famous decision procedure. It turns out that in this particular case we can determine if the formula is satisfiable in polynomial space. However, it also turns out that the problem of determining whether or not a given diagram is satisfiable is at least NP-hard, which roughly means that any possible procedure for deciding if a given diagram is satisfiable is too inefficient for practical use. For further details, see [3].

## References

1. Euclid, *Elements*, T. L. Heath, ed., second edition, New York: Dover, 1956.
2. Forder, Henry George, *The Foundations of Euclidean Geometry*, Cambridge: Cambridge University Press, 1927.
3. Miller, Nathaniel, "Case Analysis in Euclidean Geometry," available at `http://www.math.cornell.edu/~nat/diagrams`.