Intelligent Dynamic Geometry Software*

I. Automated Geometric Diagram Generation

Xiao-Shan Gao, Wei-Qiang Yang and Lei-Dong Huang
Institute of Systems Science, Academia Sinica, Beijing 100080
E-mail: xgao@mmrc.iss.ac.cn

Abstract

Most dynamic geometry software uses construction sequences with ruler and compass as input. But, geometry diagrams in geometry textbooks are usually described declaratively, and the procedure of converting such a description to constructive form is usually done by the user. Our new software is trying to mechanize this tedious and sometimes difficult procedure, and is capable of drawing geometry diagrams automatically. The result is an intelligent dynamic geometry software which can be used to input and manipulate geometric diagrams more easily.

1 Introduction

With the invention of dynamic geometry software, noticeably, Gabri [10] and Geometer's Sketchpad [9], successful experiments have been done on using dynamic geometry software to education [2, 3, 4, 11, 12]. As compared with models built with real materials, visual models built with dynamic geometry software are more flexible, powerful, and more open for manipulation. These software represent a major step forward in modernizing the teaching of geometry in middle and high schools.

In [5], an automated reasoning software *Geometry Expert (GEX)* [7] is used to build *dynamic logic models*. Precisely speaking, with GEX we can prove and discover geometry theorems automatically. Logic models can be used for more intelligent educational tasks, such as automated generation of test problems, automated evaluation of students' answers, intelligent tutoring, etc.

In this paper, we will give a brief introduction to a new piece of software developed by us, which is capable of drawing geometry diagrams automatically. Previous dynamic geometry software uses construction sequences with ruler and compass as input. Most geometry diagrams

*This work was supported in part by an Outstanding Youth Grant (No. 69725002) from the Chinese NSF and by a National Key Basic Research Project (NO. G1998030600).

in geometry textbooks are described declaratively, and the task of converting such a description to constructive form is usually done by human. This makes the use of these methods cumbersome. For some diagrams, to find a constructive solution is quite difficult and many techniques were developed for ruler and compass construction since the time of ancient Greek. This new software is trying to mechanize some of the techniques of ruler and compass construction and to provide a new generation of dynamic geometry software, which can be used to input and manipulate diagrams more easily.

We apply theories of automated diagram construction to develop a piece of intelligent dynamic geometry software. This software accepts declarative description of geometry diagrams as input and draw the diagrams automatically. As a result, the software is much easy to use and still has all the elegant properties of the usual dynamic geometry software.

Using this software, we may generate about 80 percent of the 512 diagrams in [1]. Since the geometry theorems in [1] are described constructively, our software may provide a better interface for the geometry prover reported in [1].

2 Methods of Automated Geometric Diagram Construction

Let us say that we want to draw a parallelogram $ABCD$, i.e., a quadrilateral $ABCD$ such that $AB \parallel CD$ and $AD \parallel BC$(Figure 1). In most dynamic geometry software, the drawing process is using the following *construction sequence*:

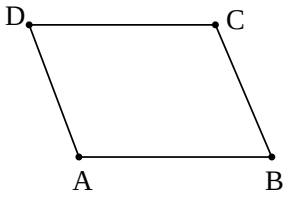


Figure 1: Drawing a Parallelogram

Take free points A, B, C.
Connect line segments AB and BC.
Draw a line l_1 which passes through C and parallel to AB.
Draw a line l_2 which passes through A and parallel to BC.
Take the intersection D of lines l_1 and l_2.
Connect line segments AD and CD.

We want to design a piece of software which can accept a *declarative description* of this diagram:

Take free points A, B, C, D.
Connect line segments AB, BC, AD and CD.
Add constraints $AB \parallel CD$ and $AD \parallel BC$.

Note that in the declarative description, we do not mention how to draw the diagram. The software will generate a construction sequence from this description and draw the diagram automatically. This function is called *automated geometric diagram construction*.

In what below, we will give a formal definition for the two ways of drawing diagrams. A *construction sequence with ruler and compass* for a diagram is a list of geometric objects in the diagram as follows:

$$O_1, O_2, \cdots, O_m$$

such that each O_i is introduced by the following basic constructions using objects already drawn $O_1, \cdots, O_{i-1}$.

Generally, a construction sequence is made up of the following sorts of constructions, which are used to introduce new geometric objects.

1. POINT(O): takes a free object O in the plane.
2. ON(O, t): takes a semi-free object O on t, where t could be a point, a line or a circle.
3. We also need the definitions for all geometric objects. For instance, LINE(O_1, O_2) is the line passing through point O_1 and O_2. CIR(O, d) is the circle with point O as the center and d as the radius.
4. INTER($O, O_1, O_2, \cdots, O_k$): O is the intersection of $O_1, O_2, \cdots, O_k (k \geq 2)$, details about the INTER construction can be found in the next section.

A *declarative description* of a diagram consists of two parts (OS, CS):

OS **is the Geometric Objects** including points, lines and circles.

CS **is the Geometric Constraints** which are given in the following table

Table 1. Geometric Constraint Relationships

	point	line	circle
point	distance	coincidence/distance	coincidence
line	coincidence/distance	parallel/angle	tangent
circle	coincidence	tangent	tangent

Remark We use DIS, ON and TANG to represent distance, coincidence and tangent constraints. There are three more often used constraints: $|AB| = |PQ|$, $\angle(ABC) = \angle(PQR)$, and midpoint$(M, A, B)$.

Algorithm 2.1 *The algorithm takes a declarative description of a geometric diagram* (OS, CS) *as input and generates a construction sequence* CT *for it if possible.*

For each object Q, let DEG(Q) be the number of constraints involving Q, and DOF(Q) the number of conditions needed to determine Q. For instance, for a point or a line, its DOF is two, while for a circle its DOF is three.

1. Let $CT = \emptyset$. For each object in OS, we assign a FLAG to it and the initial value of the FLAG is zero.

2. For all objects Q satisfying DEG(Q) $\leq$ DOF(Q), if the algorithm CONS(Q) returns a construction C, then add C to CT, add Q to the end of an ordered list l, and set the FLAG part of them to one. If l is empty, the algorithm terminates without finding a result.

3. Starting from the beginning to the end, for each R in l do the following: for each $U \in OS$, if such that there exists a constraint between R and U, DEG(U) $\leq$ DOF(U) and FLAG(U) = 0, and CONS(U) (see Algorithm 2.2) returns a construction C, then add C to CT, add U to the end of l, and set FLAG(U) to one.

4. If all objects in OS are added to l, then we find a construction sequence CT for the diagram. Otherwise, the algorithm fails.

This method works for simple diagrams only. If a diagram can be drawn with the above algorithm, we call it a *diagram without loops*. For algorithms to deal with diagrams with loops, please consult [8, 13, 6].

Algorithm 2.2 (CONS(O)) *The algorithm takes a geometric object O and the constraint set CS involving O as input and generates a construction for O.*

For convenience, we use the following representations for point P_i, line l_i, and circle O_i: $P_i = (x_i, y_i)$, where, x_i, y_i represent two variables; $l_i = (a_i, b_i, c_i)$, the relation equation is $a_i x + b_i y + c_i = 0$; $O_i = (x_i, y_i, r_i)$, where, the circle has center point (x_i, y_i) and radius r_i, so the relation equation is $(x - x_i)^2 + (y - y_i)^2 = r_i^2$.

So, the algorithm can be described as follows. Due to the limit on the length of the paper, we will omit the details for some of the cases.

CASE 1: $O(x, y)$ is a point (DOF=2). The constraint set could be:

1. $CS = \{P_0\}$: the distance of d_0 between O and P_0 is given. O is a semi-free point.
   Return On($O, CIR(P_0, d_0)$). So, x and y satisfy:

$$(x - x_0)^2 + (y - y_0)^2 = d_0^2. \tag{1}$$

2. $CS = \{l_0\}$: O is on the line l_0. O is a semi-free point.
   Return On(O, l_0). So, x and y satisfy:

$$a_0 x + b_0 y + c_0 = 0. \tag{2}$$

3. $CS = \{O_0\}$: O is on the circle O_0. O is a semi-free point.
   Return On(O, O_0). So, x and y satisfy:

$$(x - x_0)^2 + (y - y_0)^2 = r_0^2. \tag{3}$$

4. $CS = \{P_1, P_2\}$: d_1, d_2 are given as the distances between O and P_1, P_2. O is the intersection of two circles.
   Return INTER$(O, CIR(P_1, d_1), CIR(P_2, d_2))$. So, x and y satisfy:

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 = d_1^2 \\ (x - x_2)^2 + (y - y_2)^2 = d_2^2. \end{cases} \tag{4}$$

   The non-degenerate condition is:

$$P_1 \neq P_2, d_1 + d_2 \geq |P_1 P_2|. \tag{5}$$

5. $CS = \{P_1, l_1\}$: d_1 is the distance between point O and P_1, and O is the intersection of a line and a circle.
   Return INTER$(O, CIR(P_1, d_1), l_1)$. So, x and y satisfy:

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 = d_1^2 \\ a_1 x + b_1 y + c_1 = 0. \end{cases} \tag{6}$$

   The non-degenerate condition is:

$$\frac{|a_1 x_1 + b_1 y_1 + c|}{\sqrt{a_1^2 + b_1^2}} < d_1. \tag{7}$$

The details for cases $CS = \{P_1, O_2\}$, $CS = \{l_1, l_2\}$, $CS = \{l_1, O_1\}$, $CS = \{O_1, O_1\}$ are omitted.

CASE 2: O(a, b, c) is a **line** (DOF=2), the Constrain Set could be:

1. $CS = \{P_0\}$: O is a line passing through point P_0.
   Return On(O, P_0). So, (a, b, c) satisfy:

$$ax_0 + by_0 + c = 0. \tag{8}$$

2. $CS = \{l_0\}$: α is the angle between line O and line l_0. O is a semi-free line.
   Return On(O, l_0). So, x and y satisfy:

$$(ba_0 - ab_0) \cdot \cos\alpha = (aa_0 + bb_0) \cdot \sin\alpha. \tag{9}$$

3. $CS = \{O_0\}$: line O are tangent to circle O_0.
   Return On(O, O_0). So, x and y satisfy:

$$\frac{|ax_0 + by_0 + c|}{\sqrt{a^2 + b^2}} = r_0. \tag{10}$$

4. $CS = \{P_1, P_2\}$: O is the line passing through point P_1, P_2.
   Return LINE(P_1, P_2). So, (a, b, c) satisfy:

$$\begin{cases} a = y_2 - y_1 \\ b = x_1 - x_2 \\ c = x_1 y_2 - x_2 y_1. \end{cases} \tag{11}$$

   The non-degenerate condition is:

$$x_1 \neq x_2 \ \ and \ \ y_1 \neq y_2 \tag{12}$$

5. $CS = \{P_1, l_1\}$: O is the line passing through point P_1, and has the angle α with line l_1.
   Return ALINE(P_1, l_1, α). So, (a, b, c) satisfy:

$$\begin{cases} ax_1 + by_1 + c = 0 \\ (ba_1 - ab_1) \cdot \cos\alpha = (aa_1 + bb_1) \cdot \sin\alpha. \end{cases} \tag{13}$$

6. $CS = \{l_1, l_2\}$: unreasonable and no return in this situation.

Details for cases $CS = \{P_1, O_2\}$, $CS = \{l_1, O_1\}$, and $CS = \{O_1, O_2\}$ are omitted.

CASE 3: O (x, y, r) is a **circle** (DOF=3), the Constraint Set could be:

1. $CS = \{P_0\}$: O is the circle passes through P_0.
   Return ON(O, P_0). So, (x, y, r) satisfy:

$$(x - x_0)^2 + (y - y_0)^2 = r^2. \tag{14}$$

2. $CS = \{l_0\}$: circle O is tangent to line l_0.
   Return ON(O, l_0). So, (x, y, r) satisfy:

$$\frac{|a_0 x + b_0 y + c_0|}{\sqrt{a_0^2 + b_0^2}} = r. \tag{15}$$

3. $CS = \{O_0\}$: circle O is tangent to circle O_0.
   Return ON(O, O_0). So, (x, y, r) satisfy:

$$\sqrt{(x - x_0)^2 + (y - y_0)^2} = |r \pm r_0|. \tag{16}$$

4. $CS = \{P_1, P_2\}$: circle O passes through points P_1, P_2.
   Return INTER(O, P_1, P_2). So, (x, y, r) satisfy:

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 = r^2 \\ (x - x_2)^2 + (y - y_2)^2 = r^2. \end{cases} \tag{17}$$

5. $CS = \{P_1, l_1\}$: circle O passes through point P_1 and tangent to line l_1.
   Return INTER(O, P_1, l_1). So, (x, y, r) satisfy:

$$\begin{cases} (x - x_1)^2 + (y - y_1)^2 = r^2 \\ \frac{|a_1 x + b_1 y + c_1|}{\sqrt{a_1^2 + b_1^2}} = r. \end{cases} \tag{18}$$

Details for cases $CS = \{P_1, O_2\}$, $CS = \{l_1, l_2\}$, $CS = \{l_1, O_1\}$, $CS = \{O_1, O_2\}$ are omitted.

The cases $CS = \{P_1, P_2, P_3\}$, $CS = \{P_1, P_2, l_1\}$, $CS = \{P_1, P_2, O_1\}$, $CS = \{P_1, l_1, l_2\}$, $CS = \{P_1, l_1, O_1\}$, $CS = \{P_1, O_1, O_2\}$, $CS = \{l_1, l_2, l_3\}$, $CS = \{l_1, l_2, O_1\}$, $CS = \{l_1, O_1, O_2\}$, $CS = \{O_1, O_2, O_3\}$ are the ten Appollonius problems, and could be solved by the Gröbner basis method [8] and the rule based method [6].

3 Intelligent Dynamic Geometry

3.1 Intelligent Dynamic Geometry

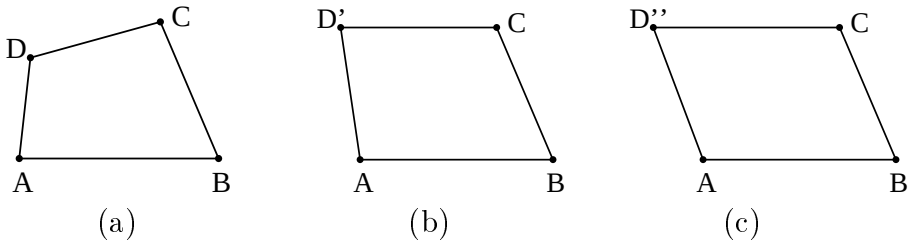
Comparing to the usual dynamic geometry, intelligent dynamic geometry has the following advantages: easy diagram generation and more flexible diagram manipulation.

Using the software to draw a diagram consists of three steps.

Drawing Sketch The user may use the mouse to draw a sketch of the diagram. Some topological properties such as coincidence and parallel, are also obtained in this step.

Generate Construction Sequence The user may add the geometric conditions, such as two lines are parallel or an angle has a certain value, to the sketch. For each geometric condition added, the software will automatically generate a construction sequence for the diagram with Algorithm 2.1.

Generate the Diagram The software will determine the positions of the geometric objects in the diagram according to the construction sequence and the sketch. This step is similar to the input procedure of the ordinary dynamic geometry software, but is done automatically according to some strategies.



(a) (b) (c)

Figure 2: Processes of drawing a parallelogram

Example 3.1 *We will show how to draw a parallelogram. First, we drag an arbitrary quadrilateral ABCD(Figure 2(a)). Next, we add a constraint AB ∥ CD. The software will redraw the diagram to make this constraint satisfied (Figure 2(b)). The corresponding construction sequence is*

$$\begin{cases} POINT(A), POINT(B), POINT(C), \\ l_1 = LINE(A,B), l_2 = LINE(B,C), \\ l_3 = PLINE(C,l_1), ON(D,l_3), \\ LINE(A,D), LINE(C,D). \end{cases} \quad (CS1)$$

We may similarly add another constraint AD ∥ BC. Now the the quadrilateral becomes a parallelogram (Figure 2(c)). The corresponding construction sequence is

$$\begin{cases} POINT(A), POINT(B), POINT(C), \\ l_1 = LINE(A,B), l_2 = LINE(B,C), \\ l_3 = PLINE(C,l_1), l_4 = PLINE(D,l_2), \\ INTER(D,l_3,l_4), LINE(A,D), LINE(C,D). \end{cases} \quad (CS2)$$

Using this software, we may generate about 80 percent of the 512 diagrams in [1].

3.2 Intelligent Dragging

Using methods of automated generation allows us to have more power to manipulation the diagram. Since the diagram is still drawn based on a construction sequence, the nice properties of dynamic geometry such as dynamic measurement, dynamic transformation and free dragging are still available. Furthermore, these properties are strengthened in the following way. If a construction sequence for a diagram has been given, we may only drag some points in the diagram. For instance, if using construction sequence (CS2) we cannot drag point D since it is the intersection of two lines. But in our case, this drawback may be fully overcomed as follows. Suppose that we want to drag point D in the parallelogram. Since the construction sequence is generated by the software, we may re-generate a new construction sequence in which D is free point.

$$\begin{cases} \text{POINT}(D), \text{POINT}(A), \text{POINT}(B), \\ l_1 = \text{LINE}(DA,), l_2 = \text{LINE}(A,B), \\ l_3 = \text{PLINE}(D,l_2), \\ l_4 = \text{PLINE}(B,l_1), \\ \text{INTER}(C,l_3,l_4), \\ \text{LINE}(C,D), \text{LINE}(C,B). \end{cases} \quad (CS3)$$

In this case, the user may drag point D freely.

3.3 Add Topological Relations Automatically

Geometric constraints are usually added during the second step of the drawing process. But, many topological constraints may be added automatically when the user draws the sketch.

The following topological relations are considered for this (1) A point is on a line or a circle. (2) Two lines are parallel or perpendicular. (3) A line is tangent to a circle. (4) Two circles are tangent to each other. Generally, we use mouse operations to add topological relations as follows: you need to choose relational objects and right-click the mouse to show a constraint dialog and fill in the constraint type. But it's often a very complicated and dull work when the geometry objects and constraints grow big. For convenience, the software allows the user to add topological relations when they drag a object. For example, when you drag a circle to change the radius, the circle might reach a position where it is tangent to the another circle. If the user drop the mouse at this time, this tangent relation is added as a constraint automatically.

3.4 Heuristics to Determine Objects with Multiple Solutions

In many cases, the constraints can not determine the geometric objects uniquely. In order to have a unique solution, we need some heuristics. Two typical cases lead to multiple solutions are to find intersection of circles and to draw semi-free objects.

We use the following two examples to show how to determine the intersection of circles.

Example 3.2 *Let point P be the intersection of a line $l_1 = LINE(A, B)$ and circle $c_1 = CIR(O, d)$. In order to have a unique and consistent solution, we assume that* $\cos(OPA)$ *is always positive or negative. This sign is first determined as follows: we choose the solution which is closer to the corresponding point in the sketch.*

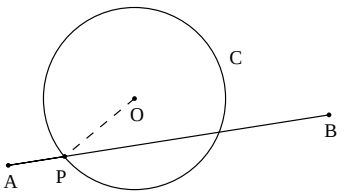


Figure 3. Intersections of a line and a circle

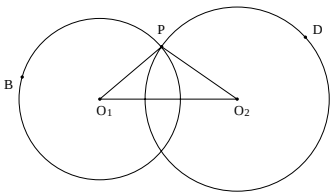


Figure 4. Intersections of two circles

Example 3.3 *P is the intersection of two circles (Figure 4). At the beginning, we choose the solution which is on the same side of O_1O_2 with the corresponding point in the sketch. Later, we will assume that the signed area PO_1O_2 will always be positive or negative.*

A semi-free object is an object whose constraints can't totally determine the objects' position. We usually have infinite number of solutions in this case.. Two typical cases are to draw a point on a circle or a line.

To draw a point P on a circle c with center O, let A be the position of Q in the sketch. Then line OQ has two intersections with circle c, among which we select the one nearest to Q as the solution to the intersection problem.

To draw a point P on a line l, let Q be the position of P in the sketch. We consider two cases. (1) If l contains a point O which is already drawn, then we form the intersections of line l and circle with center O and radius $|OQ|$ and select the one with less distance to Q as the solution. (2) If l contains no point which is already drawn, then the solution is the foot of the perpendicular drawn from Q to l.

References

[1] S.C. Chou, *Mechanical Geometry Theorem Proving*, D.Reidel Publishing Company, Dordrecht, Netherlands, 1988.

[2] J. Chuan, Geometric Constructions with the Computer, *Proc. of ATCM95*, pp.329-338.(1995)

[3] D. Dennis and J. Confrey, Functions of a Curve: Leibniz's Original Notion of Functions and Its Meaning for Parabola, *The College Mathematics Journal*, vol. 26, No. 3, pp. 124-131, 1995.

[4] X. S. Gao, C. C. Zhu, and Y. Huang, Building Dynamic Mathematical Models with Geometry Expert, I. Geometric Transformations, Functions and Plane Curves, *Proc. of the ATCM'98*, eds W.C. Yang, pp. 216-224, Springer, 1998.

[5] X. S. Gao, Building Dynamic Mathematical Models with Geometry Expert, III. A Geometry Deductive Database, *Proc. ASCM99*, W. Yang and D. Wang eds.,

[6] X. S. Gao and S. C. Chou, Solving Geometric Constraint Systems I. A Global Propagation Approach, *Computer Aided Design*, **30**(1), 47-54, 1998.

[7] X. S. Gao, J. Z. Zhang, and S. C. Chou, *Geometry Expert*, Nine Chapter Pub., 1998 (in Chinese).

[8] C. Hoffmann, Geometric Constraint Solving in R^2 and R^3, in *Computing in Euclidean Geometry*, D.Z.Du and F.Huang(eds), Word Scientific, 1995, pp. 266-298.

[9] N. Jakiw, *Geometer's Sketchpad*, User guide and Reference Manual, Key Curriculum Press, 1994.

[10] J.M. Laborde, GABRI Geometry II, Texas Instruments, Dallas, Texas, 1994.

[11] C.Z. Li and J.Z. Zhang, Readable Machine Solving in Geometry and ICAI Software MSG, in *Automated Geometry Deduction*, X.S. Gao and D. Wang (eds), Springer, pp. 67–85, 1999

[12] J. King and D. Schattschneider, *Geometry Turned On*, The Mathematical Association of America, 1997.

[13] J. Owen, Algebraic Solution for Geometry from Dimensional Constraints, in *ACM Symp., Found of Solid Modeling*A, CM PressA, ustin TX, 1991, pp. 397-407.