Case Analysis in Euclidean Geometry

Nathaniel Miller

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

Abstract. Case analysis has long been a sticking point in attempts to understand how diagrams are used in mathematics, particularly in geometry. When using diagrams, a question that immediately arises is: how many different diagrams must I consider? Indeed, one of the earliest criticisms of Euclid's Elements, which argues extensively using diagrams, was that he didn't distinguish enough cases. Some more recent commentators have argued that proofs that rely on diagrams are inherently informal because the process of finding all of the cases that need to be considered is a non-algorithmic human process that is perhaps even open-ended: each time someone finds a case that hasn't been dealt with yet, a new proof has to be constructed for that case. In this paper, I will show how we can use a well-defined syntax and semantics of diagrams to understand how case analysis works in Euclidean geometry.

In fact, case analysis in Euclidean geometry can be done by an algorithm, which has been implemented in the computerized formal proof system **CDEG** (Computerized Diagrammatic Euclidean Geometry). **CDEG** automatically does this case analysis in the course of constructing a proof. However, there is a sense in which geometric case analysis cannot be done by a computer. Consider the problem of finding all of the new diagrams that can result from extending a line segment in a given diagram outward until it intersects another element of the diagram. The algorithm used by **CDEG** solves this problem in polynomial time, but may return extra diagrams that don't represent any physically realizable situation. There is in fact a computable algorithm which returns precisely those diagrams that represent the physical situations that could occur when you extend the line–that is, it doesn't return any extra unrealizable cases–but we'll show that the problem of finding precisely the realizable cases is at least NP-hard, which roughly means that no algorithm can solve this problem in a reasonable amount of time.

1 Basic Syntax of Diagrams

First, we must say what is meant by the term diagram in this context. Figure 1 shows two examples of the sort of diagrams we want to consider. They contain dots and edges representing points, straight lines and circles in the plane, but the diagram may not look exactly like the configuration of lines and circles that it represents; in fact, it may represent an impossible configuration, like the second diagram in figure 1.

Formally, we define a diagram as follows:

Definition 1. *A **primitive Euclidean diagram** D is a geometric object in the plane that consists of*

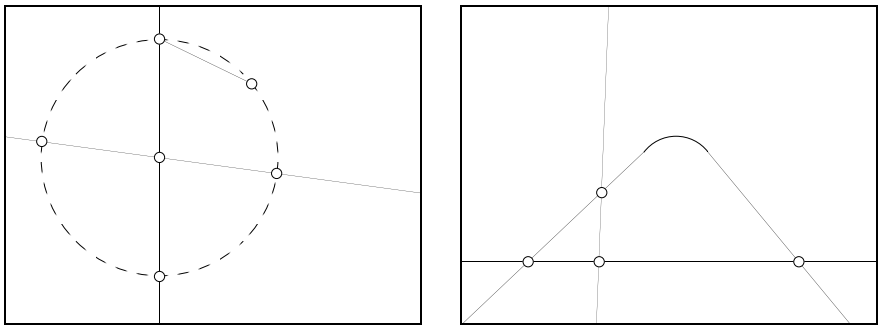

Fig. 1. Two primitive diagrams.

1. *a rectangular box drawn in the plane, called a* ***frame****;*
2. *a finite set DOTS*(D) *of* ***dots*** *which lie inside the area enclosed by the frame, but cannot lie directly on the frame;*
3. *two finite sets SOLID*(D) *and DOTTED*(D) *of* ***solid and dotted line segments*** *which connect the dots to one another and/or the frame, and such that each line segment*
   (a) *lies entirely inside the frame,*
   (b) *is made up of a finite number of connected pieces that are either straight lines or else arcs of circles,*
   (c) *does not intersect any other segment, any dot, the frame, or itself except at its endpoints, and*
   (d) *either forms a single closed loop, or else has two endpoints, each of which lies either on the frame or else on one of the dots;*
4. *a set SL*(D) *of subsets of SOLID*(D)*, such that each segment in SOLID*(D) *lies in exactly one of the subsets; and*
5. *a set CIRC*(D) *of ordered pairs, such that the first element of the pair is an element of DOTS*(D) *and the second element of the pair is a subset of DOTTED*(D)*, and such that each dotted segment in DOTTED*(D) *lies in exactly one of these subsets.*

The intent here is that the primitive diagram represents a Euclidean plane containing points, straight lines and line segments, and circles. The dots represent points, the solid line segments in SOLID(D) represent straight line segments, and the dotted line segments in DOTTED(D) represent parts of circles. SL(D) tells us which solid line segments are supposed to represent parts of the same straight line, and CIRC(D) tells us which dotted line segments are supposed to represent parts of the same circle, and where the center of the circle is. (This comment is intended only to motivate the definitions being made now, and will be explained more carefully later on.) The sets in SL(D) are called ***diagrammatic lines***, or ***dlines*** for short, and the pairs in CIRC(D) are called ***diagrammatic circles*** or ***dcircles***. Elements of dlines are said to lie on the dline, and likewise, elements of the second component of a dcircle are said to lie on the dcircle; the first component of a dcircle is called the ***center*** of the dcircle. Each solid line segment must lie on exactly one dline, and each dotted line segment must lie on exactly one dcircle. A dline or dcircle is said to intersect a given dot (or the frame) n times if it has n component segments with endpoints on that dot (or on the frame), counting a

segment twice if both of its endpoints lie on the frame or on the same dot. Notice that it follows from the preceding that dlines and dcircles can only intersect other dline and dcircles at dots.

We would now like to ensure that the dlines and dcircles come together at a dot in a way that mimics the way that real lines and circles could meet at a point. To this end, we first define the notion of diagrammatic tangency:

Definition 2. *If each of e and f is a dcircle or dline that intersects the dot d exactly twice, then e and f are defined to be **diagrammatically tangent** (or **dtangent**) at d if they do not cross each other at d.*

This means that if s_{e1} and s_{e2} are the segments that are part of e which intersect d and, likewise, s_{f1} and s_{f2} are the segments from f that intersect d, then if s_{f1} occurs between s_{e1} and s_{e2} when the segments that intersect d are listed in clockwise order, then s_{f2} also occurs between s_{e1} and s_{e2} in this list. We are going to require the dcircles and dlines to intersect at d in such a way that the dtangency relation is transitive—in other words, so that if e and f intersect at d without crossing and f and g intersect at d without crossing, then e and g don't cross either (although they might both lie on the same side of f). Since dtangency is automatically symmetric and reflexive, this makes it into an equivalence relation. We can then extend the notion of diagrammatic tangency to dlines that only intersect d once by specifying that if e is such a dline, and e intersects d directly between two members of the same dtangency equivalence class, then e is dtangent to all of the members of that equivalence class. A dline that only intersects d once is said to ***end*** at d.

We can now define a dot d to be viable as follows:

Definition 3. *A dot is **viable** if*

1. *any dcircle that intersects the dot intersects it exactly twice;*
2. *any dline that intersects the dot intersects it at most twice;*
3. *the dcircles and dlines that intersect d do so in such a way so as to make the dtangency relation transitive; and*
4. *no two dlines are dtangent at d.*

A primitive diagram D is viable if every dot in D is viable.

It follows from the preceding that if one member of a dtangency equivalence class crosses f at d, then all of the other members of the dtangency class also cross f at d (since otherwise some other member of the class would be dtangent to f at d, which would force them all to be dtangent to f at d), and that each dtangency equivalence class can contain at most one dline, which may or may not end at d, since dlines are not allowed to be dtangent to other dlines. Notice that viability is a local property of diagrams—it says that the diagram is locally well-behaved at each dot.

Next, we would like to ensure that the dlines and dcircles of our diagrams behave like real lines and circles. We do this with the following definition.

Definition 4. *A primitive diagram D is **well-formed** if it is viable and*

1. *no dotted line segment in D intersects the frame;*

2. *no two line segments intersect the frame at the same point;*
3. *every dline and dcircle in D is connected—that is, given any two dots that a dline or dcircle P intersects, there is a path from one to the other along segments in P; and*
4. *every dcircle in D is made up of segments that form a single closed loop, and the center of the dcircle lies inside that loop.*

The ***ends*** of a dline occur where it intersects the frame or a dot which it only intersects once; it follows from the preceding that each dline must have exactly two ends. We call a dline that intersects the frame twice a ***proper dline***; one that intersects the frame once a ***d-ray***; and one that doesn't intersect the frame at all a ***dseg*** (not to be confused with the solid line segments that make it up). A well-formed primitive diagram is also called a ***wfpd***. Both of the diagrams in figure 1 are well-formed.

(In principle, the diagrams should also tell you which segments make up each dline and dcircle. In the case of the first diagram in figure 1, if we know that there are three dlines and one dcircle in this diagram, there are three different ways that the segments can be assigned to dlines and dcircles that make this a wfpd, as the reader should be able to check. Notice that if we are told that there are no dtangencies in a wfpd in which every dline is proper, then there is only one way to assign segments to dlines and dcircles that is consistent with the diagram being well-formed, because you can determine which segments belong to the same dline or dcircle at a given dot by looking at the clockwise order in which the segments intersect the dot. In practice, it is usually clear which segments are intended to belong to the same dline or dcircle, and we won't indicate this unless it is unclear. We could also prove a theorem showing that every viable primitive diagram is equivalent to one in which two segments that intersect at a given dot are on the same dline iff they locally lie on a straight line, and are on the same dcircle iff they locally lie on some circle.)

Finally, we have the following:

Definition 5. *A primitive diagram is **nicely well-formed** if it is well-formed and*

1. *no two dlines intersect more than once;*
2. *no two dcircles intersect more than twice;*
3. *no dline intersects any dcircle more than twice; and*
4. *given any two non-intersecting proper dlines, if there is a third dline that intersects one of them, then it also intersects the other.*

The last clause of this definition makes non-intersection of dlines an equivalence relation, and corresponds to the uniqueness of parallel lines. The first diagram in figure 1 is nicely well-formed under two of the three assignments of segments to dlines and dcircles that make it a wfpd. Nicely-well-formed primitive diagrams are also called ***nwfpd***s.

Notice that the conditions for being viable are local conditions, the conditions for being well-formed are global conditions effecting individual dlines and dcircles, and the conditions for being nicely well-formed effect how dlines and dcircles can interact with one another globally.

2 Advanced Syntax: Corresponding Graph Structures and Equivalence Classes of Diagrams

We have now defined a primitive diagram to be a particular kind of geometric object. These diagrams contain somewhat too much information, though. The diagrams are supposed to show the topology of how lines and circles might lie in the plane. So we'd really like to look at equivalence classes of diagrams that contain the same topological information. In order to do this, we are going define for each diagram an algebraic structure called a ***corresponding graph structure*** (abbreviated ***cgs***). The definition will be somewhat technical, but the idea is simple: the diagram's corresponding graph structure just abstracts the topological information contained in the diagram. Another way of saying this is that our definition will have the property that two diagrams will have isomorphic corresponding graph structures just if there is a homeomorphism between the diagrams that preserves all of their topological structure. A diagram D's corresponding graph structure will contain four kinds of information: a graph G that contains information about how the dots, frame, and segments intersect; for each point of intersection, information about the clockwise order in which the segments and frame intersect the point; for each doubly connected component of G, a two-dimensional cell complex showing how the different regions of G lie in the plane; and for every singly connected component of G (except for the outermost component), information about which region of the graph it lies in. (Recall that two vertices v_1 and v_2 in a graph G are said to be ***singly connected*** if there is a path from v_1 to v_2 in G, and they are said to be ***doubly connected*** if for any edge e of G, there is a path from v_1 to v_2 in the graph obtained from G by removing edge e. Being singly or doubly connected are equivalence relations, and their equivalence classes are called the singly or doubly connected components of G; singly connected components are often just called connected components.) The notion of a cgs will be useful because we really want to think of two diagrams as being the same if they contain the same topological information, and so we will form equivalence classes of diagrams that have the same (isomorphic) corresponding graph structures. The corresponding graph structures are nice, constructive, algebraic objects that we can manipulate, reason formally about, or enter into a computer, rather than working directly with the equivalence classes. The data structures that **CDEG** uses to represent diagrams are essentially a version of these corresponding graph structures.

We start by defining the appropriate type of algebraic structure to capture the topology of a diagram.

Definition 6. *A **diagram graph structure** S consists of*

1. *a set of vertices $V(S)$;*
2. *a set of edges $E(S)$;*
3. *for each vertex v in $V(S)$, a (cyclical) list $L(v)$ of edges from $E(S)$ (which lists in clockwise order the edges that are connected to v, telling us how to make the edges and vertices into a graph);*
4. *a cell-complex for each doubly connected component of the graph;*
5. *a function er_S from the non-outermost connected components of the graph to the two-cells of the cell-complexes (er stands for "enclosing region", and this function tells us which region each connected component lies in);*

6. *a subset DOTS(S) of $V(S)$;*
7. *two subsets of $E(S)$, called SOLID(S) and DOTTED(S);*
8. *a set SL(S) of subsets of $E(S)$; and*
9. *a set CIRC(S) of pairs whose first element is a vertex and whose second element is a set of edges.*

We can now show how to construct a given diagram's corresponding graph structure. First note that the segments of a diagram D intersect the frame in a finite number of points, which divide the frame into a finite number of pieces. We refer to these points as ***pseudo-dots*** and to these pieces as ***pseudo-segments***.

Definition 7. *A diagram D's **corresponding graph structure** is a diagram graph structure S with the following properties:*

1. *$V(S)$ contains one vertex $G(d)$ for each dot or pseudo-dot d in D.*
2. *$E(S)$ contains one edge for every segment and pseudo-segment in D.*
3. *If d is any dot or pseudo-dot in D, then $L(G(d))$ lists the edges corresponding to the segments and pseudo-segments that intersect d, in the clockwise order in which the segments and pseudo-segments intersect d.*
4. *For each doubly connected component P of the graph G defined by $V(S)$, $E(S)$, and the lists $L(v)$, we define its **corresponding cell complex** C_P as follows:*
   - *C_P contains two-dimensional cells, one-dimensional cells, and zero-dimensional cells.*
   - *For each vertex v in P, C_P contains a corresponding 0-cell $C(v)$.*
   - *For each edge e of P, C_P contains a corresponding 1-cell $C(e)$.*
   - *Note that the segments and pseudo-segments of D that correspond to edges in P break up the plane into a finite number of connected regions, since there are only finitely many of them and they are piecewise arcs of circles and lines. Furthermore, because P is doubly connected, all but one of these (which we'll call the outer region) are simply connected. For each such region r, C_P contains a corresponding two-cell $C(r)$.*
   - *C_P is put together by connecting the one-cells to the zero-cells so that the boundary of $C(e)$ is the set containing $C(v_1)$ and $C(v_2)$ iff e connects v_1 and v_2 in G; and then attaching the two-cells to the resulting cell-complex so that the boundary of $C(r)$ is the loop that traverses $(C(G(s_1)), C(G(s_2)), \ldots, C(G(s_n)))$ in order iff the boundary of r in D consists precisely of $(s_1, s_2, \ldots, s_n)$ in clockwise order.*
5. *For each connected component p of G that does not contain the edges corresponding to the pieces of the frame, $er_S(p)$ is the unique two-cell $c = C(r)$ such that*
   - *the parts of D that correspond to p lie entirely in r, and*
   - *if they also lie entirely in a region r' corresponding to some other two-cell S, then r is contained in r'.*
6. *The sets DOTTED(S), SOLID(S), DOTS(S), SL(S), and CIRC(S) are defined such that an element a of S is in one of these sets iff the corresponding element of D is in the corresponding set in D.*

This definition now allows us to say what it means for two diagrams to contain the same information.

Definition 8. *Two diagrams D and E are **equivalent** (in symbols, $D \equiv E$) if they have isomorphic corresponding graph structures.*

This is an equivalence relation, and we normally won't distinguish between equivalent diagrams.

A ***di-angle*** is an angle formed where two dlines intersect at a dot in a diagram. (They do not have to be adjacent to one another.) A ***marked diagram*** is a primitive diagram in which some of the dsegs and/or some of the di-angles have been ***marked***. A dseg is marked by drawing a heavy arc from one of its ends to the other and drawing some number of slash marks through it. If the dseg is made up of a single solid line segment, then it can also be marked by drawing some number of slash marks directly through the line segment. A di-angle is marked by drawing an arc across the di-angle from one dline to the other and drawing some number of slash marks through it. The arc and slash marks are called a marker; two dsegs or di-angles marked with the same number of slashes are said to be marked with the same marker. A single dseg or di-angle can be marked more than once by drawing multiple arcs. A ***diagram array*** is an array of (possibly marked) primitive diagrams, joined together along their frames. (It doesn't matter how they are joined.) Diagram arrays are allowed to be empty. Figure 2 shows a diagram array containing two different marked versions of the first diagram in figure 1.

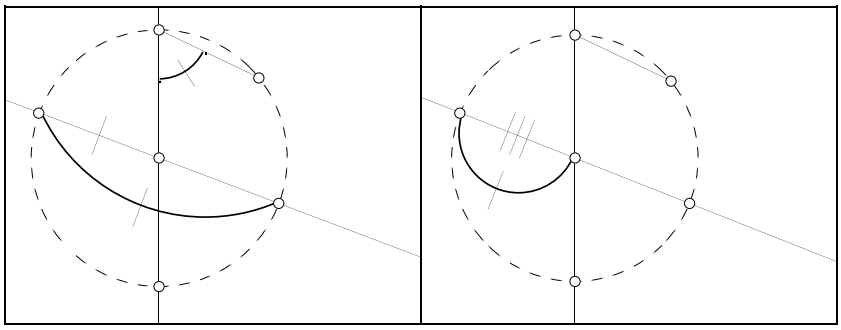

Fig. 2. A diagram array containing two marked versions of the first primitive diagram in figure 1.

We can extend our notion of diagram equivalence to marked diagrams in the natural way. We define a ***marked diagram graph structure*** to be a diagram graph structure along with a new set MARKED whose elements are sets of dsegs and sets of ordered triples of the form <vertex, edge, edge>. We next define a marked primitive diagram D's corresponding marked graph structure to consist of the corresponding graph structure of D's underlying unmarked primitive diagram along with a set MARKED that for each segment marker in D contains the set of dsegs corresponding to the segments marked by that marker, and for each di-angle marker in D contains the set of triples $<v, e_1, e_2>$ such that the di-angle with vertex corresponding to b and edges corresponding to e_1 and e_2 in clockwise order is marked with that marker. Two marked diagrams are defined to be equivalent if and only if their corresponding marked graph structures are isomorphic; and two diagram arrays are equivalent if and only if there is a bijection f from the diagrams of one to the diagrams of the other that takes diagrams to equivalent diagrams.

3 Diagram Semantics

So far, we have only talked about diagrams. Now that we know what a diagram is, we would like to discuss the relationship between diagrams and real geometric figures. By a ***Euclidean plane***, we mean a plane along with a finite number of points, circles, rays, lines, and line segments designated in it, such that all the points of intersection of the designated circles, rays, *etc.* are included among the designated points. The elements of Euclidean planes are the objects that we would like to reason about. We consider the designated points of a Euclidean plane to divide its circles and lines into pieces, which we call ***designated edges***. Note that it is very easy to turn a Euclidean plane P into a diagram. We can do this as follows: pick any new point n in P, pick a point p_l on each designated line l of P, and let m be the maximum distance from n to any designated point, any p_l, or to any point on a designated circle. m must be finite, since P only contains a finite number of designated points, lines and circles. Let R be a circle with center n and radius of length greater than m, and let F be a rectangle lying outside of R. Then if we let D be a diagram whose frame is F, whose segments are the parts of the edges of P that lie inside F, whose dots are the designated points of P, and whose dlines and dcircles are the lines and circles of P, then D is a nwfpd that we call P's ***canonical (unmarked) diagram***. (Strictly speaking, we should say a canonical diagram, since the diagram we get depends on how we pick n and the p_l; but all the diagrams we can get are equivalent, so it doesn't really matter.) We can also find P's ***canonical marked diagram*** by marking equal those dsegs or di-angles in D that correspond to congruent segments or angles in P. These canonical diagrams give us a convenient way of saying which Euclidean planes are represented by a given diagram.

Definition 9. *A Euclidean plane M is a **model** of the primitive diagram D (in symbols, $M \models D$, also read as"M satisfies D") if*

1. *M's canonical unmarked diagram is equivalent to D's underlying unmarked diagram, and*
2. *if two segments or di-angles are marked equal in D, then the corresponding segments or di-angles are marked equal in M's canonical marked diagram.*

M is a model of a diagram array if it is a model of any of its component diagrams.

This definition just says that $M \models D$ if M and D have the same topology and any segments or angles that are marked congruent in D really are congruent in M. Note that this definition makes a diagram array into a kind of disjunction of its primitive diagrams and that the empty diagram array therefore has no models.

It is immediate from the definitions that every Euclidean plane is the model of some diagram, namely its canonical underlying diagram, and that if D and E are equivalent diagrams, then if $M \models D$, then $M \models E$. In other words, the satisfaction relation is well-defined on equivalence classes of diagrams. The full converse of this statement, that if $M \models D$ and $M \models E$, then $D \equiv E$, is not true, since D and E may have different markings. However, it is true if D and E are unmarked. Also, if D is a primitive diagram that isn't nicely well-formed, then it has no models. To see this, notice that if $M \models D$, then D's underlying unmarked diagram D' is equivalent to M's canonical unmarked

diagram, which is nicely-well formed; so D' is also nicely well-formed, as diagram equivalence preserves nice well-formedness, and so D is nicely well-formed since its underlying unmarked diagram is nicely well-formed.

We are going to want to use diagrams to reason about their models. In order to do this, we are going to define construction rules that will allow us to perform operations on given diagrams which return other diagrams. So we will need some way of identifying diagrammatic elements across diagrams. To do this, we can use a ***counterpart relation***, denoted $\mathrm{cp}(x, y)$, to tell us when two diagrammatic objects that occur in different primitive diagrams are supposed to represent the same thing. Formally, the counterpart relation is a binary relation that can hold between two dots or two sets of segments in any of the primitive diagrams that occur in some discussion or proof, but never holds between two dots or sets of segments that are in the same primitive diagram. Informally, people normally use labels to identify counterparts. For example, two dots in two different diagrams might both be labeled A to show that they represent the same point. The idea of a counterpart relation is due to Shin [3].

4 Construction Rules

We would now like to be able to use diagrams to model ruler and compass constructions. In order to do this, we will define several diagram construction rules. The rules work as follows: each rule, when applied to a given nwfpd D yields a diagram array of (representatives of all the equivalence classes of) all the nwfpds that satisfy the rule (with corresponding parts of the diagrams identified by the counterpart relation). The new dlines and dcircles added by these rules are allowed to intersect any of the already existing dlines and dcircles, and the intersection points can be at new dots, as long as the resulting diagrams are still nicely well-formed. There will always be a finite number of resulting nwfpds, since each application of a rule will add a single new dot, dline, or dcircle, and the original diagram can only contain a finite number of dots and segments, none of which can be intersected more than twice by the new element, because of the conditions for niceness. The diagram construction rules are given in table 1.

Rule C3a is a special case of rule C3b, while C3b is derivable from C3a, as in Euclid's second proposition. Rules C1, C2, and C3a correspond to Euclid's first three postulates.

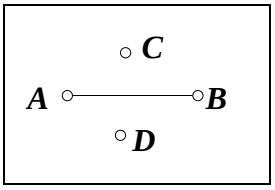

Fig. 3. What can happen when points C and D are connected?

As a relatively simple example of how these rules work, consider the diagram shown in Fig. 3. What happens if we apply rule C1 to this diagram in order to connect points C and D? We get the diagram array of all nwfpds extending the given diagram in which there is a dseg connecting points C and D. In this case, there are nine different topologically distinct possibilities, as **CDEG** confirms, which are shown in Fig. 4.

Diagram Construction Rules

C0. A dot may be added to the interior of any region, or along any existing segment, dividing it into two segments (unless the original segment is a closed loop).

C1. If there isn't already one existing, a dseg may be added whose endpoints are any two given existing distinct dots.

C2. Any dseg (or dray) can be extended to a proper dline.

C3a. Given two distinct dots c and d, a dcircle can be added with center c that intersects d if there isn't already one existing.

C3b. Given a dot c and a dseg S, a circle can be drawn about center c, with S designated to be a ***dradius*** of the dcircle. In general, we define a dseg to be a dradius of a dcircle if it is so designated by an application of this rule or if one of its ends lies on the dcircle and the other lies on the dcircle's center.

C4. Any dline or dcircle can be erased; any solid segment of a dline may be erased, as long as the part that is left is still connected; and any dot that doesn't intersect more than one dline or dcircle and doesn't occur at the end of a dseg or dray can be erased. If a solid line segment is erased, any marking that marks a dseg or di-angle that it is a part of must also be erased.

Table 1. Diagram Construction Rules.

We can apply these construction rules to diagram arrays by applying the rules to the individual primitive diagrams contained in the array.

A construction rule is said to be ***sound*** if it always models a possible real construction, meaning that if $M \models D$ and diagram E follows from D via this rule, then M can be extended to a model of E. The rules given in table 1 are sound, because in any model, we can connect two points by a line, extend any line segment to a line, draw a circle about a point with a given radius, and we can erase points, lines, and circles. In general, if every model M of D can be extended to a model of E, then we say that E is a ***geometric consequence*** of D, and write $D|\!\sim E$. This definition of geometric consequence and the notation for it are due to Luengo [2].

A diagram E is said to be ***constructible from diagram*** D if there is a sequence of diagrams beginning with D and ending with E such that each diagram in the sequence is the result of applying one of the construction rules to the preceding diagram; such a sequence is called a construction. Because our construction rules are sound, it follows by induction on the length of constructions that if E is constructible from D, then E is a geometric consequence of D.

The computer system **CDEG** uses explicit algorithms to compute the diagram graph structure that results from applying one of the construction rules to a given diagram. These algorithms are based on the idea that if we want to know how a line can possibly continue from a given dot, it must either leave the dot along one of the already existing segments that leave the dot, or else it must enter one of the regions that the dot borders, in which case it must eventually leave that region at another dot or along another edge bordering the region, breaking the region into two pieces; along the way, it can intersect any of the pieces of any components that lie inside the region.

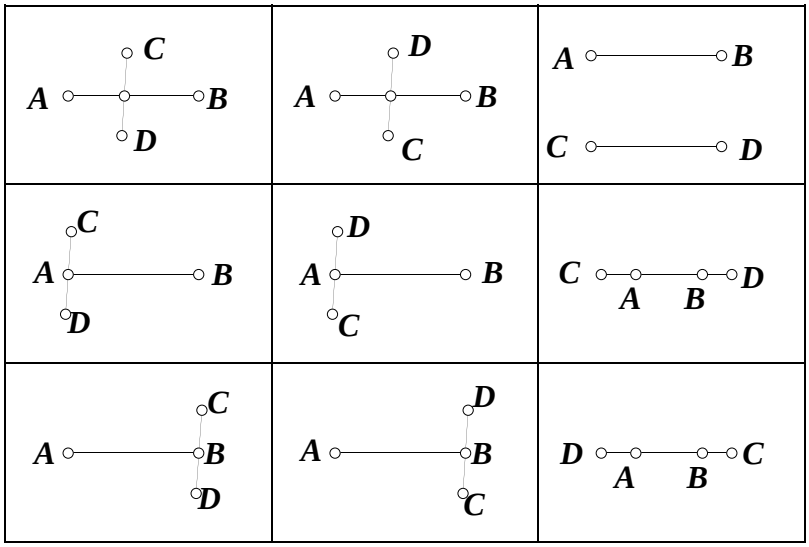

Fig. 4. The result of applying rule C1 to points C and D in the diagram in Fig. 3.

5 Satisfiable and Unsatisfiable Diagrams

In Sect. 3, we showed that only nicely well-formed primitive diagrams have models. This isn't surprising, since the definition of a nicely well-formed primitive diagram was designed to eliminate diagrams that represented unrealizable situations. A more interesting question might be: how well did our definitions succeed at eliminating these unrealizable situations? That is: did we succeed in eliminating *all* of the unsatisfiable diagrams, or are there still some nwfpds with no models?

Unfortunately, the answer is that there are indeed unsatisfiable nwfpds. Figure 5 shows a nwfpd which is unsatisfiable because according to Desargues' theorem, in any model of this diagram, line XY would have to intersect line $B'C'$ at point Z. This example is particularly striking because it contains nothing but unmarked dsegs, but there are many other possible examples.

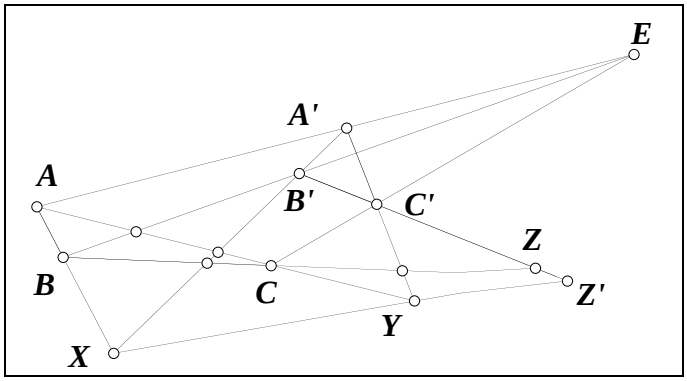

Fig. 5. An unsatisfiable nwfpd.

This shows that our definition of nice well-formedness is still too broad, because there are diagrams that are nicely well-formed, but are unsatisfiable. An obvious next question is: is there some additional set of conditions that can be added to those for nice well-formedness that will eliminate all of the unsatisfiable diagrams? It would be extremely convenient to find such a set of conditions. For example, consider what happens when we apply one of the construction rules to a satisfiable diagram. We get back an array of possible results. As it now stands, we know that because the construction rules are sound, at least one of the diagrams that we get back must be satisfiable, but many of them may not be satisfiable. If we could find a set of conditions that eliminated these unsatisfiable diagrams, then we wouldn't have to waste our time looking at these extra cases. So such a set of conditions would be extremely powerful.

The very fact that such a set of conditions would be so powerful might make us suspect they would be *too* powerful, and that such a set of conditions is impossible to find. But somewhat surprisingly, it can in fact be computed whether or not a given diagram is satisfiable. Our definition of satisfiability can be translated into the first-order language of real arithmetic, and we can apply Tarski's theorem, which says that there is a procedure for deciding if a given sentence of the first-order language of arithmetic is true or false (as a statement about the real numbers). (See [4].) It turns out that the formula that translates our definition of diagram satisfaction is Σ_1, so the decision procedure is in PSPACE. This means that we could define a primitive diagram to be ***strongly nicely well-formed*** if it is nicely well-formed and Tarski's decision procedure says that it is satisfiable. Then the strongly nicely well-formed diagrams would exactly capture the possible configurations of real Euclidean planes. The problem with this approach is that the decision procedure given by Tarski's theorem can take intractably long to run. A set of conditions that correctly determine if a diagram is satisfiable but take exponentially long to evaluate aren't really useful.

So a new question is: is there a procedure that determines whether or not a given diagram is satisfiable in a reasonable amount of time? A procedure is usually considered to be tractable in the real world only if it runs in polynomial time. So, to be more specific, our question becomes: is there a polynomial-time algorithm for determining whether or not a given diagram is satisfiable? It turns out that the answer to this question is no, assuming that P $\neq$ NP, as is widely believed to be the case. In the next section we will show that the diagram satisfiability problem is NP-hard, which means that no set of conditions that can be evaluated in polynomial time can determine if a given diagram is satisfiable.

6 NP-hardness

In this section, we show that the problem of determining if a given diagram is satisfiable is NP-hard. The problem of determining if a given boolean formula is satisfiable is well known to be NP-complete, so it suffices to show how to reduce the boolean satisfiability problem to the diagram satisfiability problem in log-space.

We will consider a ***Boolean formula*** to be a string composed of three types of symbols: boolean variables ($x_1, x_2, x_3, \ldots$), parenthesis, and the logical operators AND ($\wedge$) and NOT ($\neg$). A string of these symbols is a boolean formula if it is is a boolean vari-

able, in which case it is called an atomic formula, or if it is of the form $(A \wedge B)$ or $\neg A$, where A and B are boolean formulas. The ***proper subformulas*** of a formula are defined as follows: the proper subformulas of $A \wedge B$ are A, B, and the proper subformulas of A and B; the proper subformulas of $\neg A$ are A and its proper subformulas; and atomic formulas have no proper subformulas. The ***subformulas*** of F are F along with all of the proper subformulas of F. Given an assignment of truth values (true and false) to the boolean variables of a formula, the truth value of the formula can be determined as follows: if the formula is a boolean variable, then the truth value of the formula is the same as the truth value of the variable; if the formula is of the form $(A \wedge B)$, then the formula is true if and only if both of the subformulas A and B are true; and if the formula is of the form $\neg A$ then it is true just if A is false. A boolean formula is satisfiable if and only if there is an assignment of truth values to its propositional variables that makes the whole formula true.

For every boolean formula F, we can define a corresponding diagram $D(F)$ which is satisfiable if and only if F is. Let $F_1, F_2, F_3, \ldots, F_{f-1}$ be the proper subformulae of F, arranged in order of increasing complexity, so that if F_j is a subformula of F_k then $j \leq k$, and let F_f be F. For each i, $0 \leq i \leq f+1$, we are going to define a subdiagram $D_i(F)$. $D(F)$ will be a diagram that contains disjoint copies of all of these subdiagrams. In order to construct $D(F)$, we first pick $6f + 8$ distinct markers: six di-angle markers a_i, b_i, g_i, e_i, h_i, and m_i for each subformula F_i of F; six other di-angle markers, labeled here by one slash mark and the names Y_1, Y_2, Z, H, and R; and two dseg markers, shown as two and three slash marks. Marker R will be used to mark right angles and will also be designated by drawing the usual right angle symbol in the diagrams. In the following discussion, the marker names will also be used to refer to the measures of the angles that they represent; it should be clear from context which meaning is intended.

We let $D_0(F)$ be the subdiagram shown in Fig. 6 and $D_{f+1}(F)$ be the subdiagram shown in Fig. 10. For $1 \leq i \leq f$, we define $D_i(F)$ as follows:

- If F_i is atomic, then $D_i(F)$ is the subdiagram shown in Fig. 7.
- If F_i is $\neg F_j$, then $D_i(F)$ is the subdiagram shown in Fig. 8.
- If F_i is $(F_j \wedge F_n)$, then $D_i(F)$ is the subdiagram that contains both of the subdiagrams shown in Figures 7 and 9.

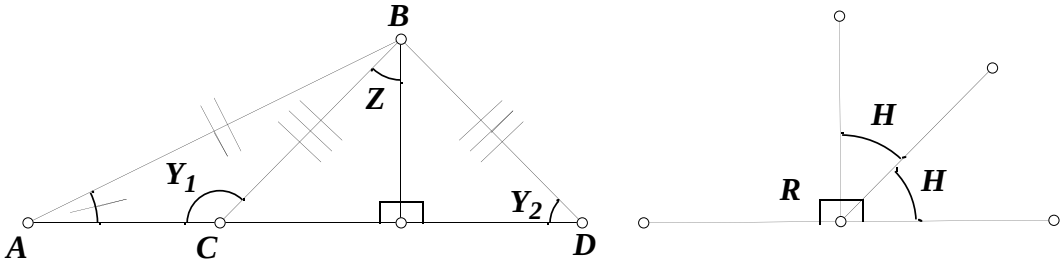

Fig. 6. $D_0(F)$

Let $D_i^\circ(F)$ be the (smallest) diagram containing $D_0, D_1, \ldots, D_i$ as disjoint subdiagrams, and let $D(F) = D_{f+1}^\circ(F)$.

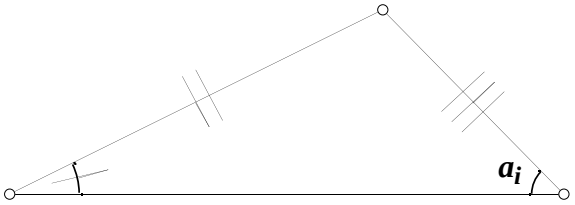

Fig. 7. Subdiagram contained in $D_i(F)$ if F_i is an atomic formula or a conjunction.

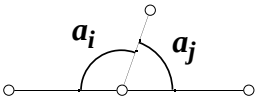

Fig. 8. $D_i(F)$ when F_i is $\neg F_j$.

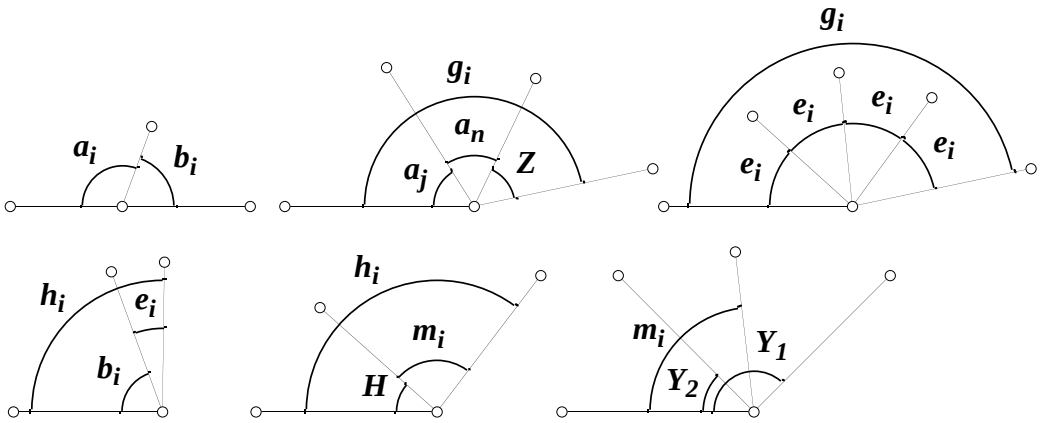

Fig. 9. Subdiagram contained in $D_i(F)$ if F_i is $(F_j \wedge F_n)$.

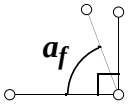

Fig. 10. $D_{f+1}(F)$

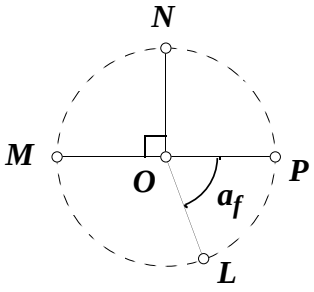

Fig. 11. $D''_{f+1}(F)$.

Note that if F has length n, then it has at most n subformulas, and the subdiagram put into $D(F)$ for each subformula has a size bounded by a constant, so the size of $D(F)$ is linear in the length of F. In fact, in order to compute $D(F)$ from F, the only thing that you need to keep track of is which subformula in F you are currently on, which you can do in log-space.

Now, note that Fig. 7 forces angle a_i to be equal to one of Y_1 or Y_2, since there are only two triangles that can be built with the given angle and sides. (For proof, see [1, p.304-7].) It follows by induction on the complexity of F_i that a_i must be equal either to Y_1 or else to Y_2, for every i: if F_i is atomic or a conjunction, then $D_i(F)$ contains Fig. 7, and if it is $\neg F_j$ then a_i is supplementary to a_j, which is one of Y_1 or Y_2 by inductive hypothesis, and so a_i is Y_2 or Y_1, since Y_1 and Y_2 are supplements. Furthermore, note that Y_2 and Z are complimentary, so $Y_2 = 90^\circ - Z$ and $Y_1 = 90^\circ + Z$.

We want to show these two possible values of each a_i correspond with the two possible truth values of F_i, so that $a_i < 90^\circ$ iff F_i is true. More specifically, we will say that a model M ***agrees*** with assignment t on a subformula F_i if $a_i = Y_2$ and F_i is true under t, or if $a_i = Y_1$ and F_i is false under t. We will show that any model of any of the D_i° that agrees with a truth assignment t on the atomic subformulas of F also agrees with t on all of the other subformulas. This means that the subdiagrams in Figures 8 and 9 act as logical NOT and AND gates, so that if F_i is $(F_j \wedge F_n)$, then $a_i < 90^\circ$ iff $a_j < 90^\circ$ and $a_n < 90^\circ$, and if F_i is $\neg F_j$, then $a_i < 90^\circ$ iff $a_j \geq 90^\circ$. This is shown in the following lemma:

Lemma 1. *Let t be a function assigning truth values to the boolean variables of F. Then:*

a. *For each $i \leq f$, there is a model M_i of $D_i^\circ(F)$ that agrees with t on all atomic subformulae F_k with $k \leq i$.*
b. *Furthermore, if M is any model of $D_i^\circ(F)$ that agrees with t on all atomic subformulas F_k with $k \leq i$, then it must also agree with t on all other subformulas F_k with $k \leq i$.*

Proof. by induction on i.

Base case: $i = 0$. In this case, we just have to show that $D_0^\circ(F)$, which is just $D_0(F)$, has a model. It has one: take any isosceles triangle, draw a perpendicular through the vertex, extend the base to one side, and connect it to the vertex to get a model of the first half of D_0. To get a model of the other half, bisect a straight angle into two right angles, then divide one of the right angles into two equal pieces.

Inductive cases:

1. F_i is an atomic formula of the form x_j. By the inductive hypothesis, $D_{i-1}^\circ(F)$ has a model M_{i-1} that agrees with t on all subformulas F_k such that $k < i$. That model satisfies all of $D_i^\circ(F)$ except for the triangle added by $D_i(F)$. Any triangle added by $D_i(F)$ will have to be congruent to one of the two triangles ABC or ABD; conversely, any model that extends M_{i-1} and contains a new disjoint triangle which is congruent to ABC or ABD will satisfy $D_i^\circ(F)$. So if $t(x_j) =$ true, let M_i be such an extension of M_{i-1} in which the new triangle is congruent to ABC, and otherwise let it be such an extension of M_{i-1} in which the new triangle is congruent

to ABD. Then M_i agrees with t on F_i, which shows part a. For part b, note that any model M of $D_i(F)$ that agrees with t on atomic formulas has a submodel that is a model of D_{i-1}; so by part b of the inductive hypothesis, $a_k = Y_2$ iff F_k is true under t if $k \leq i-1$, and this is also true if $k = i$, since F_i is atomic. This shows part b.

2. F_i is a formula of the form $\neg F_j$. By the inductive hypothesis, there is a model M_{i-1} of $D^\circ_{i-1}(F)$ that agrees with t on atomic formulae (by part a), and therefore agrees with t on all subformulae F_i with $k < i$ (by part b). Let M_i be a model extending M_{i-1} that also contains a new straight angle divided into two pieces such that the clockwise piece is congruent to the other angles that are marked by a_j. Then M_i is a model of $D_i(F)$, proving part a. To show part b, note that any model M of $D_i(F)$ that agrees with t on the atomic formulae of F must agree with t on all the subformulae F_k such that $k < i$, as before, so it suffices to show that it agrees with t on F_i, that is, that $a_i = Y_2$ in M iff F_i is true under t. Since j must be less than i (because the subformulae of F were arranged in order of increasing complexity), and a_i and a_j are supplements, $a_i = Y_2$ iff $a_j = Y_1$ iff F_j is false under t iff F_i is true under t. This proves b.
3. F_i is a formula of the form $(F_j \wedge F_n)$. We want to show that $D_i(F)$ forces a_i to be less than 90° iff a_j and a_n are both less than 90°. First note that in any model of $D_i(F)$, $m_i = (a_j + a_n + Z)/4 + b_i - 45^\circ$ (from pieces 2-5 of Fig. 9); $Y_1 < m_i < Y_2$ (from piece 6 of Fig. 9); and b_i is equal to either Y_1 or Y_2 ($90^\circ + Z$ or $90^\circ - Z$), because it is supplementary to a_i. There are three cases to consider:
   (a) $a_j = a_n = Y_2 = 90^\circ - Z$. Then $m_i = (180^\circ - Z)/4 + b_i - 45^\circ = 45^\circ - Z/4 + b_i - 45^\circ = b_i - Z/4$. Since $90^\circ - Z < m_i < 90^\circ + Z$, this means that $90^\circ - Z < b_i - Z/4 < 90^\circ + Z$, so $90^\circ - 3Z/4 < b_i < 90^\circ + 5Z/4$. Since b_i is either $90^\circ - Z$ or $90^\circ + Z$, this means that it must be $90 + Z = Y_1$. So since a_i is supplementary to b_i, this means that $a_i = Y_2$.
   (b) $a_j = Y_1$ and $a_n = Y_2$, or $a_j = Y_2$ and $a_n = Y_1$. Then $m_i = b_i + Z/4$; so b_i must be equal to Y_1 to keep m_i between Y_1 and Y_2; so $a_i = Y_2$.
   (c) $a_j = a_n = Y_2 = 90^\circ + Z$. Then $m_i = b_i + 3Z/4$, and b_i must be equal to Y_2 as before, so $a_i = Y_1$.

   So let M be any model of $D^\circ_i(F)$ that agrees with t on atomic formulas. By the inductive hypothesis, M agrees with t on all subformulae F_k with $k < i$. To show part b, it suffices to show that M agrees with t on F_i. Now, if F_i is true under t, then F_j and F_n must also be true under t (by the truth table for AND), so by the inductive hypothesis, $a_j = a_n = Y_2$, which is the first case above, so $a_i = Y_2$ in M, as required. On the other hand, if F_i is false under t, then one of F_j or F_n must also be false. So, by the inductive hypothesis, one or both of a_i and a_n must be equal to Y_2. So we are in either the second or third case above, and in both of these cases, $a_i = Y_1$ in M, as required. This proves part b. For part a, note that we can extend M_{i-1} with pieces satisfying $D_i(F)$ as long as we make a_i equal to Y_1 or Y_2 according to the three cases above, because the first five pieces of Fig. 9 serve only to define m_i, and the last piece will be satisfied as long as that m_i lies between Y_2 and Y_1.

This proves the lemma. □

We can now prove the following theorem:

Theorem 1. *Any boolean formula F is satisfiable if and only if $D(F)$ is satisfiable.*

Proof. ($\Rightarrow$) Assume that F is satisfiable. Then there is a truth assignment t of the boolean variables of F under with F is true. So, by the lemma, there is a model M of $D_f^\circ(F)$ that agrees with t on F_i for all $i \leq f$. In particular, M agrees with t on F_f. Since $F_f = F$ and F is true under t, this means that $a_f = Y_2$ in M. The only difference between $D_f^\circ(F)$ and $D(F)$ is that $D(F)$ contains D_{f+1}(F). So it suffices to show that there is an extension of M that satisfies $D_{f+1}(F)$. But $D_{f+1}(F)$ just says that $a_f < 90^\circ$. So since $a_f = Y_2 < 90^\circ$ in M, M can be extended to a model M' that satisfies $D(F)$.

($\Leftarrow$) Now assume that F is unsatisfiable. Then F is false under all possible truth assignments of its boolean variables. So, by the lemma, $a_f = Y_1 > 90^\circ$ in any model of $D_f^\circ(F)$, because any model of $D_f^\circ(F)$ agrees with some possible truth assignment on atomic formulae. So no model of $D_f^\circ(F)$ can be extended to a model of $D(F)$, since a_f would have to be less than 90° in any such model, since it would have to satisfy D_{f+1}. So $D(F)$ is unsatisfiable. □

We have shown how to reduce the question of whether or not a given boolean formula is satisfiable to the question of whether or not a given diagram is satisfiable, and this reduction can be done in log-space. Therefore, because the boolean satisfiability problem is NP-complete, we have the following corollary:

Corollary 1. *The diagram satisfiability problem is NP-hard.*

Next, consider the ***case analysis problem***: let D be a satisfiable primitive diagram, and let $S(D)$ be the set of satisfiable diagrams that can result from extending a given line segment in D outward until it intersects another segment. The problem of figuring out exactly what diagrams are in $S(D)$ is also NP-hard. To see this, let F be a boolean formula, let $D''_{f+1}(F)$ be the subdiagram shown in Fig. 11, and let $D''(F)$ be the smallest diagram containing $D_0(F), D_1(F), \ldots, D_f(F)$ and $D''_{f+1}(F)$. Then F is satisfiable iff $D''(F)$ has a model in which $a_f = Y_2$, iff $D''(F)$ has a satisfiable extension in which dseg OL is extended into di-angle MON. So since $D''(F)$ can also be produced from F in log-space, we also have:

Corollary 2. *The case analysis problem is also NP-hard.*

References

1. Euclid, *Elements*, T. L. Heath, ed., second edition, New York: Dover, 1956.
2. Luengo, Isabel,"A Diagrammatic Subsystem of Hilbert's Geometry", in *Logical Reasoning with Diagrams*, Gerard Allwein and Jon Barwise, eds., New York: Oxford University Press, 1996.
3. Shin, Sun-Joo, *The Logical Status of Diagrams*, Cambridge: Cambridge University Press, 1994.
4. Tarski, Alfred, *A Decision Method for Elementary Algebra and Geometry*, Berkeley: University of California Press, 1951.