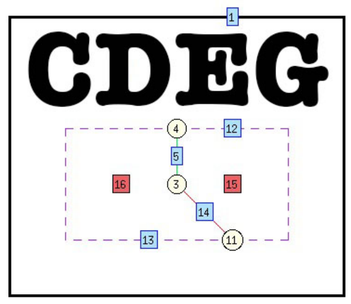
Graph Drawing for Formalized Diagrammatic Proofs in Geometry

Nathaniel Miller, University of Northern Colorado

nathaniel.miller@unco.edu



Abstract

This poster describes the use of planar graphs within **CDEG**, a computerized formal system for giving diagrammatic proofs in Euclidean geometry. It describes how **CDEG** uses an automated graph drawing algorithm to draw its diagrams, which are represented internally as planar graphs, and discusses possible ways to improve this algorithm to improve comprehension of the diagrams. It also discusses the Lemma Incorporation problem, an important but difficult graph-theoretic question in this domain.

Introduction

CDEG, "Computerized Diagrammatic Euclidean Geometry," is a computerized formal system for giving diagrammatic proofs in Euclidean geometry which uses planar graphs in drawing its diagrams. Here we discuss some of the graph-theoretic problems that arise in this context. This computer proof system implements a diagrammatic formal system for giving diagram-based proofs of theorems of Euclidean geometry that are similar to the informal proofs found in Euclid's *Elements*. The theoretical ideas underlying this system and the original version of **CDEG** are described in detail in the book *Euclid and his Twentieth Century Rivals: Diagrams in the Logic of Euclidean Geometry* [4]. A much updated version of **CDEG** is now publicly available, and can be downloaded from [1]. Interested readers are encouraged to download **CDEG** and to try it out for themselves.

When we say that **CDEG** is a diagrammatic computer proof system, this means that it allows its user to give geometric proofs using diagrams. Internally, **CDEG** represents a diagram abstractly as a planar graph along with some additional information about how elements of the graph relate to the geometric objects they represent. The nodes in the graph represent points in the plane, while the edges represent line segments and arcs of circles. In general, one diagram drawn by **CDEG** can actually represent many different possible collections of lines and circles in the plane. What these collections all share, and share with the diagram that represents them, is that they all have the same topology. This means that any one can be stretched into any other, staying in the plane. So, for example, a diagram containing a single line segment represents all possible single line segments in the plane, since any such line segment can be stretched into any other.

CDEG Diagrams

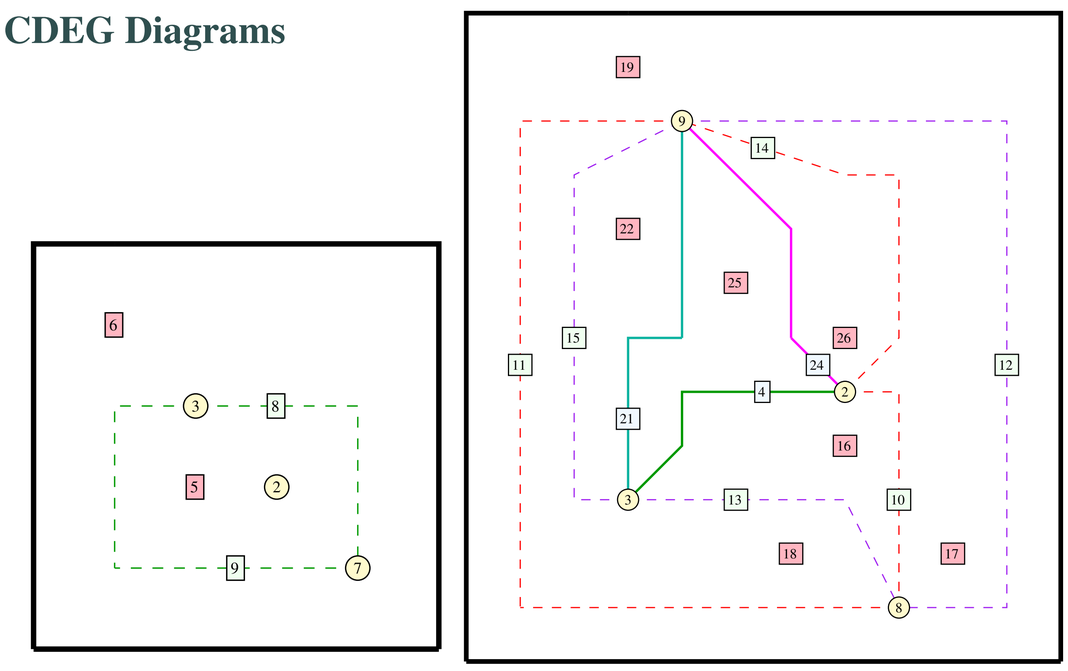


Figure 1: Two **CDEG** diagrams.

Two sample **CDEG** diagrams are shown in Figure 1. The first diagram represents a circle drawn with a point inside of it and two points on its circumference. The circle is drawn in a way that appears to be rectangular rather than circular, but recall that all we care about here is the topology of the diagram. The second diagram, which occurs in the proof of Euclid's Proposition 1, shows an equilateral triangle inscribed in the intersection of two circles. In **CDEG** diagrams, edges that represent pieces of circles are drawn with dotted lines, while edges that represent pieces of straight lines are drawn with solid lines. The user of the diagram can tell which edges represent parts of the same circle or line because they are drawn with the same color. All elements of the planar graph—its nodes, edges, and regions—are labeled with numbers with which we can refer to them.

Construction Rules and Case Analysis

Like more traditional formal systems that are sentential (that is, that operate on sentences that are strings of symbols in some formal language), **CDEG** has rules of inference that allow us to infer one diagram from another. In addition, and unlike traditional sentential formal systems, **CDEG** also has geometric construction rules. When one of these rules is applied to diagram, we may get a set of several different possible resulting diagrams. **CDEG** uses a depth-first search algorithm to identify all the possible topologically distinct planar graphs that extend the starting graph by adding the newly constructed piece.

As an example, consider the first diagram shown in Figure 1. What should happen if we connect points 3 and 7 with a line segment? In the diagram as drawn, point 2 appears to be collinear with points 3 and 7, and, indeed, one possibility is that point 2 will line on the line segment connecting points 3 and 7. However, the diagram represents all cases where there is a circle with two points on its circumference and a point inside, so there are two other possibilities: point 2 could lie on the left side of this segment, or on the right side of this segment. Thus, when **CDEG**'s construction rule is used to draw the segment connecting points 3 and 7, **CDEG** returns 3 possible resulting primitive diagrams, with point 2 on, to the left of, and to the right of the new line segment that is drawn. The three resulting primitive diagrams are shown in Figure 2.

CDEG uses a depth-first search algorithm to identify the possibilities. The new segment from point 3 to point 7 must start at point 3. From there, it can then enter region 6 or region 5. If edges 8 and 9 were line segments rather than arcs of circles, they would also be considered as possible paths for the new segment to take. If the segment enters a region, it must next hit something else inside or on the boundary of that region. If this results in a known-to-be-impossible situation, that branch is pruned from the search; otherwise, **CDEG** continues to extend the segment until the segment reaches its target (in this case, point 7), pruning impossible branches as it goes.

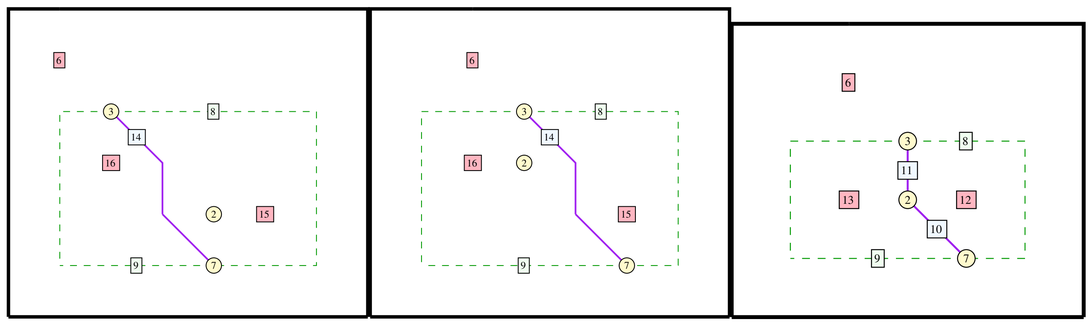


Figure 2: The three possible results when connecting points 3 and 7.

Diagram Layout Algorithms

Once **CDEG** has determined the possible diagrams that result from one of its operations, it still has to lay them out in order to display them to the user. Each diagram is represented internally as a data structure that encapsulates its topological structure as a planar graph. This data structure doesn't contain any geometric information about how to actually draw the graph in the plane. Therefore, when **CDEG** needs to display a diagram to the user, it relies on the open source library OGDF (the "Open Graph Drawing Framework" [2]) to lay out the diagrams using the Mixed Model algorithm of Gutwenger and Mutzel [3]. Before it can call OGDF, **CDEG** first has to modify its graphs in several ways:

- New nodes are added for each region label; and
- New dummy nodes and edges are added to make the graph simple and connected.

This layout method works to draw **CDEG**'s diagrams efficiently, and the Mixed Model layout algorithm that is used is designed for good angular resolution [3], which increases the amount of space between the various edges coming out of each node and thereby increases the readability of the resulting diagrams. However, it also has several disadvantages that decrease the readability of the diagrams:

1. The dummy nodes and edges that are not drawn can leave extra whitespace and unnecessary bends in the diagrams.
2. No attempt is made to encourage edges representing straight lines to be laid out in a straight line.
3. Likewise, when a line or circle passes through a point, no attempt is made to encourage the two edges representing the two sides of the line or circle to be collinear with each other.
4. Adding an object to a diagram can radically change its layout.

It seems likely that **CDEG**'s graph drawing algorithms could be modified in ways that would still run efficiently but would increase the human readability of the resulting diagrams.

Lemma Incorporation

It is normal in any system for giving proofs to have some way to incorporate and reuse previously proven results in subsequent proofs. However, in **CDEG**'s diagrammatic setting, a previously proven result takes the form of saying that a concluding diagram can be inferred from a starting diagram. If diagram D_2 can be derived from D_1 in **CDEG**, we notate this by writing $D_1 \vdash D_2$. In order to use this result in a later proof, we would like to be able to apply it to any diagram D_1' that contains D_1 as a subdiagram—that is, from which D_1 can be obtained by erasing elements of the original diagram. We notate this by writing $D_1 \subset D_1'$.

Now we can formally define how Lemma Incorporation should work. If $D_1 \vdash D_2$ and $D_1 \subset D_1'$, then we would like to be able to infer $\mathcal{D}_2'$, where $\mathcal{D}_2'$ is the set of minimal diagrams that contain both D_1' and D_2 as subdiagrams. A diagram D_2' is minimal if it doesn't contain a subdiagram that still contains D_1' and D_2 as subdiagrams.

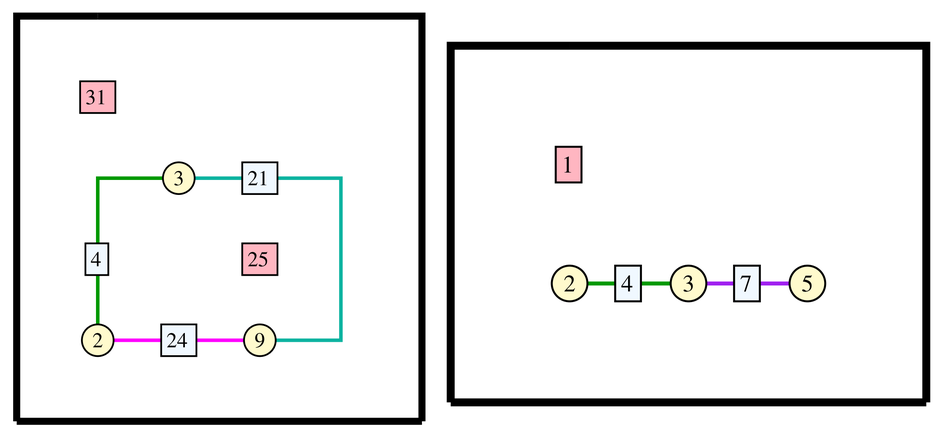


Figure 3: How many ways are there to combine these two diagrams?

As an example, consider the two diagrams shown in Fig. 3. The left one is the final diagram from Euclid's Proposition 1, while the second is the diagram from Proposition 2 to which he applies Proposition 1 to draw a triangle on segment 4. To mimic Euclid's proof, we need to be able to efficiently find all possible ways to combine these to diagrams. This is at heart an open natural graph-theoretic problem that doesn't appear to have been previously discussed in the graph-drawing literature.

References

[1] **CDEG** download page, `http://www.unco.edu/NHS/mathsci/facstaff/Miller/personal/CDEG/`

[2] Chimani, Markus, *et. al.* 2007. "The Open Graph Drawing Framework." *15th International Symposium on Graph Drawing 2007 (GD '07).*

[3] Gutwenger, Carsten and Petra Mutzel. 1998. "Planar Polyline Drawings with Good Angular Resolution." *6th International Symposium on Graph Drawing 1998, Montreal (GD '98).* Springer Lecture Notes in Computer Science 1547: 167-182.

[4] Miller, Nathaniel. 2007. *Euclid and his twentieth century rivals: Diagrams in the logic of Euclidean geometry.* Stanford, CA: CSLI Press.