CDEG: Computerized Diagrammatic Euclidean Geometry 2.0

Nathaniel Miller

University of Northern Colorado,
Greeley, CO, USA,
`nat@alumni.princeton.edu`,
`http://www.unco.edu/NHS/mathsci/facstaff/Miller/personal/diagrams/`

Abstract. This paper describes **CDEG**, a computerized formal system for giving diagrammatic proofs in Euclidean geometry.

Keywords: diagrams, computer formal systems, Euclidean geometry

1 Introduction

This paper describes the computer proof system **CDEG**, version 2.0 . **CDEG** stands for "Computerized Diagrammatic Euclidean Geometry." 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* [2]. It is based on the diagrammatic formal system **FG**, which is described in detail in my book, *Euclid and his Twentieth Century Rivals: Diagrams in the Logic of Euclidean Geometry* [14]. That book also describes an earlier version of **CDEG**; however, **CDEG** has evolved significantly since the publication of the book. In particular, a beta version of **CDEG** is now publicly available, and can be downloaded from

`http://www.unco.edu/NHS/mathsci/facstaff/Miller/personal/CDEG/`.

I encourage interested readers of this paper 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. It is based on a precisely defined syntax and semantics of Euclidean diagrams. To say that it has a precisely defined syntax means that all the rules of what constitutes a diagram and how we can move from one diagram to another have been completely specified. The fact that these rules are completely specified is perhaps obvious if you are using the formal system on a computer, since computers can only operate with such precisely defined rules. However, it was commonly thought for many years that it was not possible to give Euclidean diagrams a precise syntax, and that the rules governing the use of such diagrams were inherently informal. (See [14, Section 1.1] for a history of the use of diagrams in formal geometry.)

To say that the system has a precisely defined semantics means that the meaning of each diagram has also been precisely specified. 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. See [14] for more details concerning the syntax and semantics of Euclidean diagrams.

2 CDEG diagrams

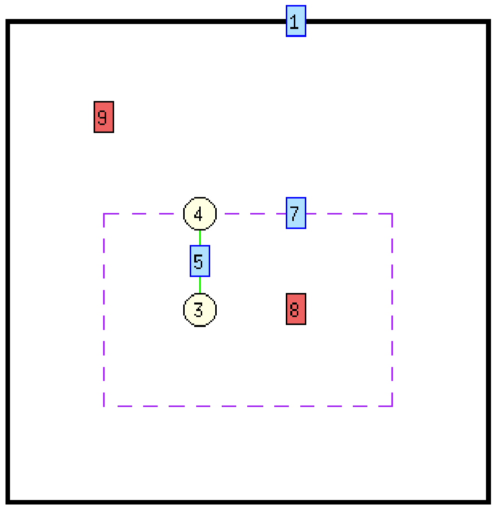

Fig. 1. A **CDEG** diagram showing the second step in the proof of Euclid's First Proposition.

A sample **CDEG** diagram is shown in Figure 1. This diagram, which occurs in the proof of Euclid's first proposition, represents a line segment along with a circle drawn with that line segment as its radius.

CDEG diagrams contain two kinds of objects: ***dots*** and ***segments***. The dots, which are shown as light yellow circles, represent points in the plane, while the segments represent pieces of lines and circles. There are actually two different kinds of segments that can occur in a diagram: ***solid segments***, which represent pieces of lines, and ***dotted segments***, which represent pieces of circles. The dots and segments of a diagram are enclosed by a bold line called the ***frame***. The dots and segments of a diagram, along with the frame, break the diagram into a collection of ***regions***. Every dot, segment, region, and piece of the frame in a **CDEG** diagram is labeled with a number, so that we can refer to it. Dots are labeled with a number inside the yellow circle; segments are labeled with a number in a blue box along the segment, and regions are labeled with a number in a red box somewhere in the middle of the region.

The segments in a diagram are part of diagrammatic lines and circles. Each line and circle is assigned a different color, so that all of the segments that make up a given line or circle with be the same color. In general, segments that are not part of the same line or circle have different colors, although once there are a lot of different lines and circles, the colors are assigned randomly, so occasionally different lines may have similar colors. Lines may continue to the frame, in which case we consider them to be infinite in that direction. Thus, infinite lines intersect the frame in two places, rays intersect it in one place, and line segments don't intersect it at all.

Just as with traditional informal geometric diagrams, pieces of **CDEG** diagrams may be marked equal. Markers can mark line segments, angles, or areas in the diagram. Rather than being marked with slash marks on the diagram, as would be done with some informal diagrams, the markings are printed as text along with the diagram. (Representations that combine text with diagrams are sometimes referred to as ***heterogenous*** representations.) Line segments are represented by collections of solid line segments; these are referred to as ***diagrammatic segments*** or ***dsegs***. If n solid segments all intersect a dot d, then they divide the 360° around d into n different smallest possible angles. We refer to these as ***primative angles***, or ***pangles***, and label them by a pair of numbers (nd, ns), where nd is the number of the dot at the vertex, and ns is the number of the segment on the counterclockwise side of the primitive angle. In general, angles are represented in diagrams by collections of these primitive angles, which we refer to as ***diagrammatic angles*** or ***di-angles***. Finally, areas of the graph are represented by collections of regions, which we refer to as ***region sets***. Markers in **CDEG** diagrams can mark dsegs, di-angles, or region sets to be equal to other dsegs, di-angles, or region sets.

Each **CDEG** diagram represents all of the possible equivalent collections of points, lines, and circles with the same topology as the diagram. However, it is often the case that when we perform some geometric operation on a diagram, such as connecting two points with a line, there may be multiple possible outcomes depending on which of these equivalent collections you start with. **CDEG** allows for this possibility through the use of ***diagram arrays***. A diagram array is just a collection of several possible diagrams showing different possible outcomes. To distinguish them from diagram arrays, we often refer to the single diagrams that make up the arrays as ***primitive diagrams***, or ***pds*** for short.

CDEG allows you to manipulate diagrams through the use of construction and inference rules. These rules are meant to be ***sound***. This means that if you start with a diagram D that represents a collection $\mathcal{C}$ of points, lines, and circles in the plane, and you apply a rule to D, then at least one of the diagrams in the resulting diagram array should still represent $\mathcal{C}$ (or, in the case of a construction rule that added a line or circle to the diagram, $\mathcal{C}$ with the appropriate line or circle added). For further details, see [14].

3 CDEG vs. Euclid's Elements

CDEG is designed so that there is a fairly direct correspondence between the construction and inference rules that it uses and the Postulates, Common Notions, and Definitions that Euclid uses in the *Elements*. These connections are laid out in Table 1. Those commands in **CDEG** that do not have a direct analogue in the Postulates, Common Notions, and Definitions of the *Elements* are set out in Table 2. Euclid's Postulates, Common Notions, and Definitions are given here in Sir Thomas Heath's literal English translation [2]. For discussion of the **CDEG** commands and how they are used, see the **CDEG** *User's Manual* [12].

Note that **CDEG** does not contain many inference rules other than those set out in Euclid's Postulates, Common Notions, and Definitions. It is commonly asserted that Euclid's *Elements* contains many subtle gaps that can only be repaired by adding further postulates in the manner, for example, of Hilbert's *Foundations of Geometry* [4]. These added postulates typically have to do with issues of betweenness, continuity, and the intersections of geometric objects. For example, in the proof of Euclid's first postulate, Euclid assumes without apparent justification that the two circles constructed in the course of the proof must intersect. (**CDEG**'s version of this proof is discussed in Section 6.) However, in **CDEG**, this issue is solved by the underlying diagrammatic machinery. The two circles do, in fact, intersect in the unique diagram that is produced after adding the two circles, which gives us an intersection point we can use in the rest of the derivation. Similarly, most of the other "gaps" in Euclid's reasoning are taken care of by the diagrammatic machinery. Thus, we can view these as being part of an unarticulated diagrammatic process rather than as flaws in Euclid's arguments.

One particular way that **CDEG** differs from Euclid's *Elements* is in its adoption of the Side-Angle-Side and Side-Side-Side triangle congruence criteria as primitive rules rather than as propositions to be derived. In the *Elements*, these are derived as Propositions 4 and 8 using the principle of superposition and Common Notion 4. (The principle of superposition states that two geometric objects must be congruent if it is possible to move one directly onto the other through a sequence of isometries. An isometry is a shape-preserving transformation, such as a translation, rotation, or reflection.) It is possible to incorporate the principle of superposition directly into a diagrammatic formal system by adding appropriate transformation rules that allow us to reason about the result of applying an isometry to a figure; this is the approach taken by **FG**, as described in [14, Section 3.3]. However, these transformation rules would have been much, much more complicated to program into **CDEG** than the triangle congruence criteria were. Furthermore, they can lead to an exponential blowup in the

		Euclid's version	**CDEG**'s version(s)
Postulate	1	To draw a straight line from any point to any point.	`con<n>ect dots`
	2	To produce a finite straight line continuously in a straight line.	`extend segment in <o>ne direction`; `e<x>tend segment`
	3	To describe a circle with any centre and distance.	`draw <c>ircle`
	4	That all right angles are equal to one another.	This is provable in **CDEG** in the same manner given by Hilbert in his *Foundations of Geometry* [4, Theorem 15], and therefore is not included as a primitive rule.
	5	That, if a straight line falling on two straight lines make the interior angles on the same side less than two right angles, the two straight lines, if produced indefinitely, meet on that side on which are the angles less than the two right angles.	`<e>rase diagram: <E>uclid's fifth postulate`
Common Notion	1	Things which are equal to the same thing are also equal to one another.	`<c>ombine markers`
	2	If equals be added to equals, the wholes are equal.	`<a>ddition of marked objects`
	3	If equals be subtracted from equals, the remainders are equal.	`<d>ifference of marked objects`
	4	Things which coincide with one another are equal to one another.	`mark <o>ne single object`; `apply <S>AS`; `appl<y> SSS`
	5	The whole is greater than the part.	`<e>rase diagram: <s>eg c.`; `<a>ng c.`; `<r>egion c.`
Definition	15	A **circle** is a plane figure contained by one line such that all the straight lines falling upon it from one point among those lying within the figure are equal to one another.	`mark <r>adii`

Table 1. The correspondence between **CDEG**'s rules and Euclid's Postulates, Common Notions, and Definition 15.

Other	1	These commands allow you to add new dots to the diagram. Euclid freely adds new arbitrary dots to segments without justification—for example, in the proof of Proposition 5. These commands are also useful in creating the starting diagram for a derivation.	`<a>dd dot to segment`; `add dot to <r>egion`
	2	This command allows objects to be removed from a diagram. It was unnecessary for Euclid to include such a rule, since he could always just ignore parts of a diagram. However, this rule is frequently useful in making diagrams more readable and in reducing the number of cases to be considered.	`<d>elete objects`
	3	These are not sound inference rules; they exist in **CDEG** for the purpose of creating starting diagrams.	`<f>ree <e>rase diagram`; `<f>ree marking`
	4	These **CDEG** commands are not directly part of derivations, but rather concern the operation of the program.	`get <h>elp`; `<p>rint diagram as text`; `<q>uit`; `se<t> pd`; `<s>ave/load diagrams`

Table 2. CDEG commands that don't have direct analogues in Euclid's Postulates, Common Notions, or Definitions.

number of cases that need to be considered, and Euclid only uses superposition in order to prove these two propositions. Thus, it was simpler to build these two triangle congruence criteria into **CDEG** as primitive rules. This also explains why these triangle congruence rules are listed in Table 1 as corresponding to Common Notion 4.

Because it duplicates all of Euclid's proof methods, **CDEG** should be theoretically able to prove versions of all of Euclid's Propositions from the first four books of the *E*lements, which is the part that deals purely with planar geometry.

However, anyone who tries to actually use **CDEG** to duplicate these books will quickly realize that in practice it will be very difficult to use **CDEG** to duplicate all of Euclid's proofs. One issue is that as the diagrams become more complicated, the amount of time required for one step in the proof can grow exponentially. In particular, the commands that draw circles and lines can take an amount of time that is exponential in the number of objects in a diagram. Thus, some computations may take impractically long. Furthermore, the result of applying a construction rule to a single primitive diagram is a diagram array containing all of the topologically distinct possible diagrams that could occur when the newly constructed object is added, and the number of new diagrams in such an array can also be exponential in the number of objects in the original diagram. So the number of cases that need to be considered can also grow very quickly. See [14, Section 4.1] for more discussion of this phenomenon.

A related issue is that of unsatisfiable diagrams. Ideally, we would want **CDEG**'s construction rules to return as few diagrams as possible, in order to minimize the number of cases that need to be considered. Unfortunately, these rules do sometimes produce diagrams that represent arrangements of circles and lines that cannot be physically realized with actual straight lines and circles. It turns out that this is practically unavoidable. In [13], it is shown that the question of determining which diagrams that result from applying a construction rule are physically realizable is at least NP-hard, which means that it is not practically computable in a reasonable amount of time.

Another issue that exacerbates the problem of an exploding number of cases to consider is the lack of lemma incorporation in **CDEG**. Lemma incorporation refers to the use of previously derived propositions and lemmas in proving new theorems. Most proof systems include the ability to do this, but **CDEG** does not, because it is technically harder to implement lemma incorporation in a diagrammatic setting. Of course, previously derived propositions and lemmas can always be rederived in the course of a proof. However, the additional objects in the diagram in the course of a later proof normally necessitate considering even more cases. Thus, the lack of lemma incorporation here can lead to a huge blowup in the length of a given proof. For a much more extensive discussion of lemma incorporation, again see [14, Section 4.1]. Lemma incorporation will hopefully be included in some future version of **CDEG**.

In the meantime, one way to use **CDEG** is to try to duplicate one of Euclid's proofs, but to only complete the proof for one branch of the many possible cases that arise. This allows the user to avoid tediously looking at many different cases that are all essentially similar. Interestingly, this actually mirrors Euclid's normal practice. In the *Elements*, he normally only gives a proof for one single case, the hardest one, leaving the other cases for the reader to fill in. For discussion of this practice, see Heath's commentary on Proposition 2 in [2].

4 Why a computer system?

CDEG is essentially a computer implementation of the formal system **FG** described in [14]. Actually implementing the formal system on a computer was a highly non-trivial matter that took several years worth of work. Why would we want to implement an existing formal system on a computer?

The first reason that we might want a computer implementation is to demonstrate that this system really is completely formal: that the diagrams that are being manipulated are, indeed, completely specified as formal objects, and that the rules of the system are completely specified on these objects. With traditional, sentential formal systems, we do this by writing our axioms in a formal language, and then carefully writing rules of inference as typographical manipulations of sentences in this formal language. However, when our formal objects are diagrams, it is difficult to achieve this level of specificity without a computer implementation. Diagrams are complicated formal objects, and we have very strong informal intuitions about how they should work that may cloud our ability to judge if our rules have been completely formally specified.

Furthermore, even if our rules are completely formally specified, without a computer implementation, it will be quite difficult to play with the formal system to see what derivations are like, and to make sure that they really work the way that we think they will. This is particularly true in geometry, where constructions can lead to case branching, with a large number of cases that are virtually impossible to keep track of without using a computer. Thus, we may not be able to prove everything we think we can.

This worry is not just academic. Several other diagrammatic formal systems have been proposed by other researchers and have appeared in print but have later turned out to have ill-defined and/or unsound rules. For example, Isabel Luengo's formal system **DS1**, described in [7] and [6], turned out to be unsound, as explained in [14, Appendix C]. Likewise, John Mumma's **Eu**, described in [9], [10], [11], and, at a previous conference in the Diagrams series, in [8], is also unsound, as described in [15]. Neither of these proposed formal systems for geometry was implemented as a computer system, and neither worked quite in the way that their designers intended. Furthermore, both systems were examined by quite a number of article referees and dissertation committee members who failed to notice their significant problems. Thus, we should approach any proposed diagrammatic formal system with a certain amount of healthy skepticism. A working computer system is one way to allay some of this skepticism.

Secondly, a computer system is the only way to make a formal system widely available. Many potential users will not be able to make sense out of a formal system that is just specified mathematically, but will be able to try out a computer implementation. One author of a review of [14], writing before **CDEG** became publicly available, put it this way: "It is therefore disappointing that the implementation **CDEG** is not, as far as I can see, publicly available. Aside from the question of whether the system is completely formal, it also seems to me that without some computer implementation the system may not be widely taken up." [5] Hopefully, now that the formal system is available as a computer program, a larger variety of people will be interested in it and will try it out.

The third reason for a computer implementation is to be able explore exactly what the formal system is able to prove. Above, I claim that **CDEG** should be able to duplicate the first four books of Euclid's *Elements*. The only way to verify this claim is to systematically go through each of Euclid's proofs, and to see how to duplicate it within **CDEG**. To date, I have done this with many different proofs from Euclid's Book I, but have not yet gone systematically through all of Euclid's proofs. This is a future project of mine, and one that would be essentially impossible without the computer implementation.

One small example of the way that the computer implementation sheds light on the formal system has to do with the way that **CDEG** handles subtraction of segments. Euclid's Common Notion 3 states that "If equals be subtracted from equals, the remainders are equal." Common Notion 3 is discussed in [14, Section 3.4]. In this section, it is explained that **FG** lacks a rule for subtraction of segments, since this rule would be derivable from the rule for addition of segments. A proof is given for how to derive this rule, and one step involves copying a length from one point to another "by using circles, or else by using the transformation rules." As it turns out, this is not quite correct. While coping a length can be done in **FG** using the transformation rules, it can't be done with circles without already having the subtraction rule. Since **CDEG** replaces the transformation rules with triangle congruence rules, if this derivation can't be done with circles, then it can't be done at all. Why can't it be done with circles? It is because Euclid's third Postulate only allows us to draw circles with a given center and point on the circumference. If we want to draw a circle at a given point using a radius defined by a segment at a different point, we will have to use the construction given in Euclid's Proposition 2. This construction, however, requires using segment subtraction. In other words, we can prove Proposition 2 using segment subtraction, or else we can prove segment subtraction using Euclid's Proposition 2. This means that segment subtraction and Proposition 2 are equivalent with respect to **CDEG**'s other rules. Thus, unlike in **FG**, segment subtraction is included in **CDEG** as a primitive rule. Without it, neither Proposition 2 nor anything that depends on it would have been derivable in **CDEG**. However, this only became apparent to me when I tried to prove Proposition 2 using an earlier version of **CDEG** that lacked the segment subtraction rule, and discovered that I couldn't.

5 Changes from the previous version of CDEG

As mentioned above, a previous version of **CDEG**, version 1.0, was discussed in [14], but was never made publicly available. The main reason that it was never made publicly available had to do with the way that it produced drawings of diagrams. Internally, **CDEG** represents a diagram 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 produce a graphical representation of the diagram by laying out the graph in the plane. The original version of **CDEG** discussed in [14] produced a layout for each graph by calling a separate program, GDToolkit's BLAG ("Batch LAyout Generator") to produce an orthogonal planar layout of the original graph, and then called a third program, GraphWin, part of LEDA ("Library of Efficient Data Types and Algorithms") to actually draw the resulting graph. This worked to produce a graphical representation of the graphs, but made it impossible to make **CDEG** publicly available. Potential users would have had to installed these other two programs on their computers in order to use **CDEG**, and, furthermore, both programs were constantly evolving and required an expensive licensing fee to use. I therefore needed to find an alternative means for **CDEG** to lay out its graphs.

The new version of **CDEG** relies instead on OGDF (the "Open Graph Drawing Framework," which is described in [1]) to lay out its diagrams. This framework is licensed under the Gnu Public License (GPL), and I was therefore able to incorporate it into **CDEG** without any licensing issues, since **CDEG** is also being released under the GPL. This new version uses the mixed model algorithm of Gutwenger and Mutzel [3] to lay out the diagrams. This algorithm has the additional advantage that it produces much more readable diagrams than the orthogonal layout algorithm employed by BLAG, since the segments leaving a given vertex are generally much more evenly arranged. OGDF is linked to **CDEG** when it is compiled, which means that end users only have to install **CDEG**.

Other significant changes to **CDEG** include the addition of the triangle congruence rules and rules for deleting pieces of diagrams, as well as numerous bug fixes. It also now has the ability to draw its own output diagrams rather than relying on an external program to do this. However, the version of **CDEG** that is now available is a beta version and most likely still contains bugs. If you try out **CDEG** and discover any bugs, please let me know by sending an email to `nat@alumni.princeton.edu`.

6 A Sample CDEG session

As a brief example, this section shows how to reproduce the proof of Euclid's Proposition I from Book I of the *Elements*. A more detailed version of this example is given as a tutorial in the **CDEG** *User's Manual* [12].

First we start **CDEG** and ask it what commands are available:

```
Welcome to CDEG!
(Type h for help.)
CDEG(1/1)% h
Options are:
<a>dd dot to segment, draw <c>ircle,
<d>elete objects, <e>rase diagram, get <h>elp,
apply <m>arker inference rules, con<n>ect dots,
extend segment in <o>ne direction,
<p>rint diagram as text, <q>uit,
add dot to <r>egion, <s>ave/load diagrams,
se<t> pd, or e<x>tend segment.
CDEG(1/1)%
```

The prompt here (`CDEG(1/1)%`) tells us that we are currently working with the first primitive diagram in a diagram array that contains 1 primitive diagram. Since we have just started the program, this is the empty primitive diagram. This is the initial diagram that is displayed. It is shown in Figure 2. It contains a single region bounded by the frame; **CDEG** has assigned this region the number 2. **CDEG** assigns each object in a diagram a unique number by which it can be identified. Next, we use the "r" command ("`add dot to <r>egion`") to add two new dots to this region:

```
CDEG(1/1)% r
Enter region number: 2
CDEG(1/1)% r
Enter region number: 2
```

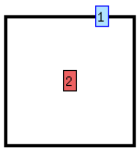

Fig. 2. The empty primitive diagram as drawn by **CDEG**.

CDEG now displays the resulting diagram, which adds two more dots, numbered 3 and 4. We can connect them using the `con<n>ect dots` command.

```
CDEG(1/1)% n
Enter first dot's number: 3
Enter second dot's number: 4
```

The resulting diagram is shown in Figure 3; this is the starting diagram for our proof of Euclid's Proposition 1. Next, we will `draw a <c>ircle` centered at dot

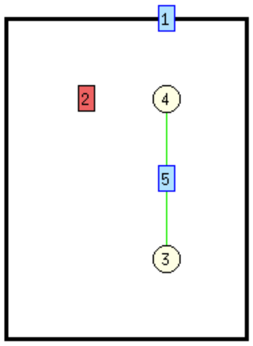

Fig. 3. A **CDEG** diagram showing a single line segment.

3 and going through dot 4.

```
CDEG(1/1)% c
Enter center dot's number: 3
Enter radius dot's number: 4
```

The resulting diagram is the one that was shown in Figure 1 in Section 2. The diagrammatic circle in this diagram looks rectangular rather than circular, but all we care about here is the topology of the diagram.

Next, we want to draw another circle centered at dot 4 and going through dot 3. We will then form a triangle by connecting the endpoints of the segment to one of the points, dot number 12, on the intersection of the two circles.

```
CDEG(1/1)% c
Enter center dot's number: 4
Enter radius dot's number: 3
CDEG(1/1)% n
Enter first dot's number: 3
Enter second dot's number: 12
CDEG(1/1)% n
Enter first dot's number: 4
Enter second dot's number: 12
```

The resulting diagram is shown in Figure 4.

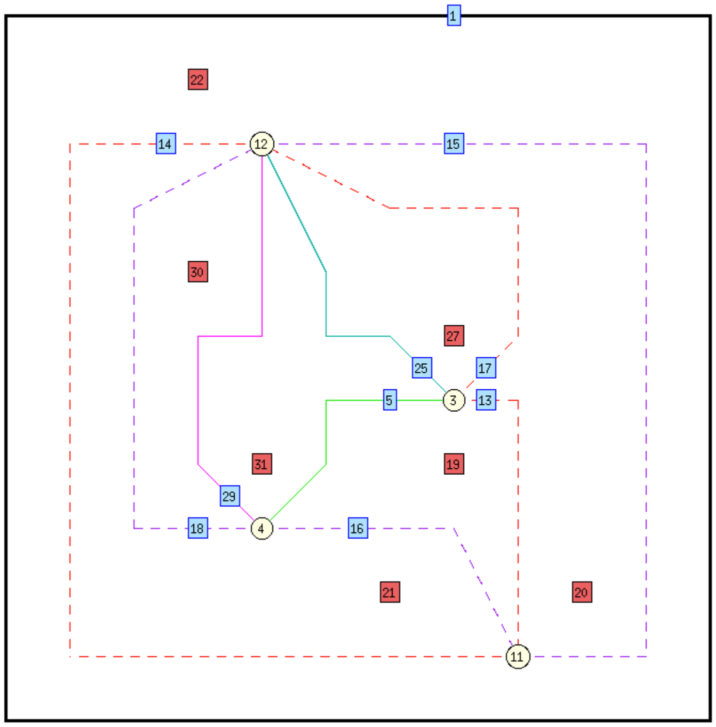

Fig. 4. A **CDEG** diagram showing the fourth step in the proof of Euclid's First Proposition

Next, we want to mark segment 5 congruent to segment 25. We can do this using the `mark <r>adii` command in the `apply <m>arker inference rules` menu. This command lets us mark congruent all radii of a given circle. In order to use it, we must identify the circle that the radii are part of by identifying one of the segments that make it up. In this case, the segments are both part of circle dcirc10, which is the purple circle in Figure 4, and we can identify this circle by indicating segment 16, which lies on this circle.

```
CDEG(1/1)% m
Marker inference rule to apply:
 (choices are:
<a>ddition of marked objects, <c>ombine markers,
<d>ifference of marked objects, <f>ree marking,
mark <o>ne single object, mark <r>adii,
apply <S>AS, or appl<y> SSS) r
Enter the number of one seg on the circle: 16
marker33 marks DSeg(solidseg25) DSeg(solidseg5)
```

Similarly, we can mark segment 5 congruent to segment 29 because they are both radii of the other circle.

```
CDEG(1/1)% m
CDEG(1/1)% m
Marker inference rule to apply:
 (choices are:
<a>ddition of marked objects, <c>ombine markers,
<d>ifference of marked objects, <f>ree marking,
mark <o>ne single object, mark <r>adii,
apply <S>AS, or appl<y> SSS) r
Enter the number of one seg on the circle: 17
marker34 marks DSeg(solidseg29) DSeg(solidseg5)
marker33 marks DSeg(solidseg25) DSeg(solidseg5)
```

The diagram that is displayed here is the same as that previously displayed and shown in Figure 4. The congruence markings that have been added are displayed as accompanying text.

Finally, we can combine these two markings using the `<c>ombine markers` command, which is also found in the `apply <m>arker inference rules` menu. This command takes the place of transitivity: if we have a geometric object that is marked with two different markers, we can combine them into one marker that marks everything that is marked by either marking. Note that a dseg may be made up of several pieces, so the dseg is specified as a list of segments.

```
CDEG(1/1)% m
Marker inference rule to apply:
 (choices are:
<a>ddition of marked objects, <c>ombine markers,
<d>ifference of marked objects, <f>ree marking,
mark <o>ne single object, mark <r>adii,
apply <S>AS, or appl<y> SSS) c
Type of marker to combine:
(choices are <s>eg, <a>ng, or <r>egion) s
Enter dseg:
```

```
Enter next seg index, or 0 to quit:5
Enter next seg index, or 0 to quit:0

marker34 marks DSeg(solidseg29) DSeg(solidseg25) DSeg(solidseg5)
```

Finally, we may want to clean up the diagram by erasing the superfluous pieces added in the course of our construction. We can do this using the `<d>elete objects` command.

```
CDEG(1/1)% d
Type of object to erase:
 (choices are <p>oint, <c>ircle, <l>ine,
line <e>nds, <m>arker, or all <u>nused markers.)
c
Enter the number of one seg on the circle: 16
marker34 marks DSeg(solidseg29) DSeg(solidseg25) DSeg(solidseg5)
CDEG(1/1)% d
Type of object to erase:
 (choices are <p>oint, <c>ircle, <l>ine,
line <e>nds, <m>arker, or all <u>nused markers.)
c
Enter the number of one seg on the circle: 17
marker34 marks DSeg(solidseg29) DSeg(solidseg25) DSeg(solidseg5)
CDEG(1/1)% d
Type of object to erase:
 (choices are <p>oint, <c>ircle, <l>ine,
line <e>nds, <m>arker, or all <u>nused markers.)
p
Enter dot to erase:
11
marker34 marks DSeg(solidseg29) DSeg(solidseg25) DSeg(solidseg5)
```

The resulting diagram is shown in Figure 5, and shows an equilateral triangle on our original base.

Thus, we have shown how to construct an equilateral triangle on the given base, duplicating Euclid's Proposition 1.

References

1. Chimani, Markus, Carsten Gutwenger, Michael Jünger, Karsten Klein, Petra Mutzel, and Michael Schulz. 2007. "The Open Graph Drawing Framework." *15th International Symposium on Graph Drawing 2007 (GD '07)*, Sydney, Australia.
2. Euclid. 1956. *The Elements*. New York: Dover, 2nd edn. Translated with introduction and commentary by Thomas L. Heath.
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.

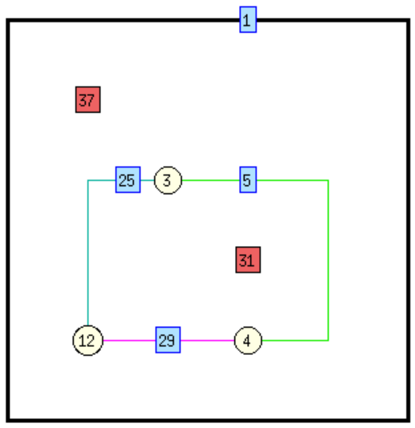

Fig. 5. A **CDEG** diagram showing the triangle obtained in the final step of the proof of Euclid's First Proposition.

4. Hilbert, David. 1971. *Foundations of Geometry*. La Salle, Ill.: Open Court Publishing Co., 4th edn. Translated by Leo Unger.
5. Lack, Stephen. 2008. "Book Review: Euclid and His Twentieth Century Rivals." *Australian Mathematical Society Gazette*, Vol. 34, No. 4, September 2008, pp. 274–275.
6. Luengo, Isabel. 1995. *Diagrams In Geometry*. Ph.D. thesis, Indiana University.
7. Luengo, Isabel. 1996. "A Diagrammatic Subsystem of Hilbert's Geometry." In G. Allwein and J. Barwise, eds., *Logical Reasoning with Diagrams*. New York: Oxford University Press.
8. Mumma, John. 2008. "Ensuring generality in Euclid's diagrammatic arguments." In G. Stapleton, J. Howse, and J. Lee, eds., *Diagrammatic Representation and Inference: 5th International Conference, Diagrams 2008*. Springer Lecture Notes in Computer Science 5223: 222–235.
9. Mumma, John. 2006. *Intuition formalized: Ancient and modern methods of proof in elementary geometry*. Ph.D. Dissertation, Carnegie Mellon University. Retrieved May 20, 2010 from `http://www.contrib.andrew.cmu.edu/~jmumma/list.html`.
10. Mumma, John. 2010. "Proofs, pictures, and Euclid." *Synthese* Volume 175 (2): 255–287. DOI 10.1007/s11229-009-9509-9.
11. Mumma, John. 2008. "Review of Euclid and his twentieth century rivals: Diagrams in the logic of Euclidean geometry." *Philosophia Mathematica* 16(2): 256–264.
12. Miller, Nathaniel. **CDEG** *User's Manual*. Available from `http://www.unco.edu/NHS/mathsci/facstaff/Miller/personal/CDEG/`.
13. Miller, Nathaniel. 2006. Computational complexity of diagram satisfaction in Euclidean geometry. *Journal of Complexity* 22(2):250–274.
14. Miller, Nathaniel. 2007. *Euclid and his twentieth century rivals: Diagrams in the logic of Euclidean geometry*. Stanford, CA: CSLI Press.
15. Miller, Nathaniel. 2012. "On the Inconsistency of Mumma's Eu," *Notre Dame Journal of Formal Logic*, in press. Preprint available at `http://www.unco.edu/NHS/mathsci/facstaff/Miller/personal/diagrams`.