

# **Geometry Expressions Manual**

Saltire Software  
PO Box 1565, Beaverton, OR

# Contents

I. Solving Geometry Problems with <i>Geometry Expressions</i> .....	3
Constraints and Constructions: Two Ways to Work .....	4
How Does Geometry Expressions Solve Problems? .....	9
Calculating Output.....	21
Controlling Variables .....	26
Importing and Exporting Expressions.....	30
Performance Tips.....	31
II. <i>Geometry Expressions</i> Tutorials .....	33
Tutorial 1: Define and solve a problem.....	34
Tutorial 2: Create and manipulate a curve. ....	44



# I. Solving Geometry Problems with *Geometry Expressions*

Geometry Expressions™ is an interactive application that lets you specify geometry problems with symbolic constraints — for example, set a line to have length  $a$ , set an angle to be  $\theta$ , or make two lines perpendicular — rather than entering numeric values. The requested measurement is also output symbolically, as a mathematical expression.

After you've read about the basic ideas underlying Geometry Expressions, try running the *Geometry Expressions Tutorials* to help you get started.

## Constraints and Constructions: Two Ways to Work

Use Geometry Expressions to solve problems with a straightforward three-step process:

1. Draw geometric objects such as points, lines, or circles. You need not draw them exactly — Geometry Expressions will make the necessary adjustments as you work.
2. Constrain their relationships. As you specify constraints, the drawing adjusts to satisfy them. You can constrain the problem fully, or you can leave some elements unspecified.
3. Request a measurement. Geometry Expressions adds any missing variables required for the calculation, then outputs the requested expression or value.

It's not important to draw objects exactly at first; instead, the application corrects the drawing one step at a time as you define the problem by adding constraints. This constraint-based approach enables a flexible, exploratory work style. It also readily lends itself to defining a problem symbolically, as it allows you to specify constraints — distances, angles, slopes, etc. — in terms of symbols. Geometry Expressions provides a rich set of geometry objects and constraints, described in the [User Interface Reference](#) and embedded Help system.

You may be familiar with other interactive geometry applications, such as Geometer's SketchPad® or Cabri Geometry™, which use a different approach:

1. Draw independent geometric objects.
2. Define dependent objects using constructions.

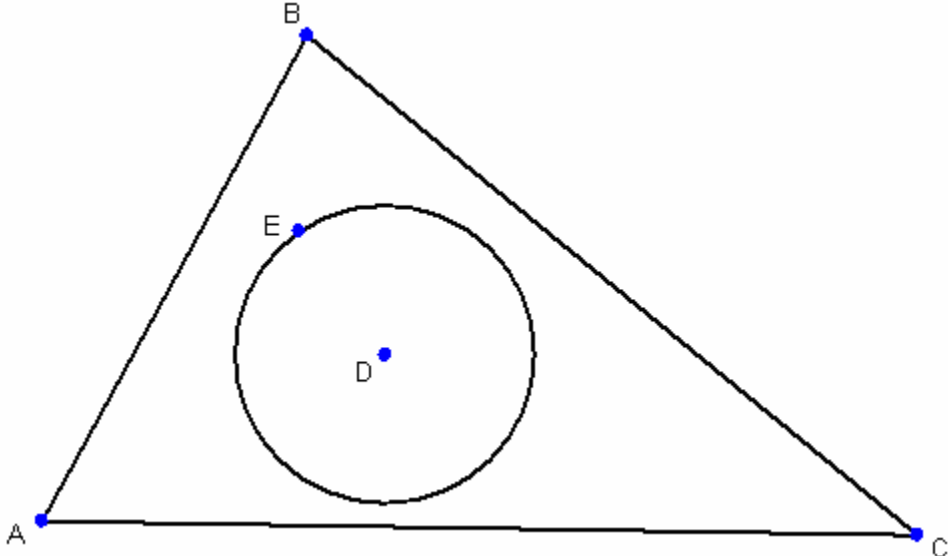
Geometry Expressions also enables this work style, providing a rich set of constructions, too (see the [User Interface Reference](#) and embedded Help system).

A **construction** defines a new geometry object in terms of existing objects. A construction-based approach requires you to distinguish between independent and dependent objects, and draw the independent objects accurately from the start. The drawing starts out correct and remains correct at each step. To enable this, the approach typically requires some foreknowledge of the problem geometry.

To compare these two approaches, we'll use both to create a drawing that shows the incircle of a triangle — the circle inside the triangle that's tangential to all three sides.

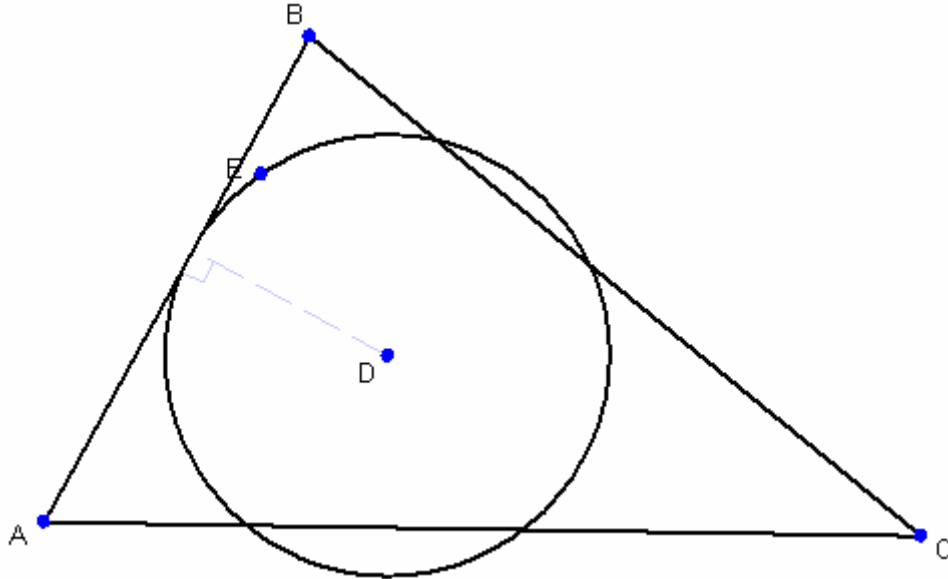
**Constraint-based Drawing**

A constraint-based approach starts with a drawing that contains all the pertinent geometrical objects, but does not at first require them to be correctly related (though the circle should lie more or less inside the triangle to distinguish it from the excircles):

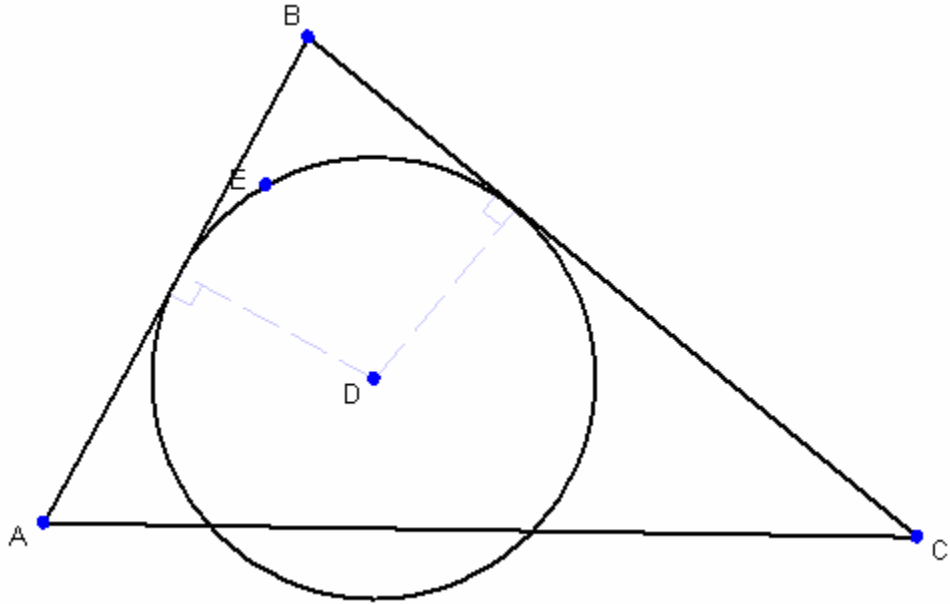


All we know of the incircle is that it is tangent to all three sides of the triangle. So we describe the problem by applying...

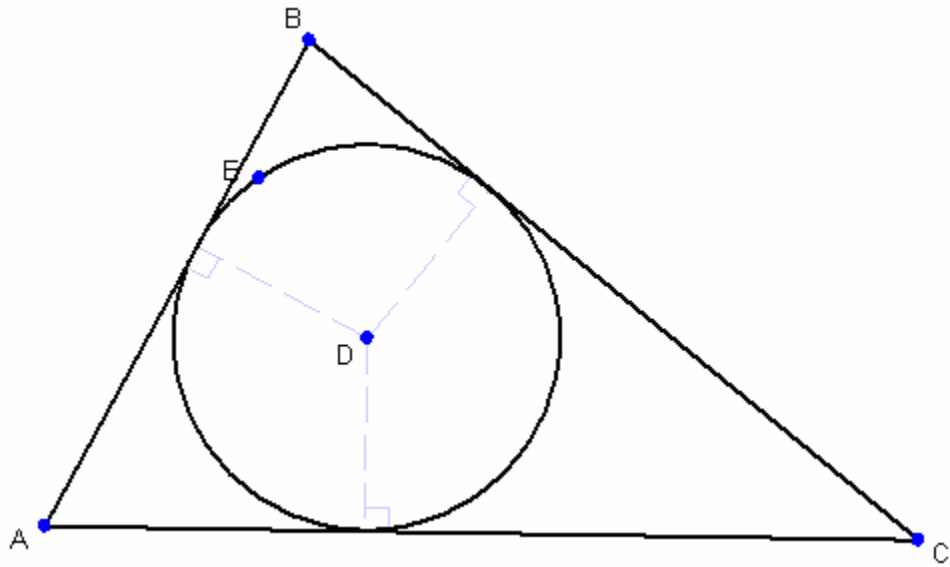
- 1. ...the first tangency constraint:



- 2. ...the second tangency constraint:



3. ...and the last:

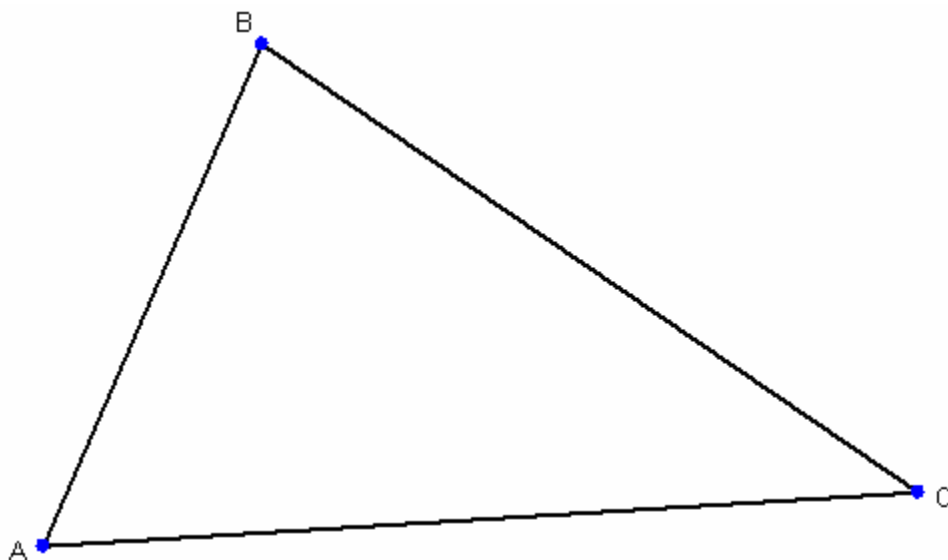


The drawing now accurately represents the desired situation.

### ***Construction-based Drawing***

By contrast, a construction-based approach builds an accurate drawing step-by-step, relying on previous knowledge: that the center of an incircle is the intersection of two angle bisectors of the triangle.

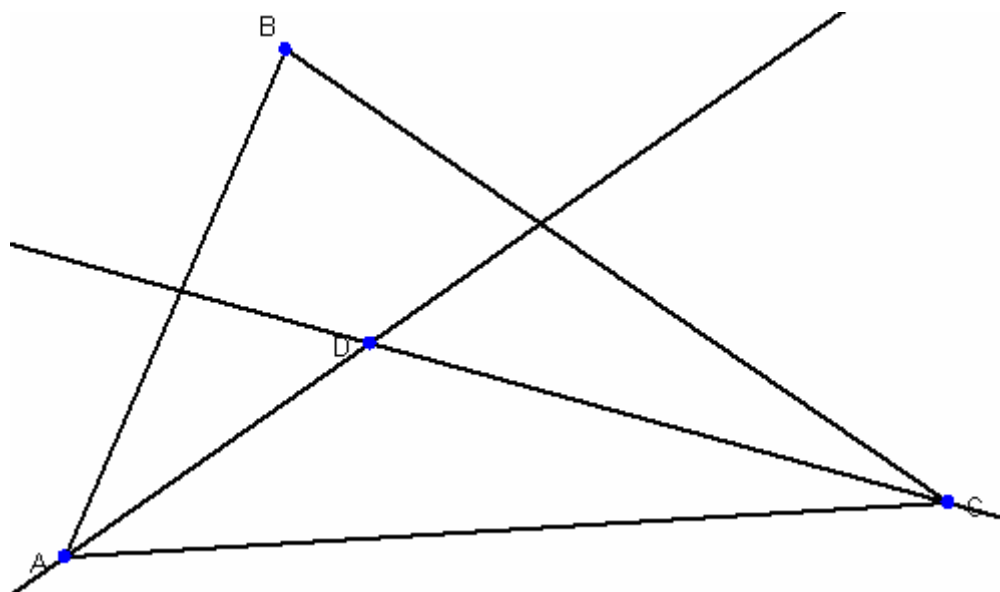
1. Draw the triangle:



2. Construct the angle bisector of one angle.

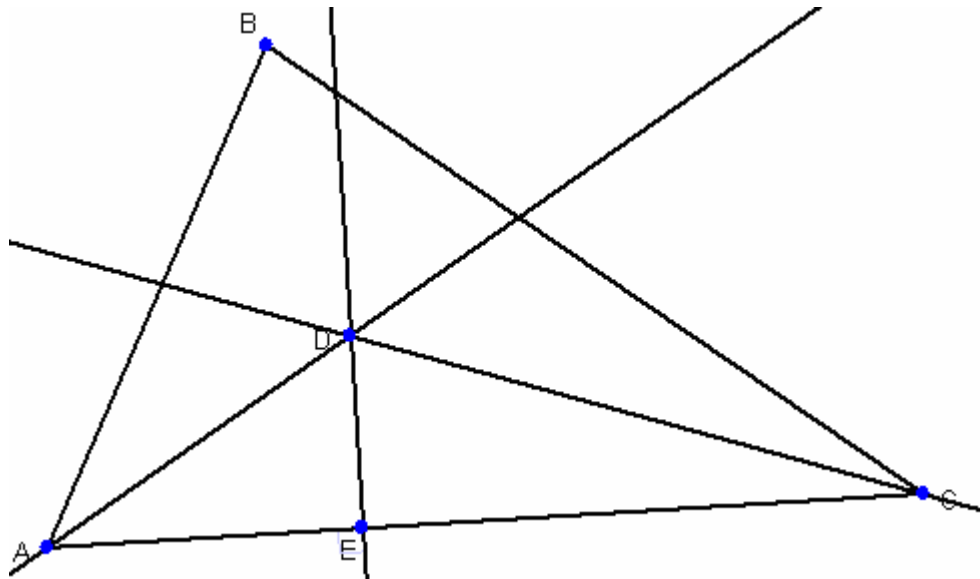
3. Construct the angle bisector of the second angle.

4. Construct the intersection point D of the two angle bisectors:

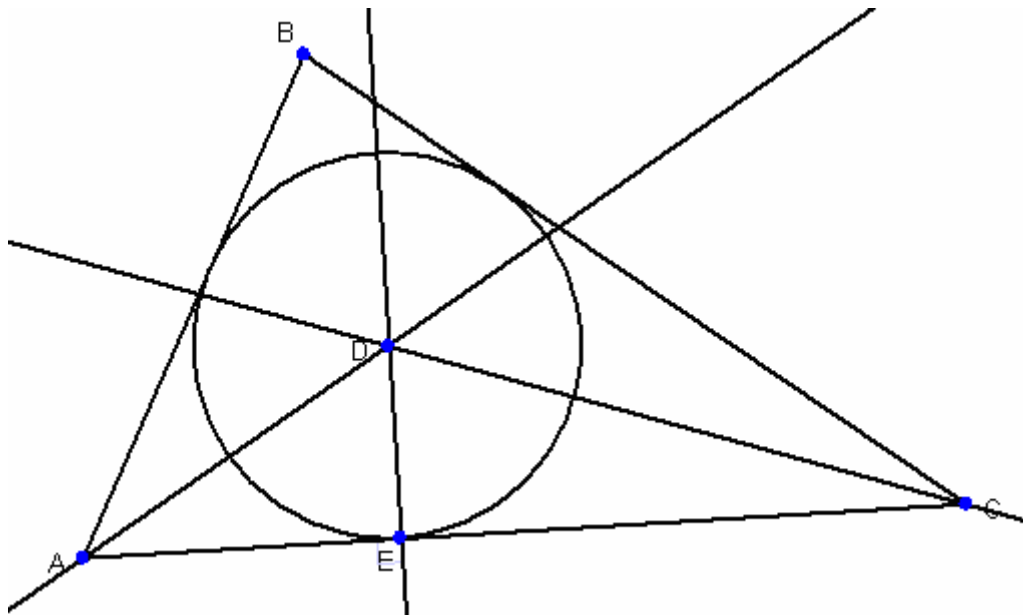




5. Construct a line perpendicular to AC through D. Construct the point E, where the new line intersects AC:



6. Draw a circle centered at D which passes through E:



It requires some geometric foreknowledge or ingenuity to come up with this construction, nor is it clear without a proof that the circle is indeed tangent to sides AB and BC.

Construction-based drawing is the approach used by other interactive geometry applications, and therefore you may be used to it. Constraint-based drawing, on the other hand, enables a more natural, exploratory style of problem solving. In practice, you'll probably use a combination of these two approaches, making the geometry much easier to create.

## How Does Geometry Expressions Solve Problems?

You can easily use Geometry Expressions without understanding anything about its internal process, but for various reasons, you'll find it helpful to understand how it builds the geometry problem and calculates the requested output. This understanding will make clear:

- how the application uses your drawings to resolve ambiguities;
- when and why the application adds variables;
- why it sometimes prompts you to resolve conflicts among constraints, and how to resolve these when they occur.

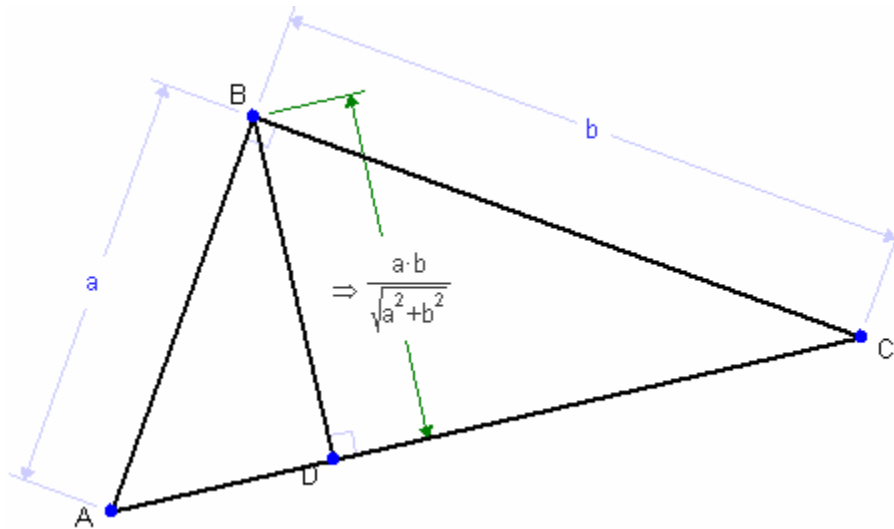
To shed light on these matters, we offer the following brief explanation of Geometry Expressions' internal process.

### ***The Internal Construction Sequence***

Though it's often natural and straightforward to describe a geometry problem in terms of constraints, internally, Geometry Expressions uses constructions.

1. As you specify geometric objects and constraints, it creates a construction sequence for them.
2. It executes those constructions to create a symbolic model of your geometry problem.
3. It creates a drawing that includes the objects and conforms to the constraints. To do so, it assigns sample numeric values for the variables.

For example, in the drawing below, AB is constrained to be length  $a$ , BC is constrained to be length  $b$ , AB is constrained to be perpendicular to BC, BD is constrained to be perpendicular to AC, and D is constrained to lie on AC.



Given this input, Geometry Expressions internally creates a construction sequence such as:

1. Place point A at an arbitrary location, creating variables to represent the coordinates of point A — by default,  $(u_0, v_0)$ .
2. Put point B distance  $a$  from A in an arbitrary direction, creating another variable to represent the slope of line AB — by default,  $\theta_0$ .
3. Calculate a sample value for  $a$  by measuring the distance of the line AB, using the native coordinate system. (Though the drawings we've shown so far do not show the axes, coordinates, or grid, you can reveal or hide them easily by clicking on the coordinate tool on the toolbar. See the [User Interface Reference](#) or embedded Help system.)
4. Create a line perpendicular to AB through B.
5. Put point C on this line at distance  $b$  from B.

Two points are possible at this distance — either to the left or to the right of the line AB. Geometry Expressions takes its cue from where you drew the line segment and placed the point. If the drawing shows C to the right of AB, Geometry Expressions determines that's what you intended.

6. Find a line perpendicular to AC through B.
7. Put D at the intersection of this line and AC.

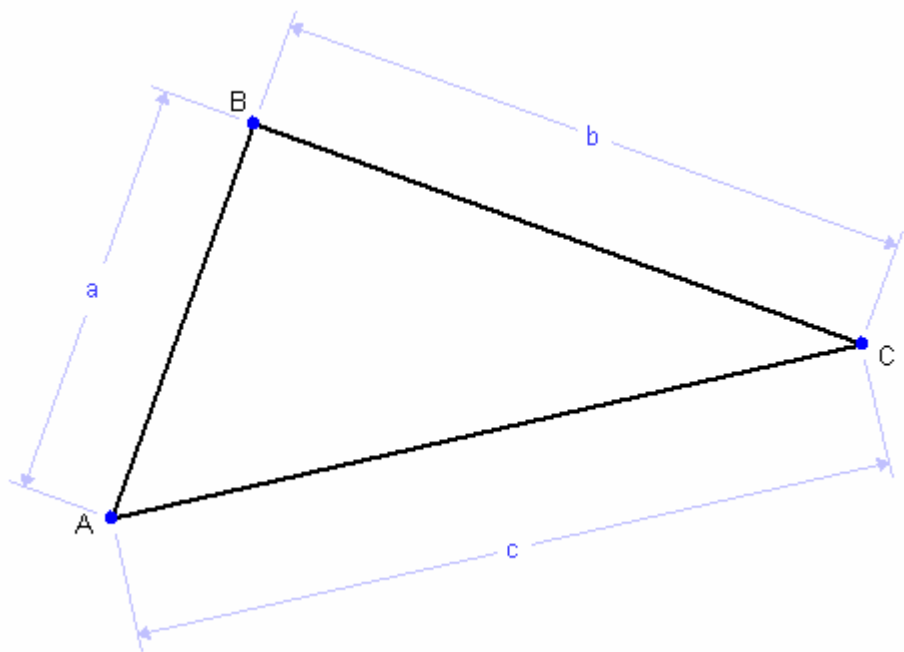
The existence of a construction sequence for Geometry Expressions is not equivalent to a problem being mathematically well defined. Unfortunately, Geometry Expressions' toolbox of constructions cannot cope with a number of mathematically well defined problems. Nevertheless, with a little ingenuity, you can usually find an alternative way to describe the problem, one that the application can construct.

Application-added variables, resolving geometrical ambiguities, and what happens when Geometry Expressions cannot find a construction sequence are all discussed in more detail below.

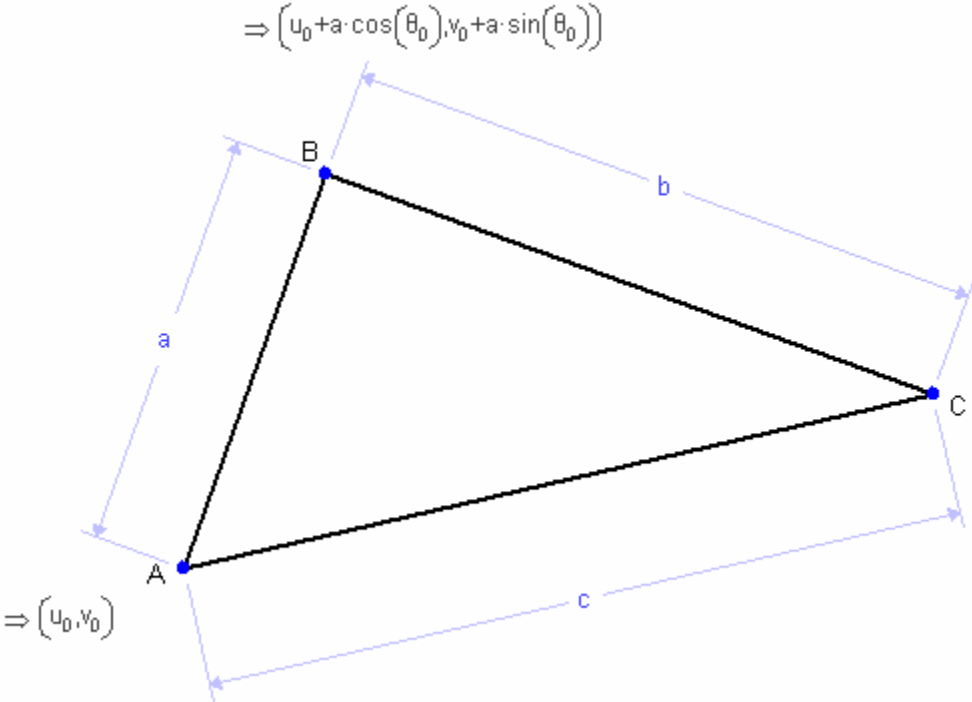
### ***Application-added Variables***

Geometry Expressions does not require you to specify your model in every detail. The underlying geometry engine automatically fills in any variables you've left unspecified.

For example, the following drawing (typical of many geometry problems) constrains the shape of the triangle but not its location:



Asking for the coordinates of points A and B, then, requires the system to add variables for the location of point A and the direction of line AB:

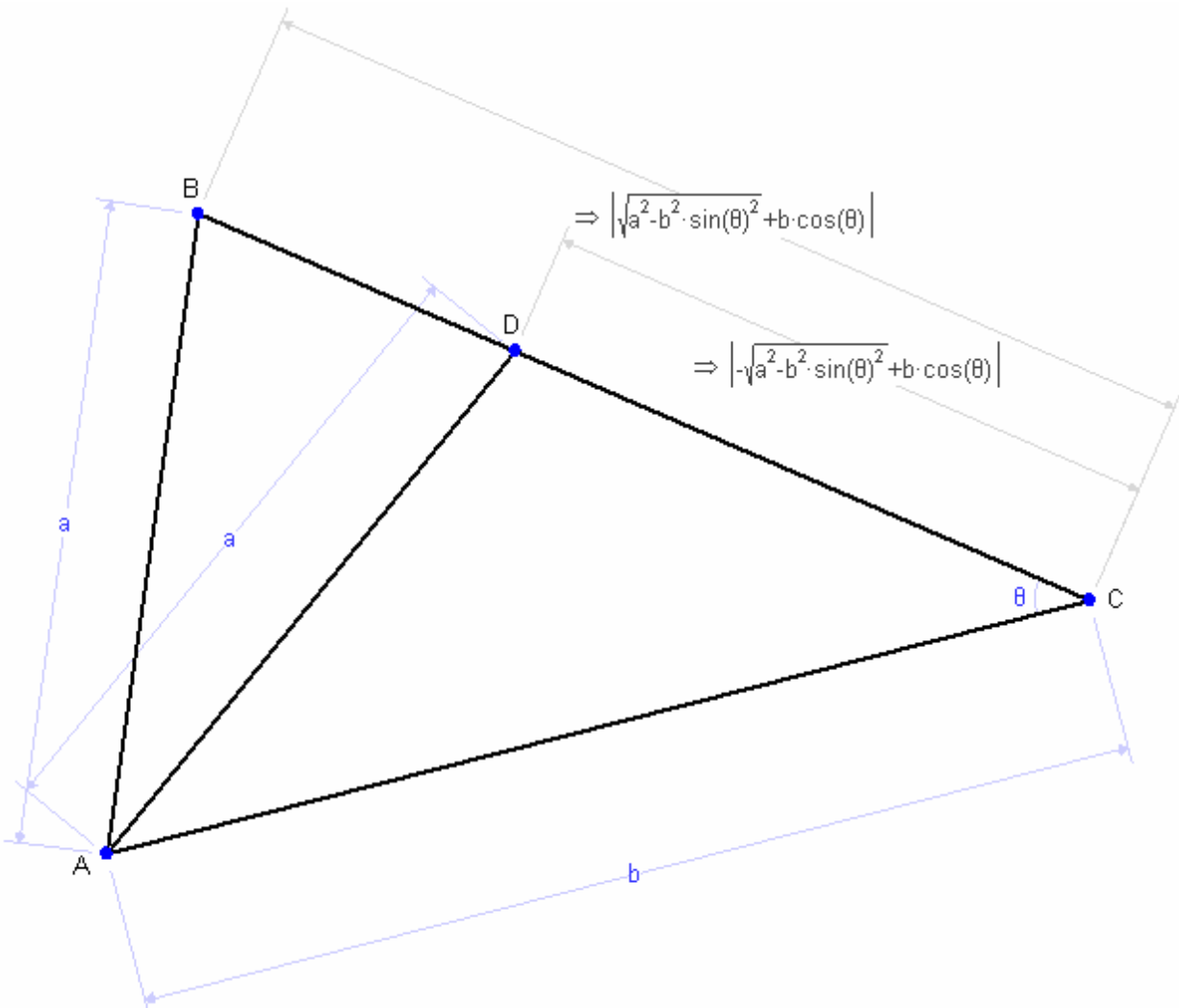


By default, Geometry Expressions creates the variables  $u_0$  and  $v_0$  for the  $x$  and  $y$  coordinates of point  $A$ , and  $\theta_0$  for the direction of the line  $AB$ . It associates numeric values with these variables based on the location of the pertinent object on the drawing's coordinate system.

Geometry Expressions cannot always calculate sample values so easily. When a variable is used in more than one place, or when it is involved in a complicated expression, the application solves equations to come up with appropriate sample values. However, whatever values it assigns, you can always set your own values directly in the Variables panel (discussed below, in "Controlling Variables", p. 26).

### ***Resolving Ambiguity***

Geometry Expressions uses your drawing to resolve ambiguities. For example, the following drawing shows two triangles,  $ABC$  and  $ADC$ . Both have two sides of constrained length, one of length  $a$  and the other of length  $b$ , as well as an angle  $\theta$ , an unincluded angle:

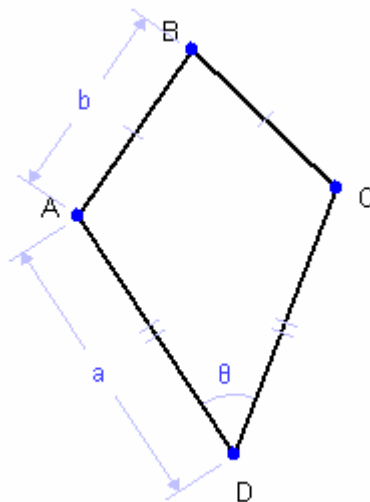


Two sides and the included angle uniquely define a triangle, but two sides and an unincluded angle do not. So if you specify a triangle using two sides and an unincluded angle, it's not unique: two possible solutions exist. In the drawing above, it could be either ABC or ADC.

In ambiguous cases like this, Geometry Expressions determines which solution you intend according to how you drew the objects.

Both triangles have one included angle (BAC and DAC), one unincluded yet constrained angle (ACB and ACD), and an unconstrained and unincluded angle (ABC and ADC). The key difference between the two triangles is whether this last angle is acute or obtuse. So the application consults your drawing to see how you drew this key angle. If you drew it as acute, the application constructs the triangle ABC; if you drew it as obtuse, it constructs ADC instead.

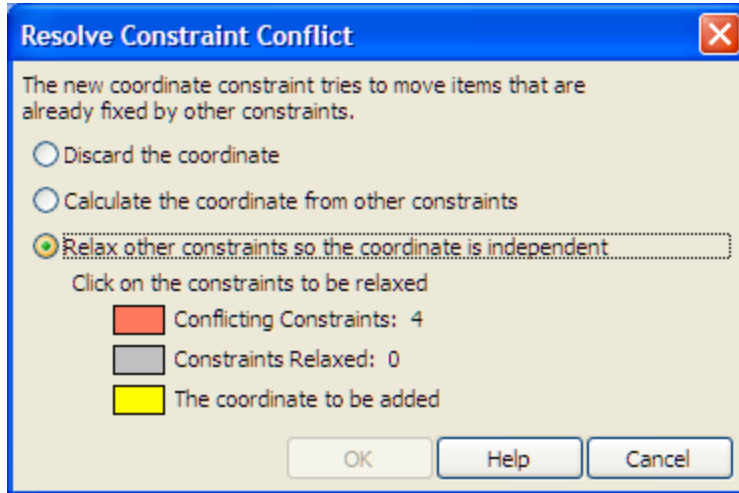
A trapezoid with the constraints specified below also has two possible solutions for the angle ABC — one convex, the other concave. Because the drawing is convex, Geometry Expressions chooses the convex solution:



### **Constraint Conflicts**

You might try to add more constraints than necessary, creating a situation in which the constraint you're trying to add conflicts with existing ones. In such cases, Geometry Expressions cannot find a construction sequence.

If you enter a constraint for a geometry object that is already constrained, you'll see a dialog such as this:



You can resolve constraint conflicts in any of three ways:

- Cancel the operation to leave the drawing as it was without the new constraint.
- “Calculate [new constraint] from other constraints.” Like canceling, this choice eliminates the new constraint. Unlike canceling, however, Geometry Expressions also calculates the selected geometry's value and displays the result.
- To add the new constraint, you must relax one of the conflicting ones. Choosing the bottom radio button affords the opportunity to choose which; all the conflicting constraints are highlighted in red, the constraint you tried to add is highlighted in yellow (*Figure 1*).

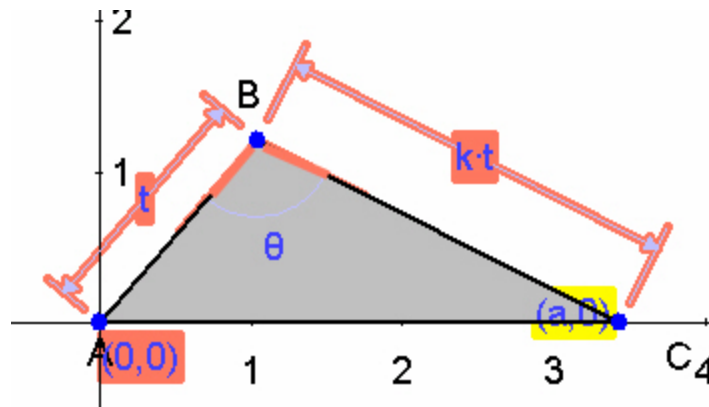


Figure 1

When you select one of these constraints (*Figure 2* shows the result of clicking on  $\theta$ ), the highlight of the current constraint changes to green.

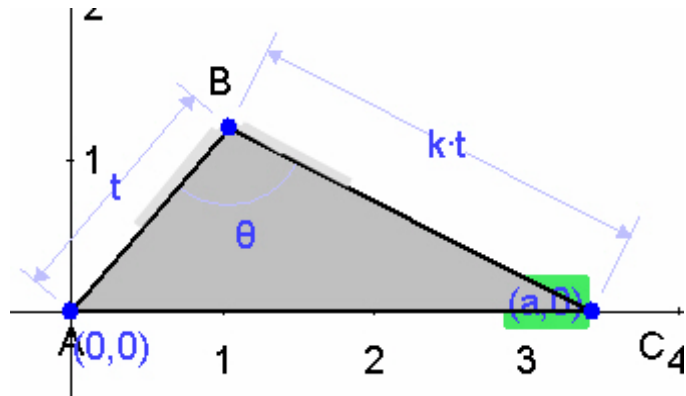


Figure 2

**OK** adds your new constraint, then calculates and displays the new value of the variable associated with the relaxed constraint (Figure 3).

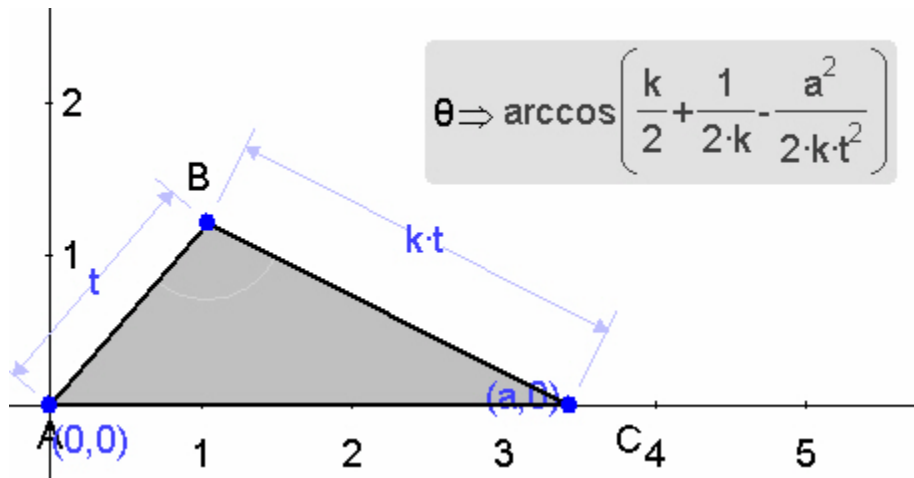


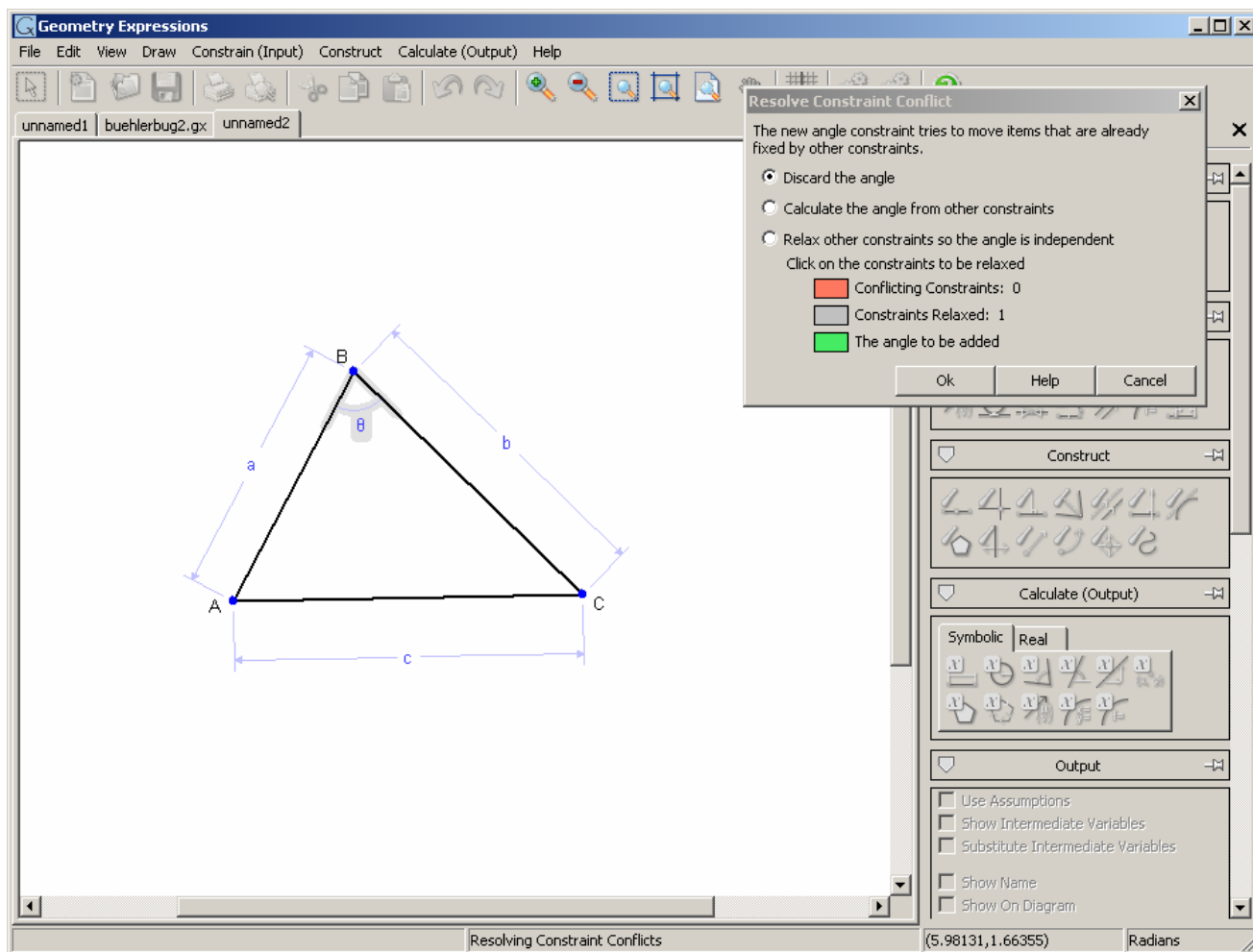
Figure 3

You'll need to resolve constraints conflicts whenever you add a new constraint that is:

- not independent of those already added;
- not generically applicable, though possibly applicable in specific cases; or
- for which a construction sequence cannot be derived.

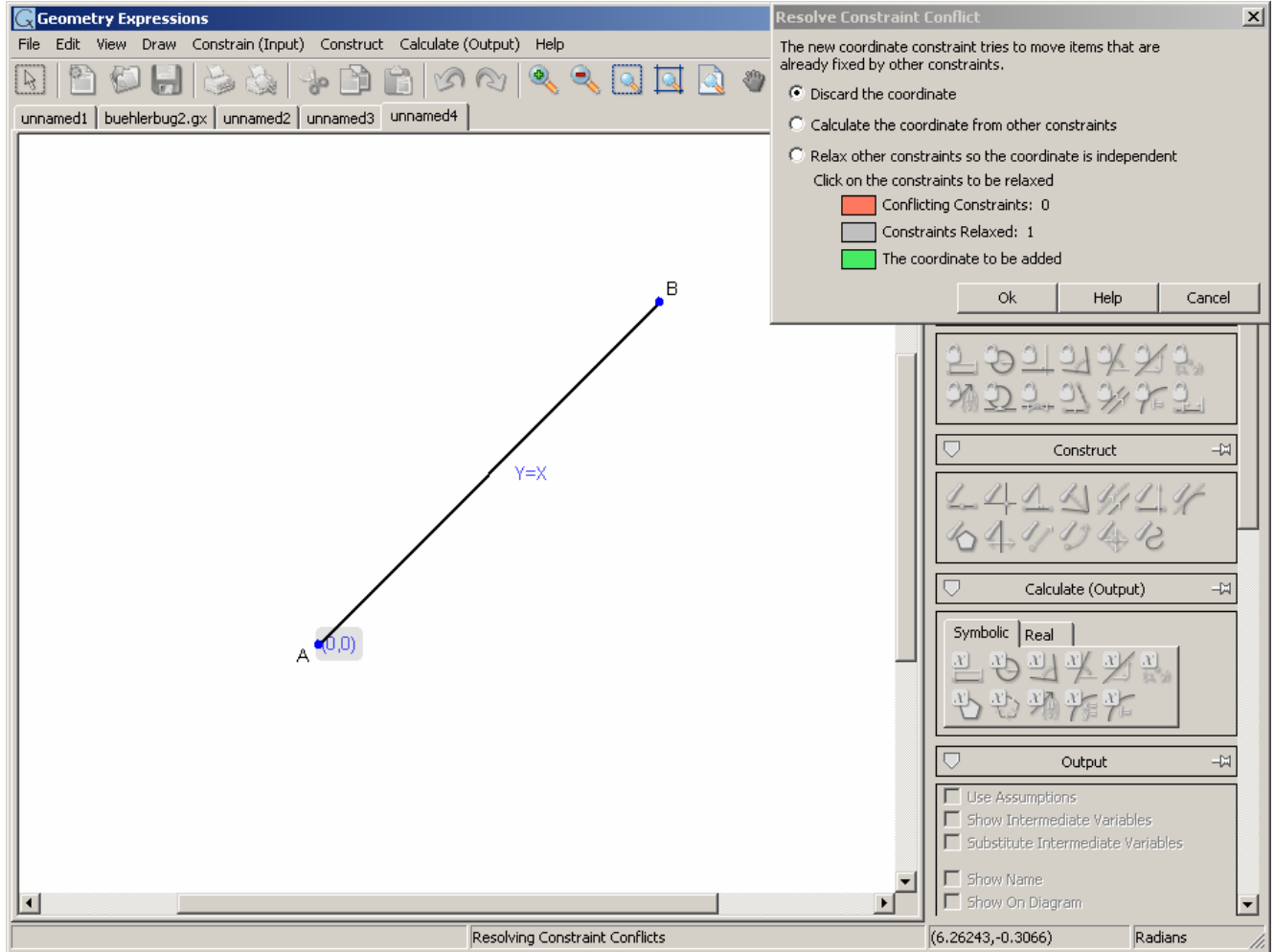


Below is an example of a constraint that isn't independent: three sides already define the triangle, so  $\theta$  is dependent on the side lengths.

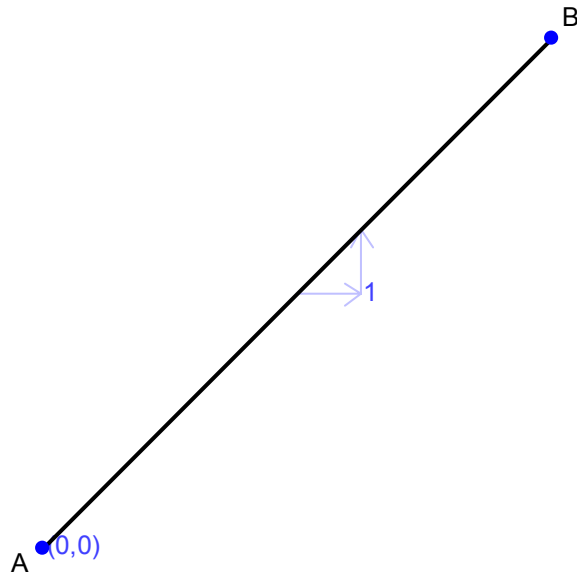


To resolve this conflict, you can eliminate one of the constraints — either one of the problematic side lengths, or  $\theta$ .

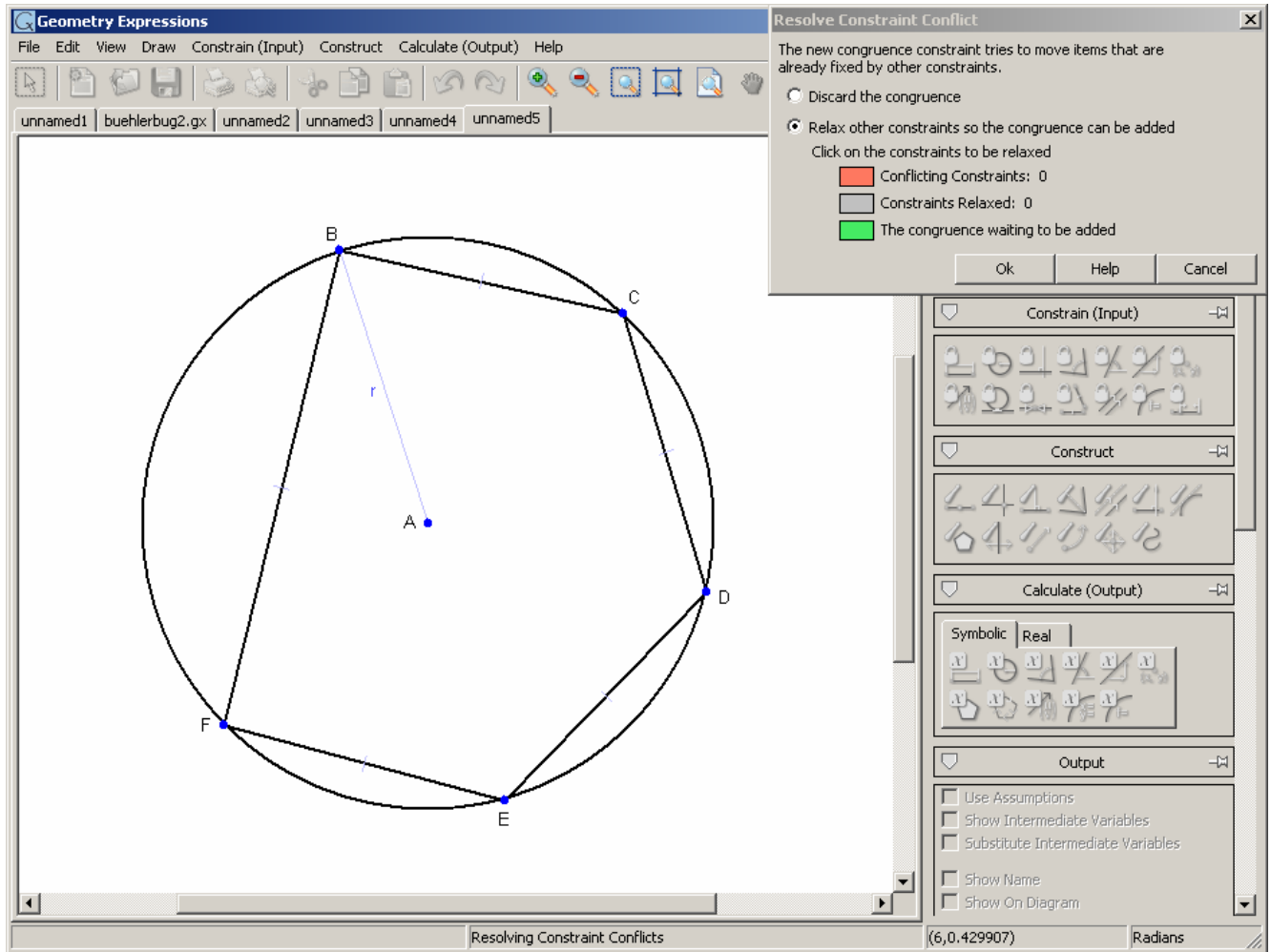
Below is an example of a constraint that, while it appears to be independent, would not be so in all cases. The line segment has been assigned the equation  $Y=X$ , and the point A has been assigned the coordinates  $(0,0)$ . The point  $(0,0)$  does indeed lie on the line  $Y=X$ , but other values for its coordinates might not. For example, if you tried to constrain A to be at coordinates  $(1,0)$ , the  $Y=X$  constraint would be violated. Constraints in Geometry Expressions must be applicable generically, rather than relying on specific values.



To resolve this conflict, instead of setting the equation of the line, you could set its slope:



Some problems seem to violate neither of these rules; nevertheless, Geometry Expressions can't find a construction sequence that satisfies them. For example, suppose you're trying to determine the length of the side of a regular pentagon inscribed in a circle of radius  $r$ . You might try to draw the pentagon by constraining its sides to be of equal length:



The constraints are certainly independent, so why the prompt to resolve a conflict? The answer lies in the details of how Geometry Expressions converts the constraint description into a construction sequence.

A construction sequence is a sequence of primitive geometrical operations, each of which creates a single geometric object. For example:

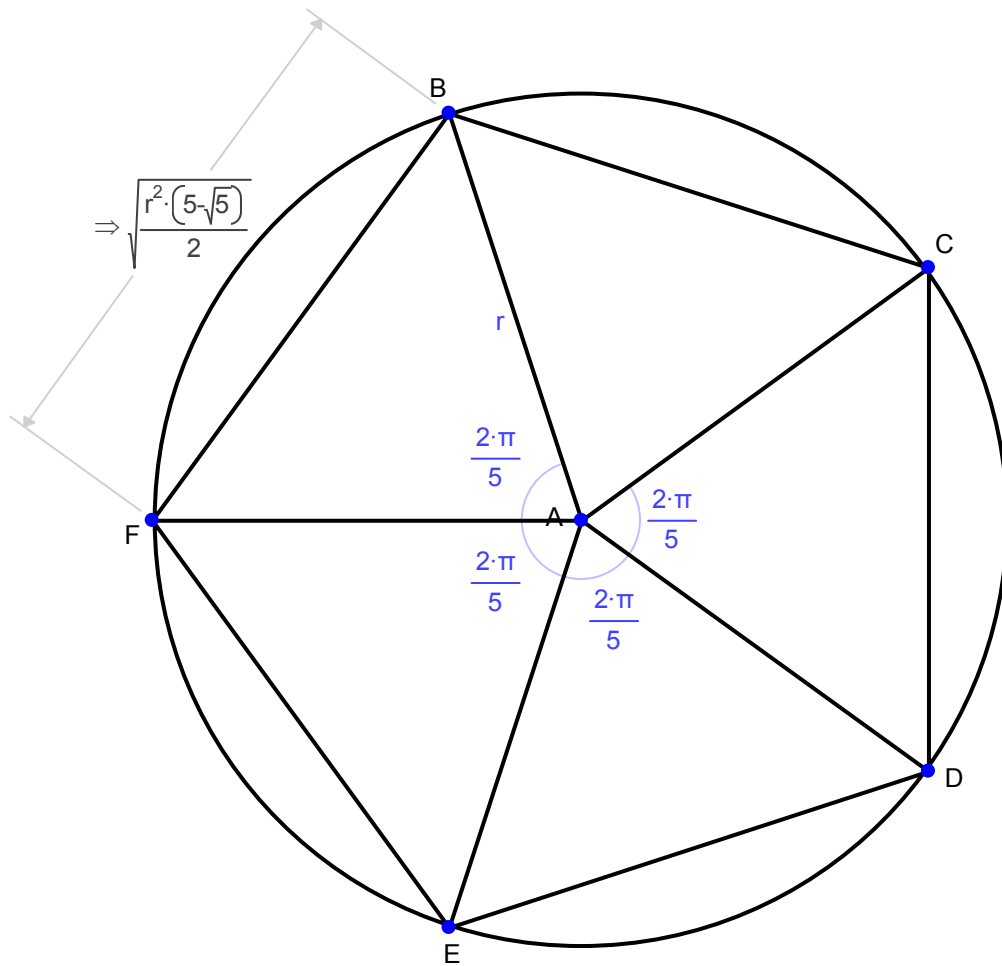
Create a point on a given line, distance  $a$  from a given point.

or:

Create a circle tangent to three given circles.

To draw the specified pentagon, Geometry Expressions starts with a circle of a particular radius. It then constructs a point on the circumference at an arbitrary location. Next, it must construct another point, this time one whose location is not arbitrary. But calculating the point's position would require simultaneously resolving all the congruent distance constraints. Geometry Expressions can't resolve that many constraints at once; it can resolve them only in batches of two or three, with each batch defining just one geometrical object. So Geometry Expressions is unable to create the construction sequence.

To solve this problem, you need to try a different approach. For example, specify the angles from the center of the circle, which enables Geometry Expressions to construct the points one by one:



## Calculating Output

Geometry Expressions calculates results based on the objects as you've drawn them, and the constraints or constructions you've specified. If you haven't supplied all of the necessary constraints, the system adds any missing variables automatically. (For an example, see "Application-added variables", p. 10.)

You can ask for calculations in either numeric or symbolic terms by choosing the appropriate tab (**Real** or **Symbolic**) when you request the output. Symbolic calculations yield results as mathematical expressions.

Geometry Expressions can present the results of its calculations in different ways, depending on your output settings. Two in particular require discussion—whether you wish to see absolute values for certain terms, and how you wish to treat intermediate variables.

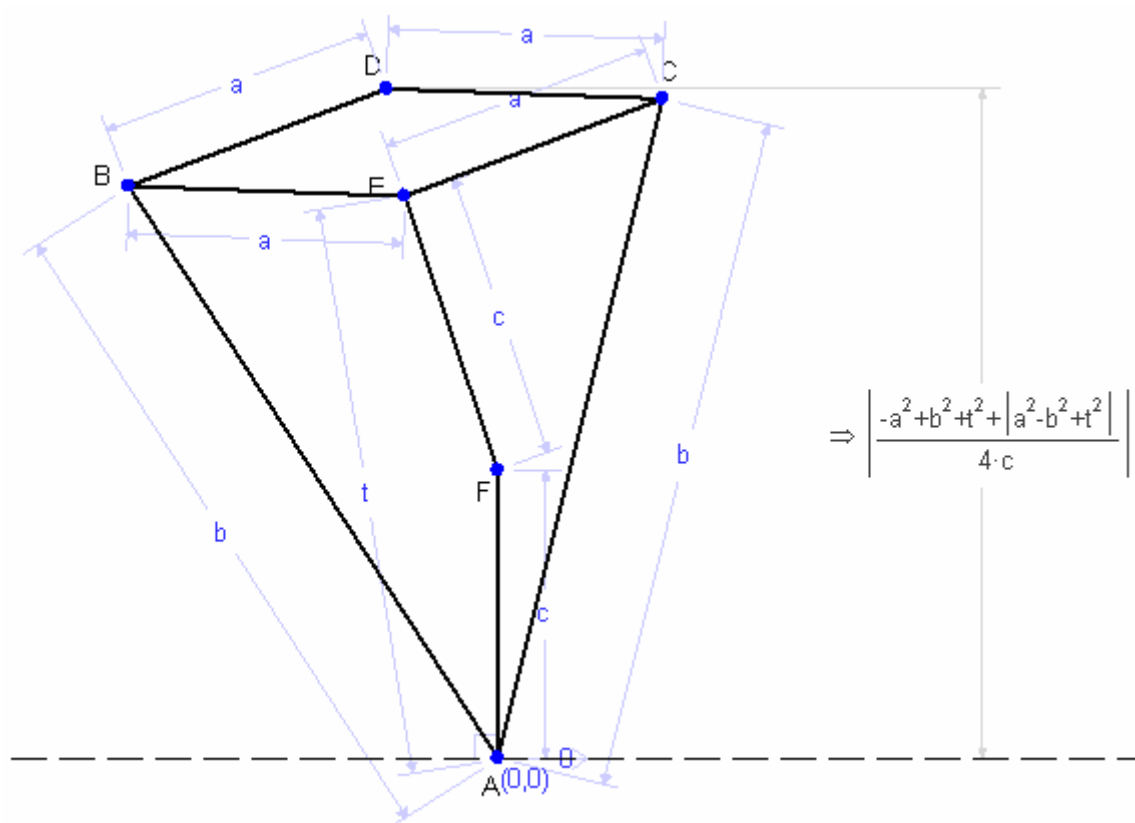
### *Eliminating Absolute Values*

**Use Assumptions** is a checkbox in the Output panel that determines whether expressions include or eliminate absolute values: the assumption in question is whether the operand of the absolute value is positive or negative.

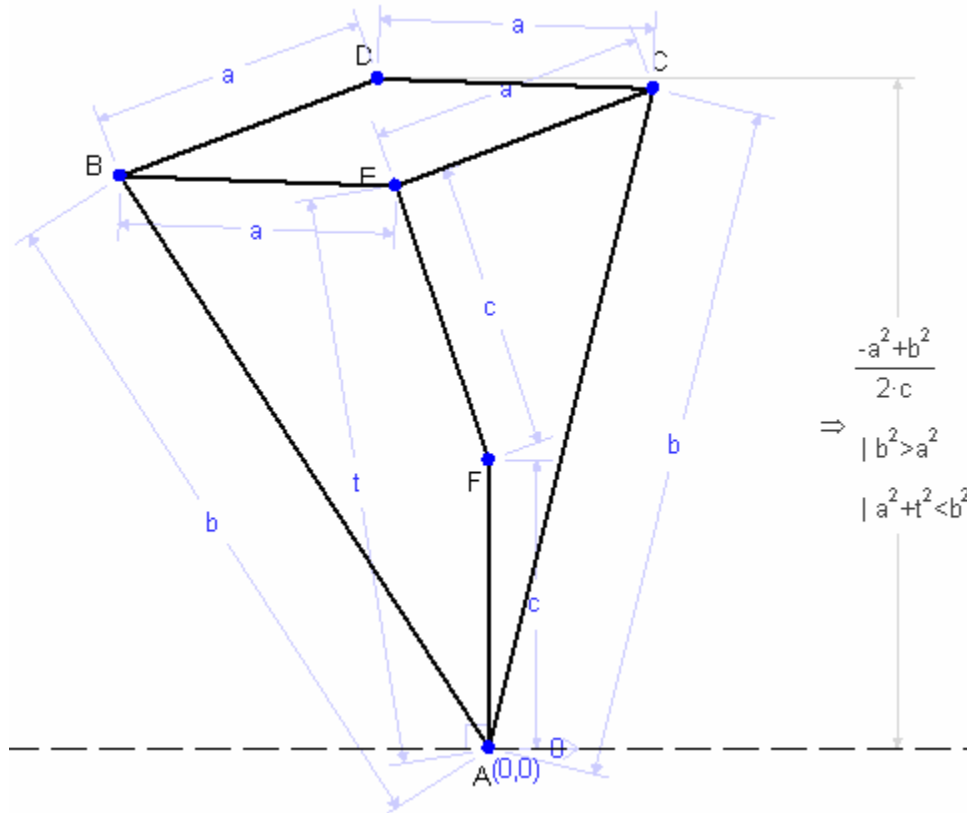
When checked, the application assumes that a given term is positive, if the value of the absolute value operand is positive. If it's negative, the corresponding term is assumed to be negative.

When unchecked, pertinent terms appear in expressions enclosed in absolute value bars.

For example, without assumptions, in this drawing of Paucellier's linkage, the height of D depends on parameter  $t$ :



If you check **Use Assumptions**, the expression simplifies and no longer involves  $t$ . However, other assumptions can sometimes be added, as shown below. Check these assumptions against the geometry of the diagram to see if they accurately express your intentions.



The second assumption in the drawing above is equivalent to assuming that angle AEB is obtuse; given the geometry of the drawing, is reasonable.

Assumptions are a means to simplify expressions and apply only to absolute values. If an expression doesn't include any terms in absolute values, the checkbox will have no effect.

### ***Using Intermediate Variables***

As described in "Calculating Output" above, the application expresses results in terms of:

- variables you've added, and
- additional variables the application added, if you haven't fully specified the drawing.

However, expressions can sometimes be so large and complex as to be difficult to read and understand. To make output expressions clearer and more compact, you can choose to use intermediate variables.

An *intermediate variable* is a variable defined in the same terms — your own variables plus those the system added — for the purpose of simplifying an expression and enhancing its readability. Not every expression needs or can benefit from intermediate variables, but for some expressions, their use allows one straightforward name to replace a significant portion of the expression.



You can control how Geometry Expressions handles intermediate variables in three ways:

- An **Intermediate Variable Complexity** setting determines how complex an intermediate variable must be before it appears in an expression.
- A **Use Intermediate Variables** checkbox determines whether, above this threshold, they appear or not.
- A **Show Intermediate Variables** checkbox affects their display.

These three settings interact with each other to let you control the calculations that Geometry Expression performs and how it presents them.

**Intermediate Variable Complexity** is an application setting, available from **Edit > Settings**. From the **Math** tab, this option allows you to set a threshold for intermediate variable complexity in terms of a number from 2 to 100. This number determines the complexity of the term to be replaced with a variable in an expression.

A low number tells the application to replace even simple terms with intermediate variables, so you'll see more of these variables. A progressively higher number means that intermediate variables will represent progressively larger and more complex terms in expressions, so you'll see fewer of them.

**Use Intermediate Variables**, a checkbox in the Output panel, determines whether intermediate variables above the complexity threshold are used or not. When checked, terms more complex than the threshold will be replaced by intermediate variables in expressions. Terms simpler than the threshold are not replaced.

When unchecked, intermediate variables never appear and the threshold has no effect.

**Show Intermediate Variables**, also a checkbox in the Output panel, determines whether intermediate variable definitions are displayed with results. When checked, you'll see the results, including intermediate variables, followed by additional lines displaying the definition of each intermediate variable used.

When unchecked, you'll see intermediate variables in expressions according to the other two settings, but not their definitions.

**Show Intermediate Variables** affects only how an expression is displayed; it has no effect on the underlying expression itself. And it's relevant only when **Use Intermediate Variables** is checked. When **Use Intermediate Variables** is unchecked, there are no intermediate variables to display.

To summarize:

- **Intermediate Variable Complexity** sets a threshold above which intermediate variables are used to calculate expressions, and appear in them. They thus affect the details of the expression that's calculated, as well as affecting the expression's readability.
- **Use Intermediate Variables** determines whether to use intermediate variables above the threshold, or not. This also affects the details of expressions.
- **Show Intermediate Variables** determines whether to display the definitions of intermediate variables when they are present.

# Controlling Variables

As you create geometry objects and add constraints or constructions, Geometry Expressions creates the required variables; assigns numeric values according to the constraints, constructions, and drawing; and lists any variables you create along with their values in the Variables panel.

The numeric value assigned to a variable is like a sample value; it does not replace the symbolic value — the expression. Instead, the application uses the numeric value for four purposes:

- to determine where to draw the geometric objects on the page,
- to resolve ambiguous geometry,
- to determine whether to substitute a positive or negative value for an absolute value when using assumptions, and
- to take numeric measurements from the drawing.

You don't have to accept the sample values assigned by the application. You can control the values of variables various ways:

- Assign a value directly.
- Lock a variable to its current value, so that it won't change when you drag geometry objects in the drawing.
- Assign a range of values to the variable for creating an animation.

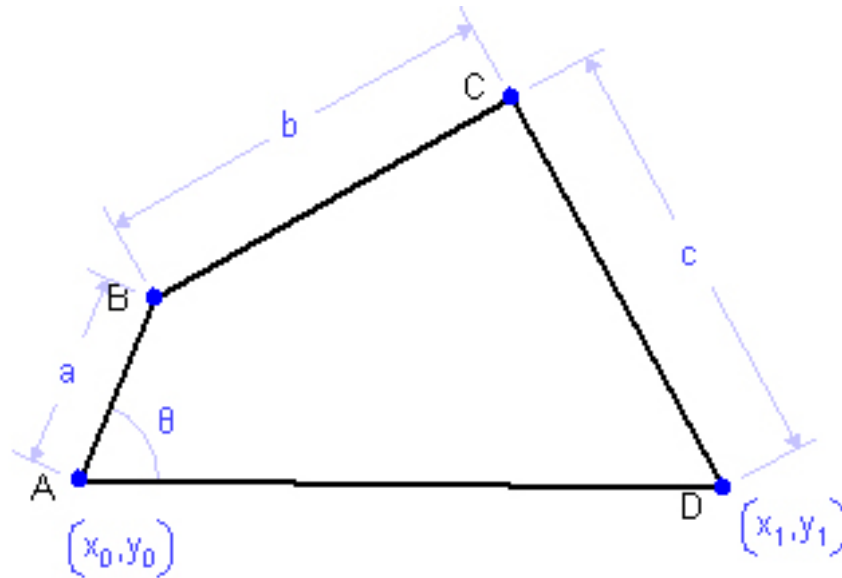
These capabilities are discussed below.

## ***Assignment***

If you wish a given variable to have a specific value, you can assign it directly by selecting the variable in the panel and entering the desired value in the input field. (See the [User Interface Reference](#) or embedded Help system.) If you assign a value that results in a geometry that the application cannot construct, the problematic objects in the drawing will disappear. However, they're not gone; they'll reappear when you set the variable to a value consistent with the rest of the drawing.

## ***Locking***

By default, when you drag points in a Geometry Expressions model, it adjusts the variable values to accommodate the drag as best it can. For example, in the model of a four-bar linkage below, dragging point B changes lengths  $a$  and  $b$  and angle  $\theta$ :



However, you may wish the drag to act as if lines AB, BC, and CD were rigid, and only angle  $\theta$  adjustable. To arrange this, lock all the variables except  $\theta$ .

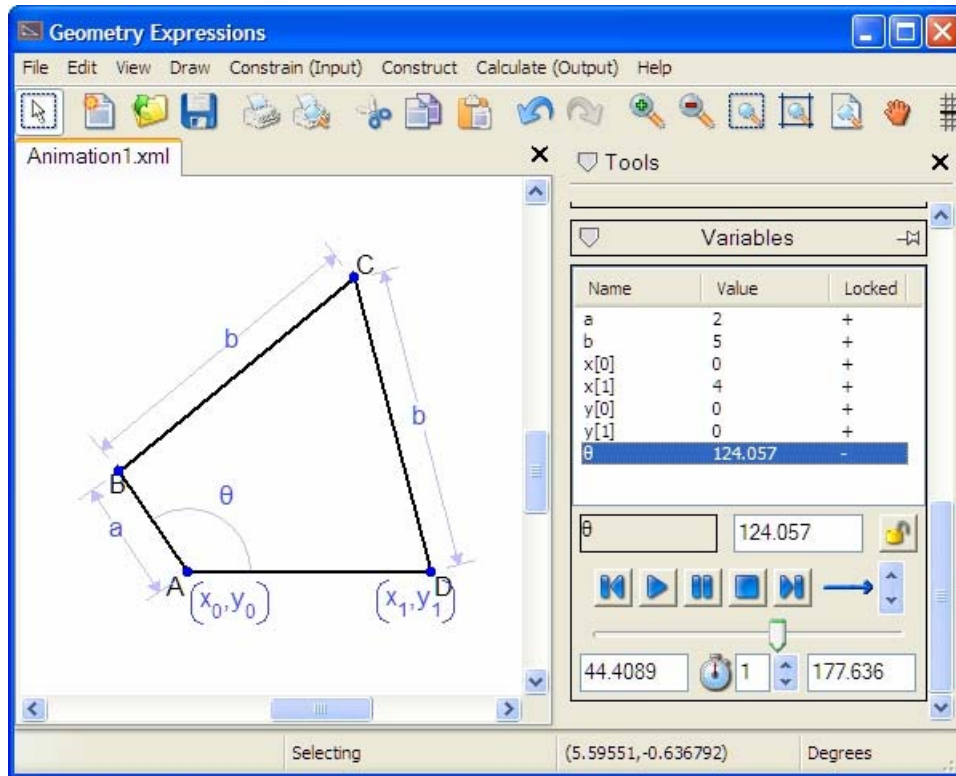
Locking a variable applies only to its behavior when you drag the drawing. You can still assign it a value by entering it in the input field.

### ***Animation***

Geometry Expressions allows you to set a range of values to a variable, then run through the range to animate your drawing, using the familiar video playback interface. To animate a drawing:

1. Select a variable to drive the animation.
2. Assign a range of values to that variable.
3. Press **Play**.

In the figure below we select  $\theta$  to crank the linkage:

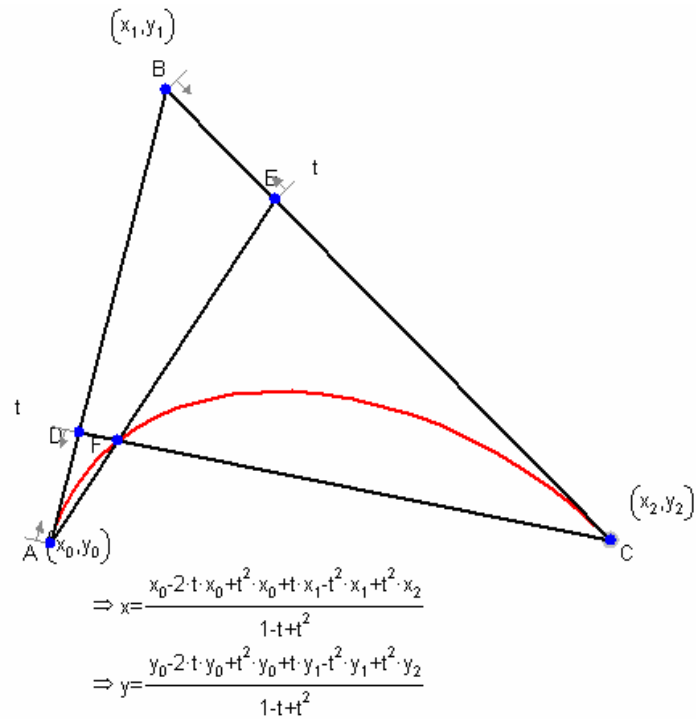


From other interactive geometry applications, you may be familiar with animating points along line segments or curves. Geometry Expressions provides this animation capability as well, using the constraint *point proportional along a curve*.

A point proportion  $t$  along a curve is defined thus:

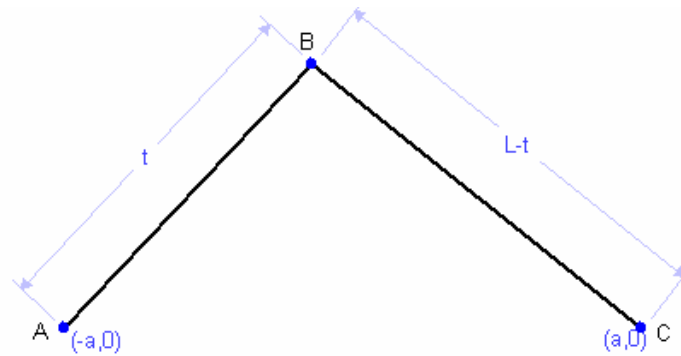
<b>for:</b>	<b>definition:</b>
line segment AB	point $(1-t)A + tB$
circle	point on the circle that subtends angle $t$ at the center
locus or envelope	point at variable value $t$

For example, in the following drawing, D is defined proportion  $t$  along AB, and E is defined proportion  $t$  along BC. The curve is the locus of F as  $t$  varies between 0 and 1:

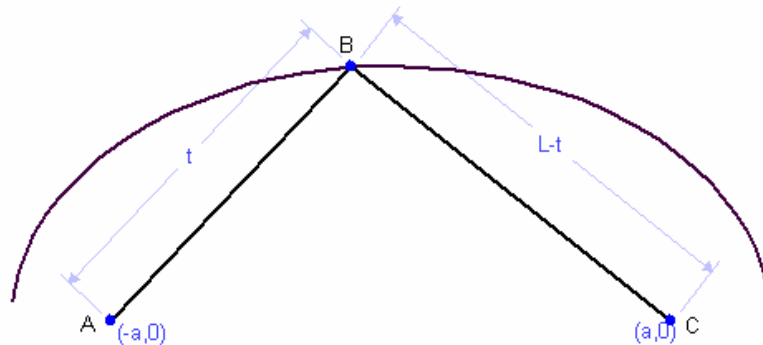


To animate this, we assign  $t$  the range 0 to 1, then push **Play**.

The construction of both locus and envelope curves can be defined in terms of any variable. For example, starting with the drawing below:



...we create a locus over values of  $t$ , keeping other variables constant:



## Importing and Exporting Expressions

You can copy expressions from Geometry Expressions and paste them into an algebra application such as Maple™ or Mathematica®, and you can also copy expressions from such applications and paste them into Geometry Expressions.

To enable this, Geometry Expressions uses the standard MathML™ interchange format. MathML — Mathematical Markup Language — is an extension of the widely adopted XML protocol, designed for mathematics applications.

The official MathML web page and specification is at:

[www.w3.org/Math/](http://www.w3.org/Math/)

MathML comes in two varieties:

- Presentation MathML
- Content MathML

When copying from a companion application into Geometry Expressions, use Content MathML, which is the variety that Geometry Expressions accepts.

As a convenience, Geometry Expressions places expressions you copy on the clipboard as both Presentation MathML and Content MathML. Many applications can accept this dual input and choose the variety they require.

Some applications, however, might not be able to accept dual input. If you are having difficulty with an expression copied from Geometry Expressions:

1. Consult the user's guide or help system for the target application to determine which variety of MathML it accepts.
2. Use **Edit > Copy As...** to invoke a dialog enabling you to choose between the two varieties. Only the one you choose is placed on the clipboard.

## Performance Tips

Various factors affect the simplicity and readability of the result and also, in some cases, the speed at which it is reached:

- How clearly have you defined your problem? The fewer unknowns you use, the faster the calculation and the more likely that the resulting expression will be simple and clear.
- How constrained is the problem? Fully constrained problems produce better results than underconstrained ones, because you control which unknowns are used.
- Using intermediate variables makes expressions less likely to simplify, but more likely to produce a result.
- For faster calculations, use lower settings of **Intermediate Value Complexity**.
- Use assumptions for simpler results, if your expression has absolute values in it. But check the assumption to make sure it's what you intend; if not, change the drawing to reflect your intention more accurately.





## II. *Geometry Expressions* Tutorials

These tutorials are provided to help you get started using *Geometry Expressions*. They refer to the ideas discussed in *Solving Geometry Problems Using Geometry Expressions*, which we recommend you read before proceeding with this tutorial.

### ***Documentation Conventions***

You see:      It means:

---

***Enter***      the Enter key

Control-Z      Press ***Control*** and ***Z*** at the same time.

Enter      Press ***Enter*** (This is usually done as the final step in completing some desired action.)

**File > New**      Execute the ***New*** command from the ***File*** menu.

# Tutorial 1: Define and solve a problem.

In this tutorial, you will:

- explore the tools,
- define a simple problem,
- calculate the solution,
- resolve a constraint conflict, and
- add and calculate an expression.

As you work, remember that there's no need to draw lengths and angles accurately. Just draw a sketch. Then, as you enter constraints and constructions, Geometry Expressions will update your sketch accordingly.

If you do happen to make a mistake, Geometry Expressions has multiple levels of **Edit > Undo** (Control-Z) and **Edit > Redo** (Control-Y) so that you can go back and forth through a sequence of steps.

## Explore the user interface.

1. Start Geometry Expressions
2. **File > New**

A new blank page opens with the tool palettes on the right:

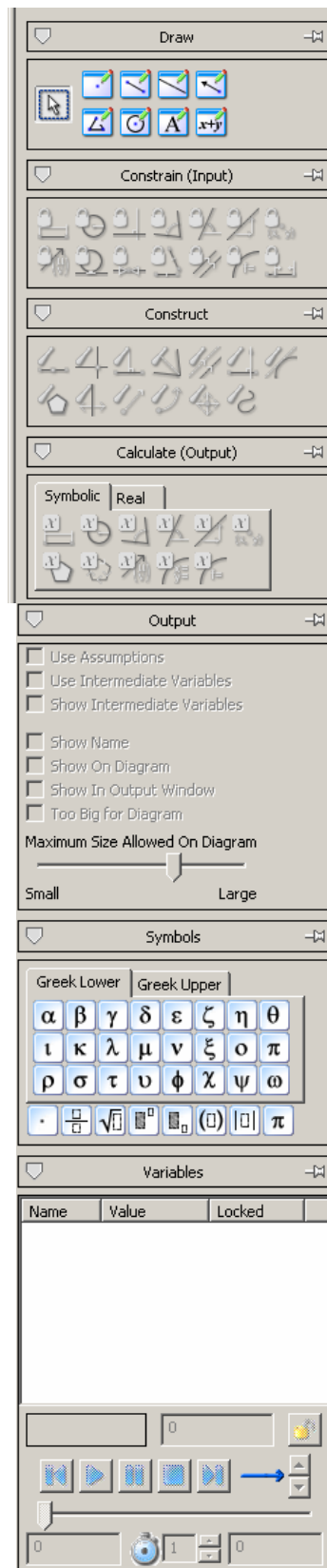
- **Draw** to create geometry objects,
- **Constrain (Input)** to input any constraints,
- **Construct** to create common, often used, geometry objects and
- **Calculate (Output)** to make calculations associated with selected geometry elements.

Below these are palettes that provide ways to:

- control the way expressions are calculated,
- insert symbols, and
- assign variable values.

Depending on your display size, you may need to scroll to see them all.

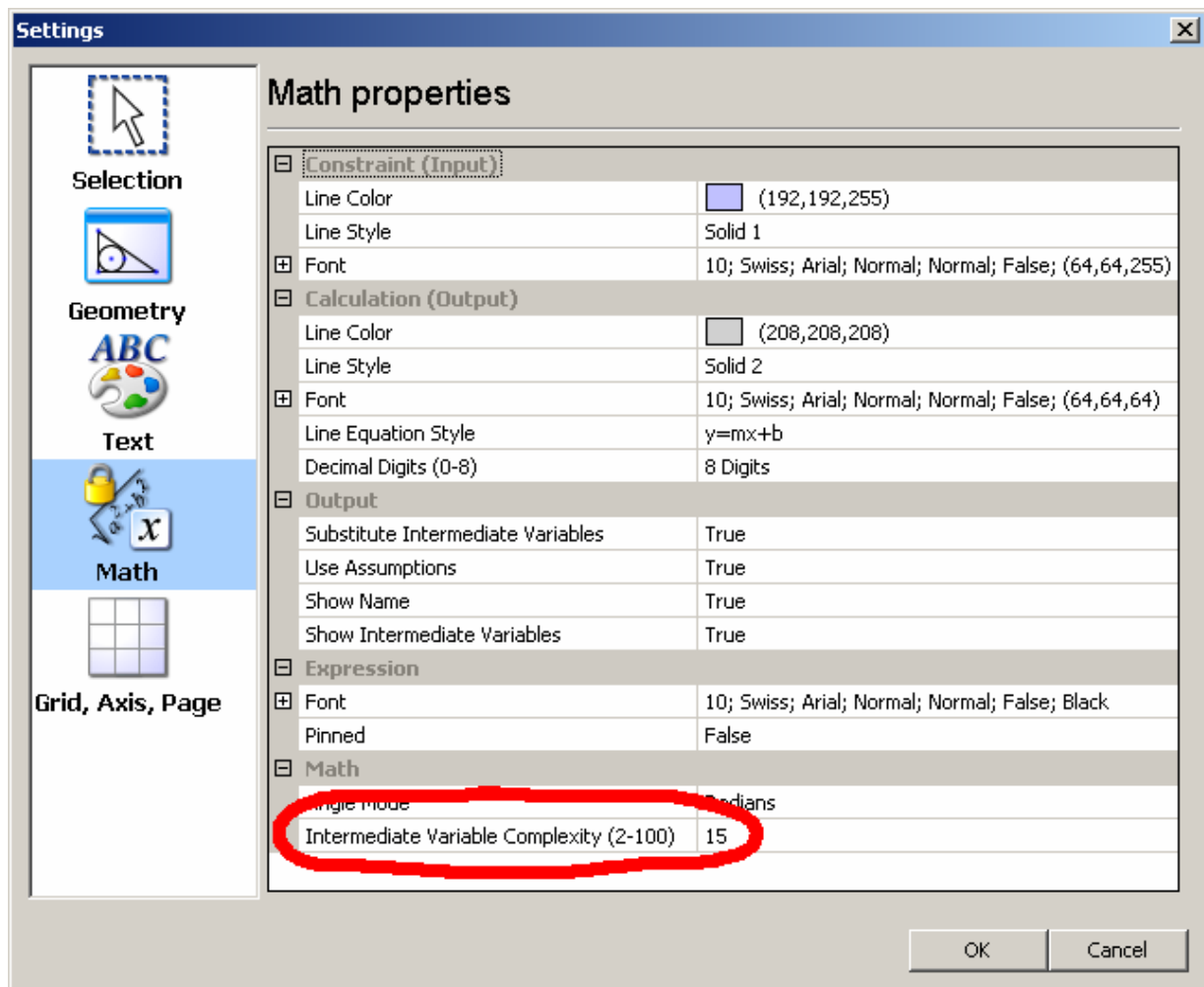
3. You can open or close each palette individually by clicking the arrow at the left of the palette title. In this way, you can close palettes that you won't be using. These tutorials won't be using



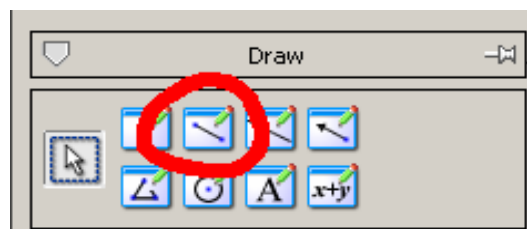
the Symbols palette, so you may close it now, if you wish.

You can customize the user interface in various ways. The options available in the **View** menu control zooming, scaling, axes etc. and the items in **Edit > Settings**, address other visual, text and mathematical properties.

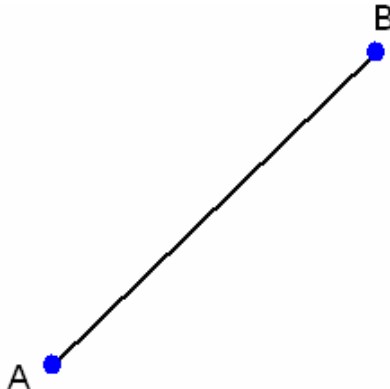
Under **Math**, for example, you can set the threshold for Intermediate Variable Complexity. A low threshold tells the application to substitute only simple terms with intermediate variables. The more complicated ones will not be substituted. A progressively higher threshold results in the automatic substitution of progressively larger and more complex terms with intermediate variables.



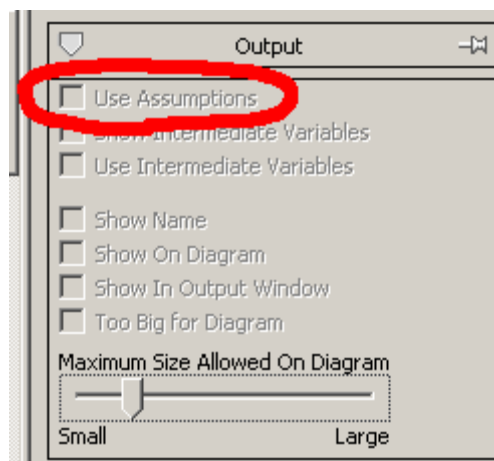
4. In the Draw palette, choose the line segment tool.



- Starting near the bottom left of the drawing area, click and hold the mouse button down as you draw a line segment going upward and to the right. Now release the mouse button. This will complete the drawing of a line segment. Geometry Expressions places the first point at the beginning of the line (where the cursor is when you first press the mouse button) and the second point at the end of the line segment (where you released the mouse button). Another way of drawing a line segment is to click over the start point and move the cursor to the position of the end point and click again. (I.e. it's not necessary to hold the mouse button down while positioning the cursor.) The end points are labeled A and B respectively.

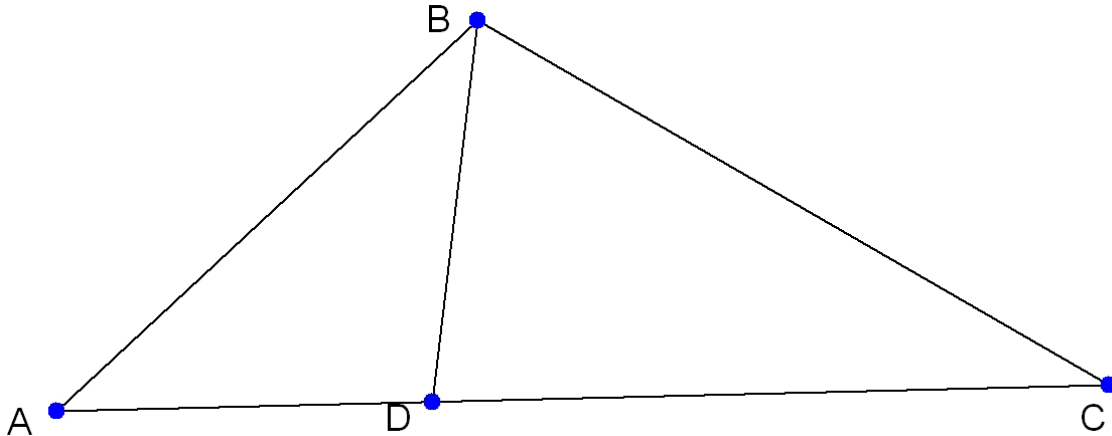


- With the line segment tool still selected, move the mouse around the drawing area, passing over one or both of these points. Geometry Expressions tracks the location of the drawing cursor, snapping it to an existing geometry object when the cursor gets close enough.
- Try creating other kinds of geometry objects using some of the other drawing tools. When you're ready to go on, undo them all except the line AB.
- Go to the Output palette. (Scroll down to find it, if it is not already visible.) This is where you will see a variety of options for controlling the output answers to your geometrical questions. An output expression will have to be highlighted before you will be able to select any of the options in this palette. For example, if the **Use Assumptions** checkbox is selected, then Geometry Expression will assume that symbolic lengths will be positive. This assumption often simplifies the output.



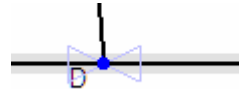
## Define a problem.

To start, we'll use the first line segment and add three more to create something like this:



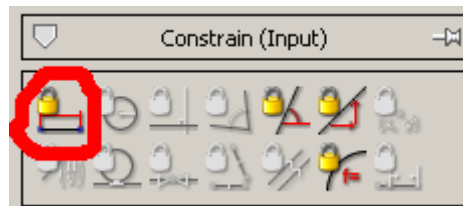
1. With the line segment tool selected, place the cursor near point B and draw three more line segments. (Remember, there's no need to draw precisely.)

To ensure that point D lies along the line AC, move the cursor near to AC until the



line becomes selected, D will then snap to this line.

2. Change to the selection tool and select line AB by clicking the mouse anywhere along its length.
3. With one object selected, some of the tools in the Constrain (Input) palette are enabled. Click the Distance/Length constraint. If a tool is not available its icon will be grayed.



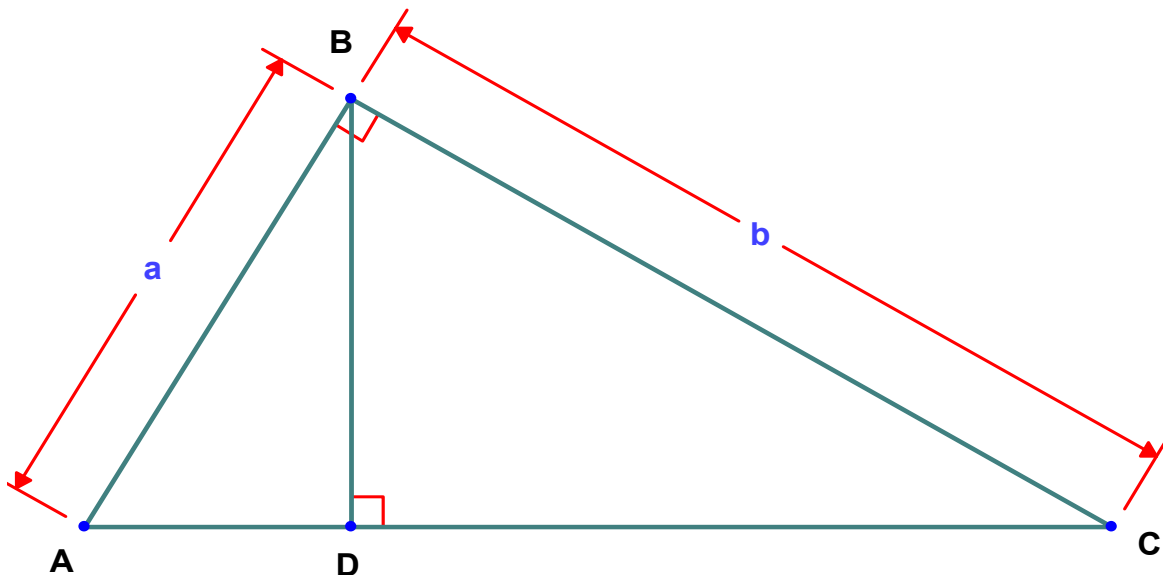
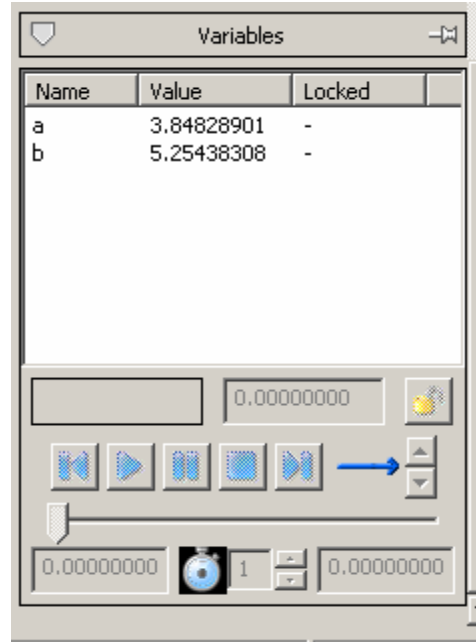
The constraint appears, along with the default variable name of  $a$ , selected and ready to change. Press **Enter** to accept it.

4. Select line BC and repeat the previous two steps to constrain its length to  $b$ .
5. You've now added two variables to your drawing. Go to the Variables palette (scroll down, if necessary) to see their names and values. The values displayed in the Variables palette are the current values, as taken from the default coordinate system. Notice that these values will change as you vary the lengths of the sides. Vice versa, if you select a variable and change its value in the Variables palette, the geometry will change accordingly.

- Select the variable  $a$  in the Variables list and, in the input field below, round the value to the nearest whole number.
- Select both lines  $AB$  and  $BC$ . (You can select more than one object at a time by making the first selection, then holding down **Shift** as you make subsequent selections.)
- With two objects selected, you'll see a different set of constraint tools enabled. Click the Perpendicular constraint to constrain the angle to  $90^\circ$ .  $ABC$  is now a right triangle.

Notice that point  $C$  moved, not point  $A$ , though either could have moved to satisfy the constraint. When it can, Geometry Expressions moves the most recently added object.

- There's one last constraint to add. Select both lines  $BD$  and  $AC$ , and constrain their angle to be perpendicular as well.

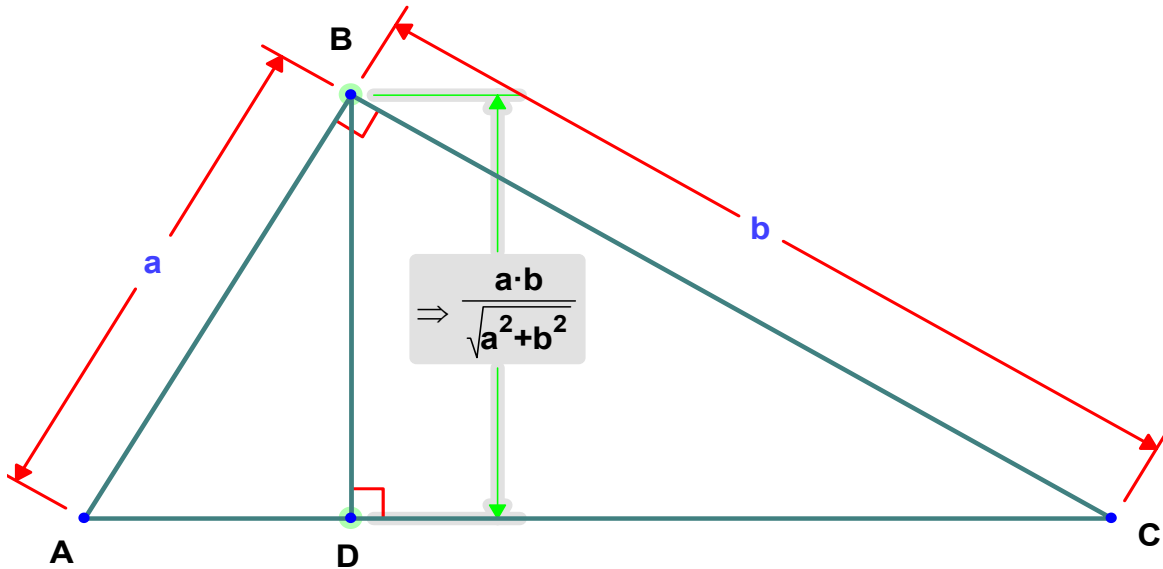


### **Calculate output.**

Suppose you want to know the length of the line  $BD$ :

- Select  $BD$ .
- The Calculate (Output) palette offers output as both numbers — **Real** — and expressions — **Symbolic**. Click on **Symbolic**, if it's not already in front, then click the length icon.

An expression appears, showing the requested length in terms of the two variables,  $a$  and  $b$ . If you select this expression, you will be able to move it wherever you wish.



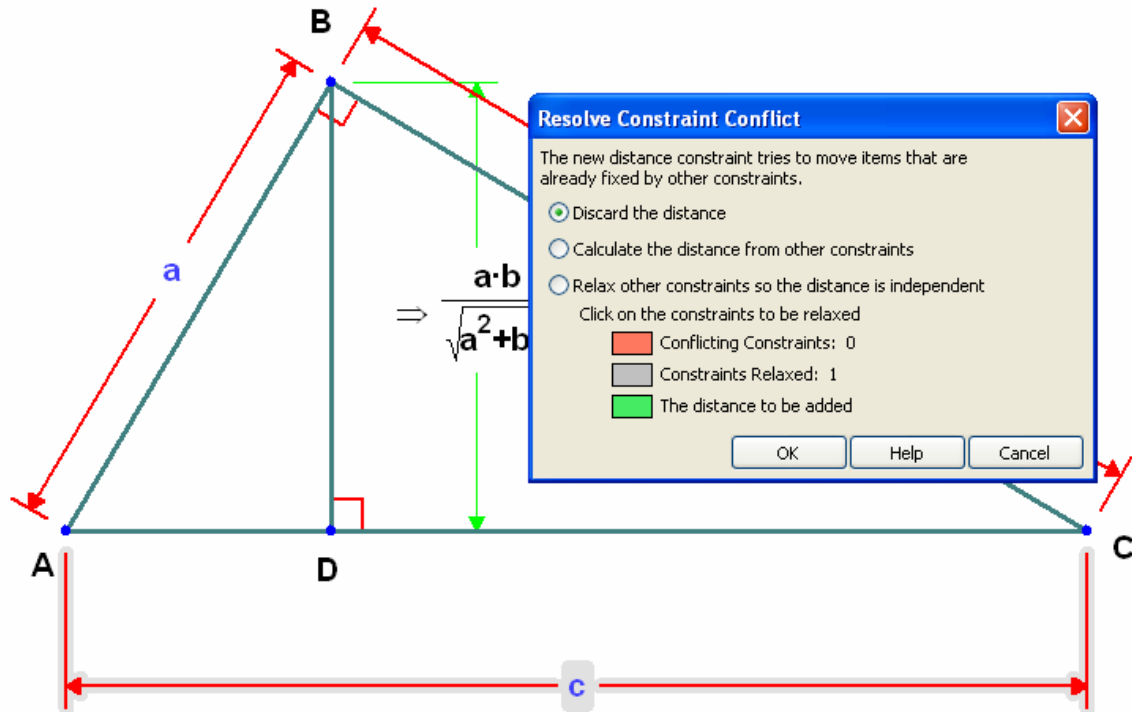
The double arrow => that appears to the left of the expression indicates that the expression is an output that the application has calculated.

***Resolve a constraint conflict.***

With the lengths of two sides and the included angle already constrained, the triangle is fully defined; any new constraint you add will conflict with the existing ones. While this may be obvious, you may later encounter conflicts that are less obvious. This task takes you through the steps necessary to resolve such conflicts.

1. Select the line AC and add a length constraint. You'll see this dialog:





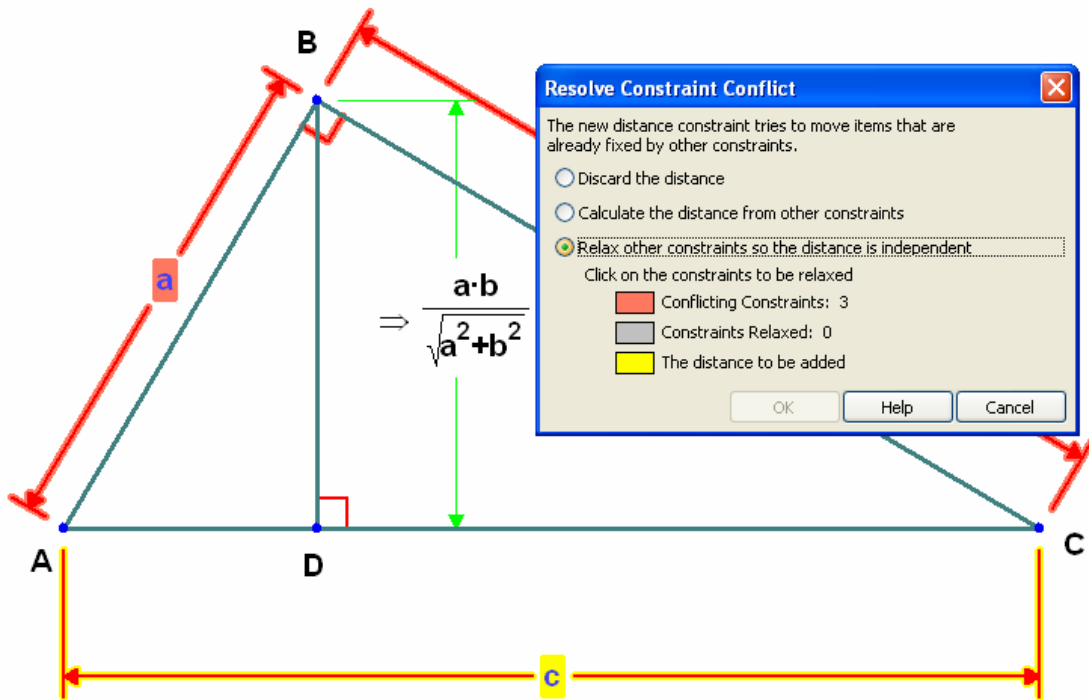
2. The dialog offers you three choices:

- *Discard the distance.* This will close the dialog and put you back in the position you were before you attempted to constrain the length of AC..
- *Calculate the distance from other constraints.* This will change your attempt to constrain the length to a request for the length to be calculated for you.
- *Relax other constraints so the distance is independent.* This will give you an opportunity to remove one of the conflicting constraints so that the one you were trying to impose will become an independent constraint and no longer depend on existing constraint. You can then choose which constraint to relax.

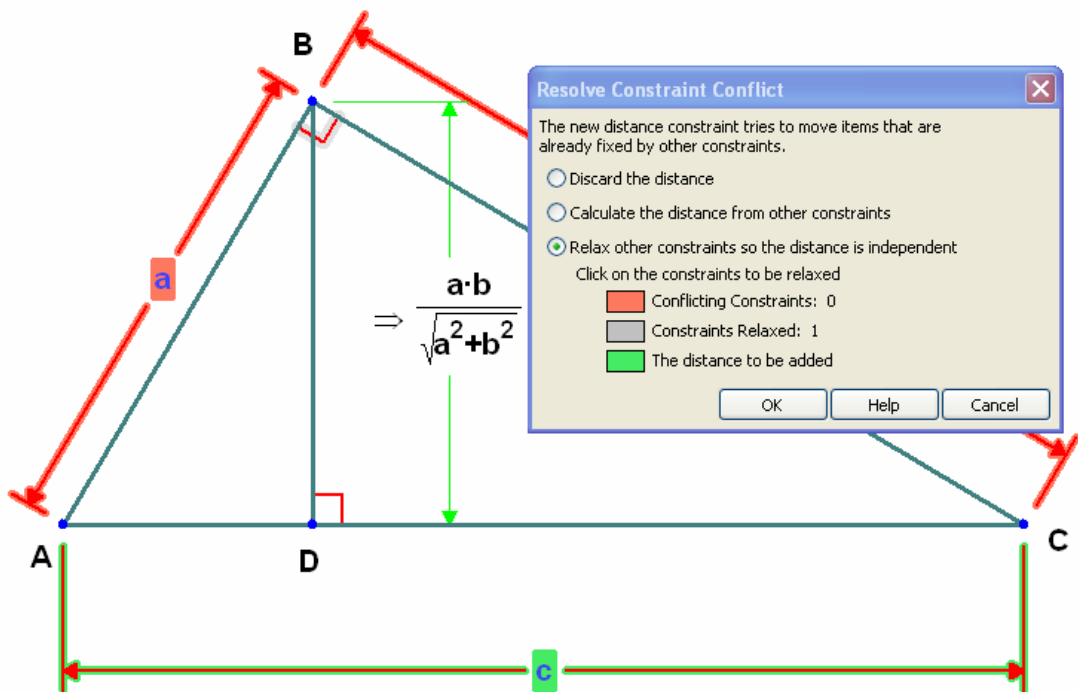
3. Click the second radio button, *Calculate the distance from the other constraints*, and then click the **OK** button. You'll see the expression appear with the arrow ( $\Rightarrow$ ) that indicates it's an output. (It may also be given a name. This will be the case if the **Show Name** box is checked in the **Output** palette.

4. **Edit > Undo**

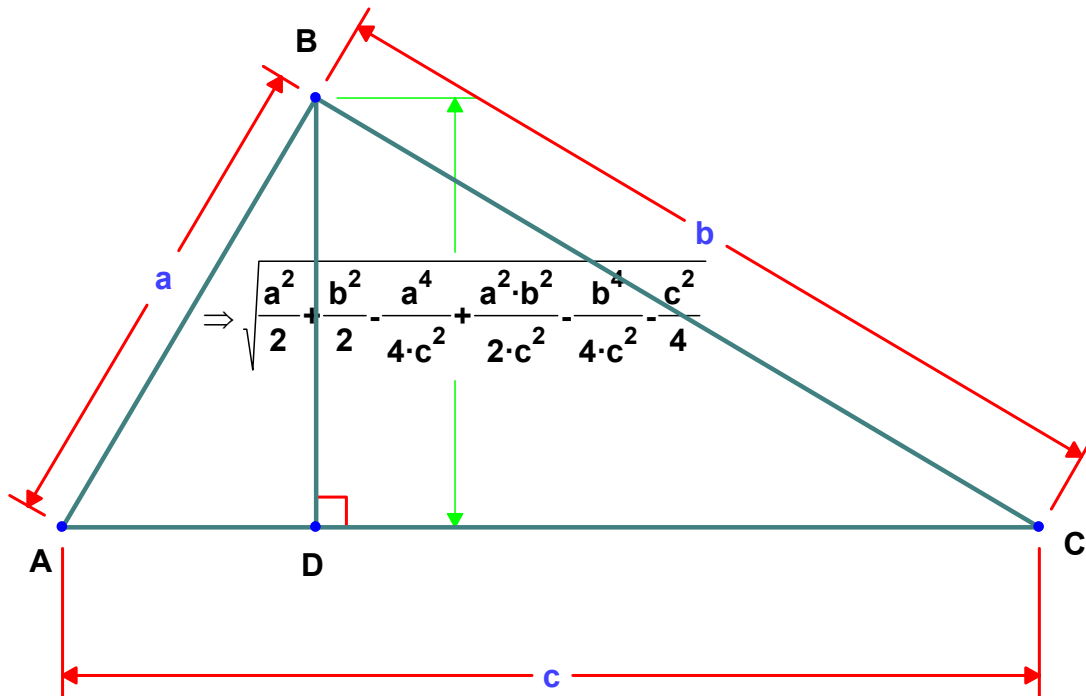
5. Try to add the new constraint again. This time, click the third radio button, *Relax other constraints so the distance is independent*. As soon as you click it, the conflicting constraints highlight in red and the new one in yellow:



- Click on the angle at B. Its highlight changes to indicate it's selected; the conflicting constraint highlights vanish, and the new one now highlights in green.



7. Click **OK**. Now the figure includes the length constraint  $c$ , while the perpendicular angle constraint at B is gone.



8. **Edit > Undo** again. We're going to need that perpendicular angle later. Leave the drawing as it was, without length constraint  $c$ .

### ***Make your own expression.***

Geometry Expressions can calculate output as expressions, but you can also define and add your own expressions to a drawing.

1. Select the two points A and D.
2. Calculate the distance between them as a symbolic output.
3. Select the expression you've just output.
4. In the Output panel, check **Show Name**.

To the left of the arrow, the name  $z_n$  appears. (Because you may have created one or more of these already while exploring, we can't predict the exact number in the subscript.) This is the symbol the application assigns by default as the name of the expression.

5. Select the two points D and C, and repeat the previous three steps to get a second expression, this one named  $z_{n+1}$ . (Subscripts increase by one each time the symbol they apply to is reused.)

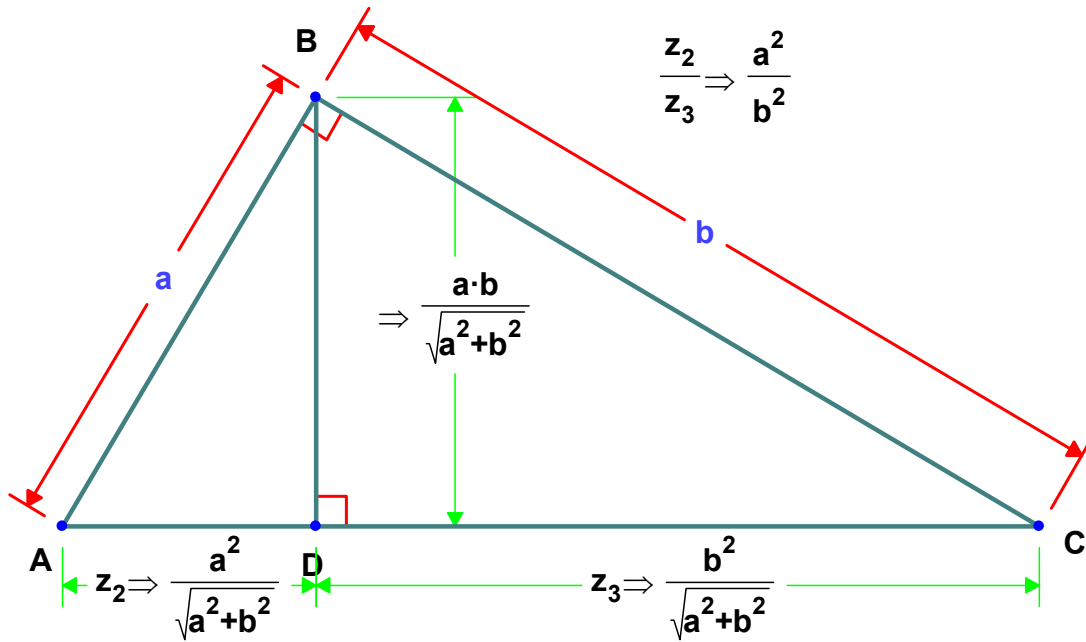
Using these names, you can make an expression of your own to refer to these distances. For example, you can calculate the ratio of the two lengths.

- In the Draw palette, choose the Expression tool.
- Click in the drawing where you want to put the expression. The expression appears as a 0, selected for overwriting. To replace the zero, enter:

$$z[n] / z[n+1]$$

(Square brackets indicate subscripts. Replace the  $n$  and  $n+1$  with the subscripts for your own named expressions.)

- The expression is calculated and displayed.



## Tutorial 2: Create and manipulate a curve.

This tutorial lets you try some of the more advanced features of Geometry Expressions. In this tutorial, you will:

- create a locus,
- lock a variable's value to see the effect it has on the drawing,
- animate the drawing by setting start and stop values for a variable,
- calculate two kinds of equations for the locus,
- reflect the locus in a line, and
- compare the equations for the locus and its reflection.

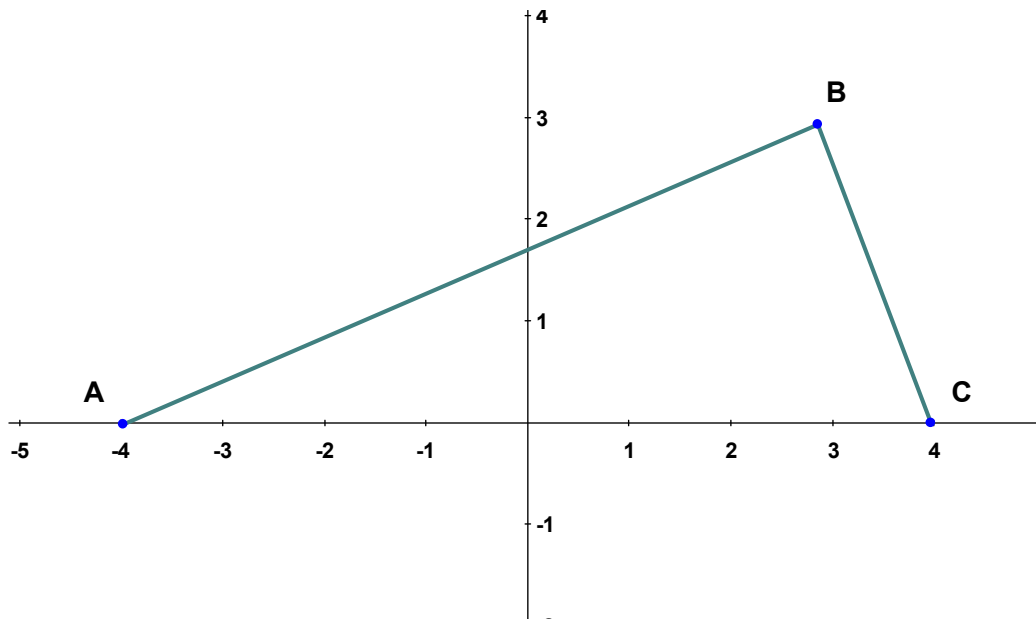
### ***Draw a locus.***

We're now going to re-create the old exercise of drawing an ellipse using a pencil, two pins, and a piece of string.

1. Turn on the axes by clicking on the grid tool in the toolbar.

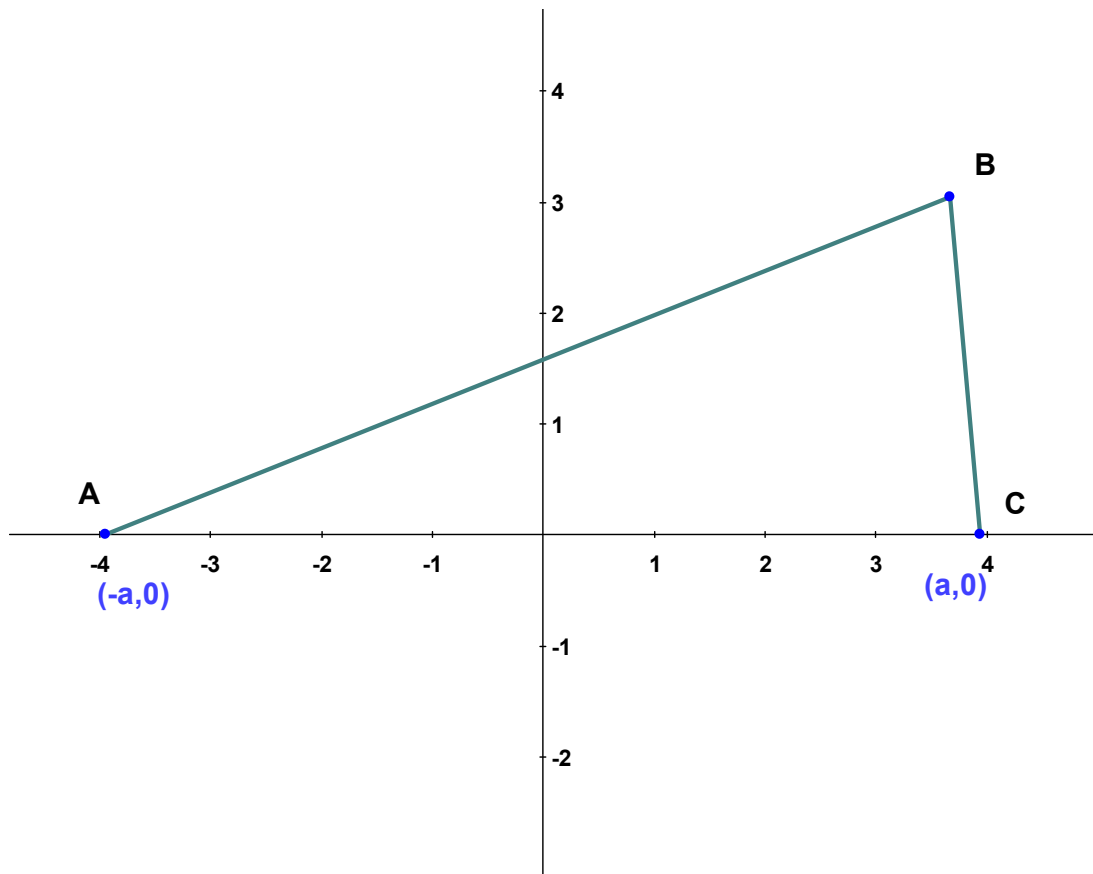


2. Make two line segments, AB and BC. Points A and C are the pins, while B represents the pencil.

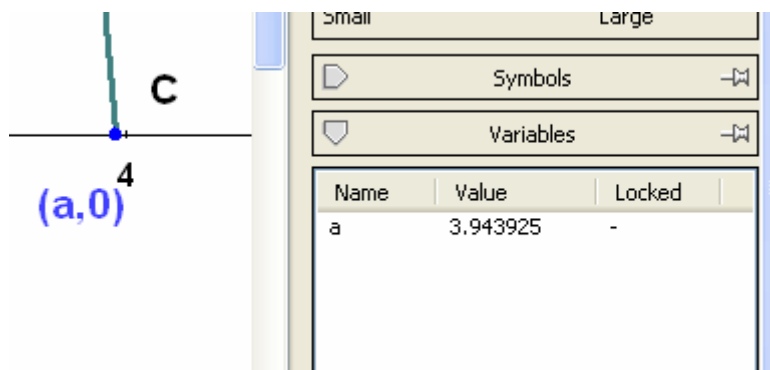


3. Select point A.

4. Add an input constraint of type *Coordinate* to set the coordinates of the point.
5. When the constraint highlights, enter (not forgetting the parentheses):  
 $(-a, 0)$
6. Select point C and constrain its coordinates to be at  $(a, 0)$ .



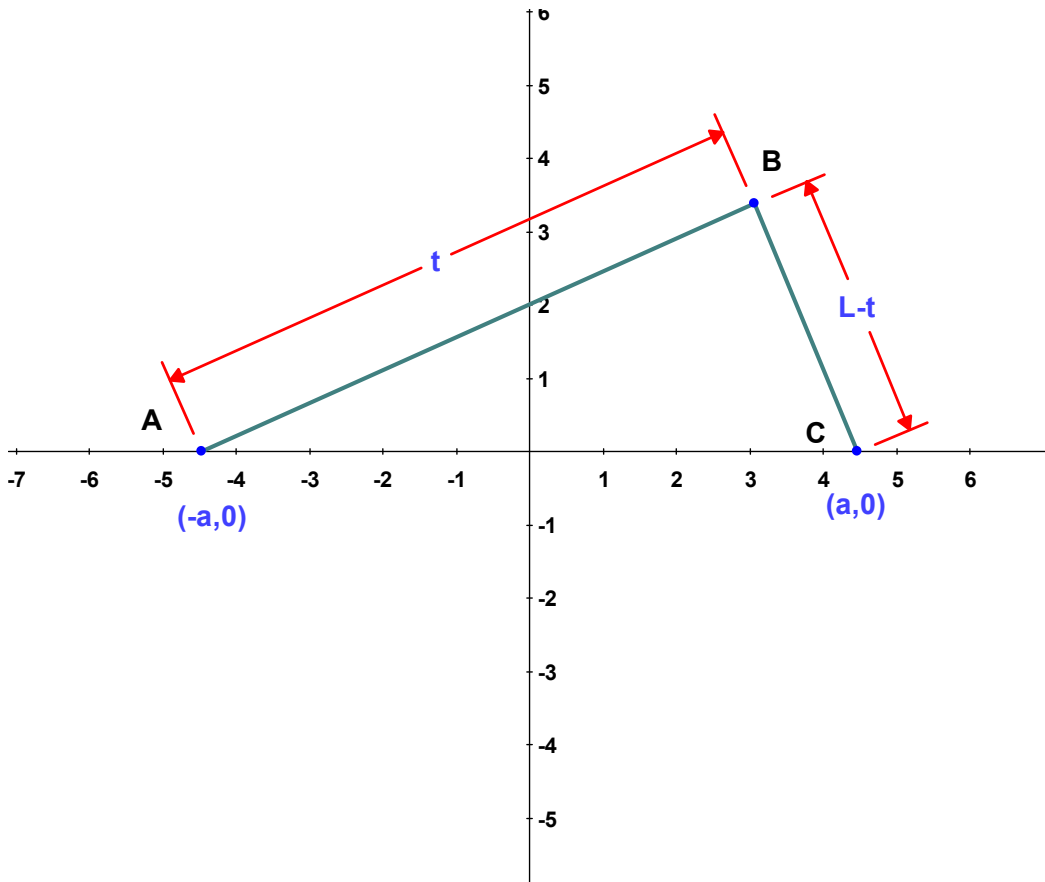
7. Scroll down, if necessary, to see the value of  $a$  in the Variables palette.



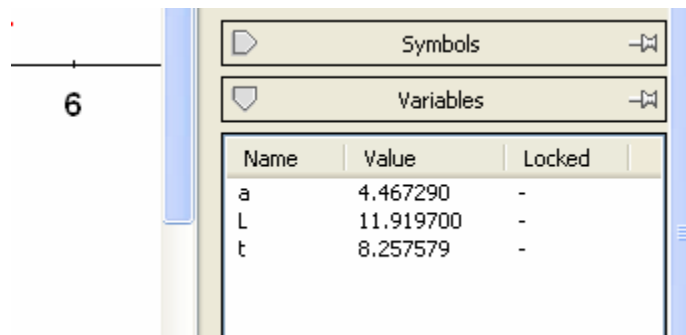
8. Reselect A and drag it to the right a short distance. As you drag, notice:
  - A is now constrained to lie along the X axis; moving it up or down has no effect.
  - When you move A, C also moves.

— When you move A, the value of the associated variable,  $a$ , changes in the Variables list.

9. Select the line AB and constrain its length to be distance  $t$ .
10. Select the line BC and constrain its length to be  $L-t$ .  $L$  now represents the length of the string.

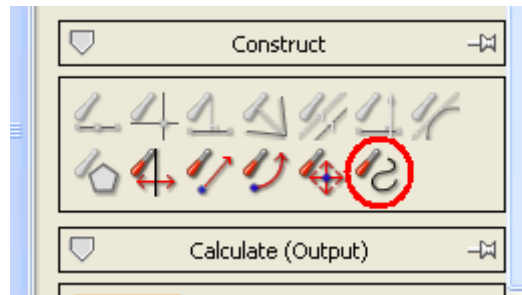


11. Notice that the two new variables have appeared in the Variables list.



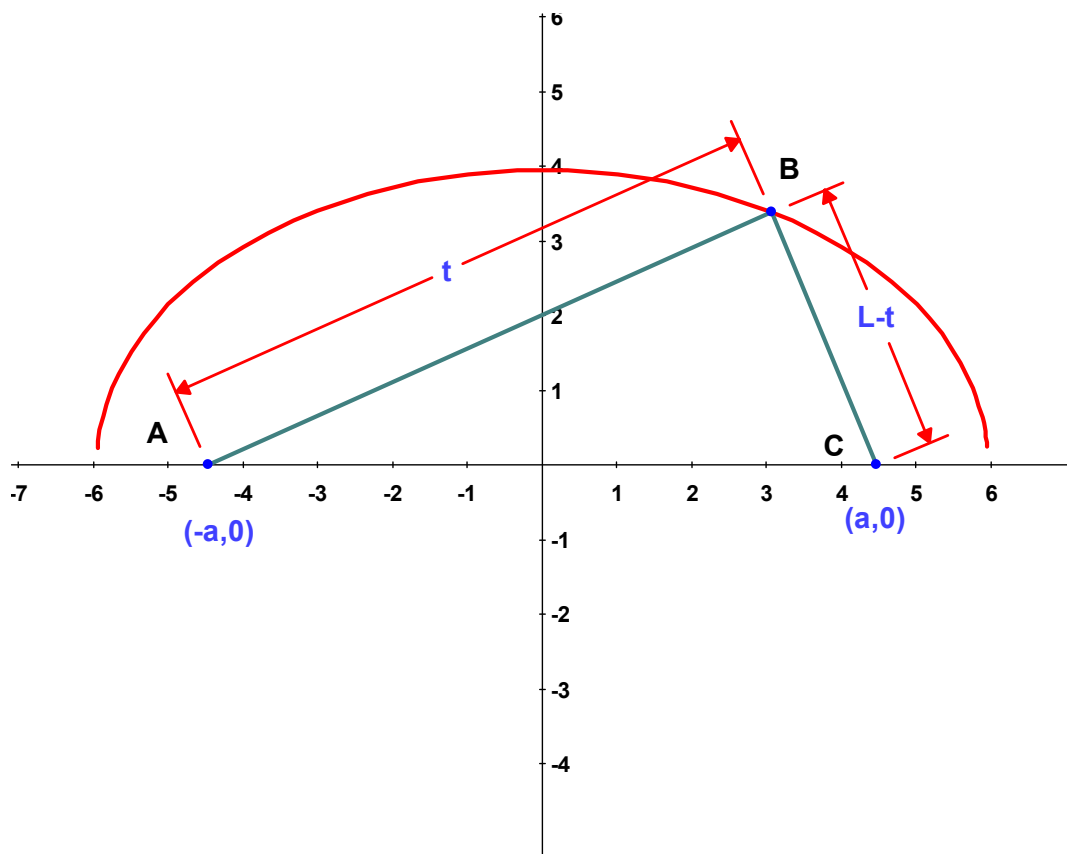
12. Select B.

13. In the Construct palette, select Locus to construct a locus through B.



14. In the resulting dialog, choose  $t$  as the parameter to vary, and enter start and end values of 0–25, guesses that will probably produce a complete curve.

The locus appears — half of an ellipse, above the X-axis.



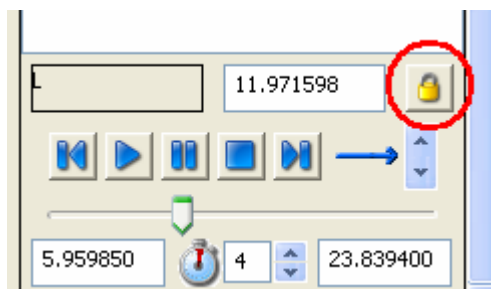
15. Now drag B up and down, the equivalent of changing the length of the string, and notice how the ellipse changes. You can also drag A or C to change the position of the pins.

### **Lock a variable.**

When you draw an ellipse this way using real pins, pencil, and string, the length of the string can't change. To emulate this real-world behavior, we can lock the value of the variable  $L$ .



1. In the Variables list, select  $L$  and click the lock icon below the list. A plus sign (+) appears next to the locked variable in the Locked column.

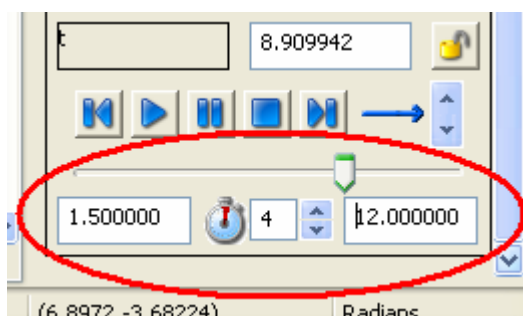



2. Select points A or C and drag it. With the length of the string constant, the locus will change accordingly.

### ***Animate a variable.***

You can also animate the drawing. To do so, we can set start and stop values for the variable  $t$ .

1. In the Variables list, select  $t$ .
2. The first input field below the video playback interface specifies the start value. Enter 1.5.
3. The middle field specifies the number of seconds for one cycle of the animation. Accept the default value of 4. The input field to its right specifies the stop value. Enter 12.



4. To animate the drawing, click the **Play** button ()

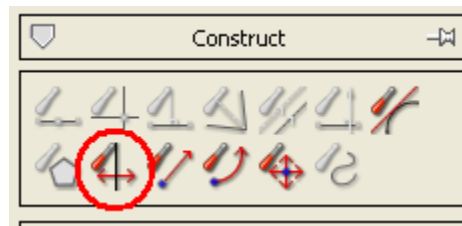
**NOTE:** The start and stop values that yield an interesting animation depend to some extent on where you've located objects in your drawing. Experiment to find values that make an interesting animation. Values that do not allow GE to create a construction sequence will cause objects to disappear briefly. They'll reappear when values make sense again, or when the animation stops.

### Calculate parametric and implicit equations.

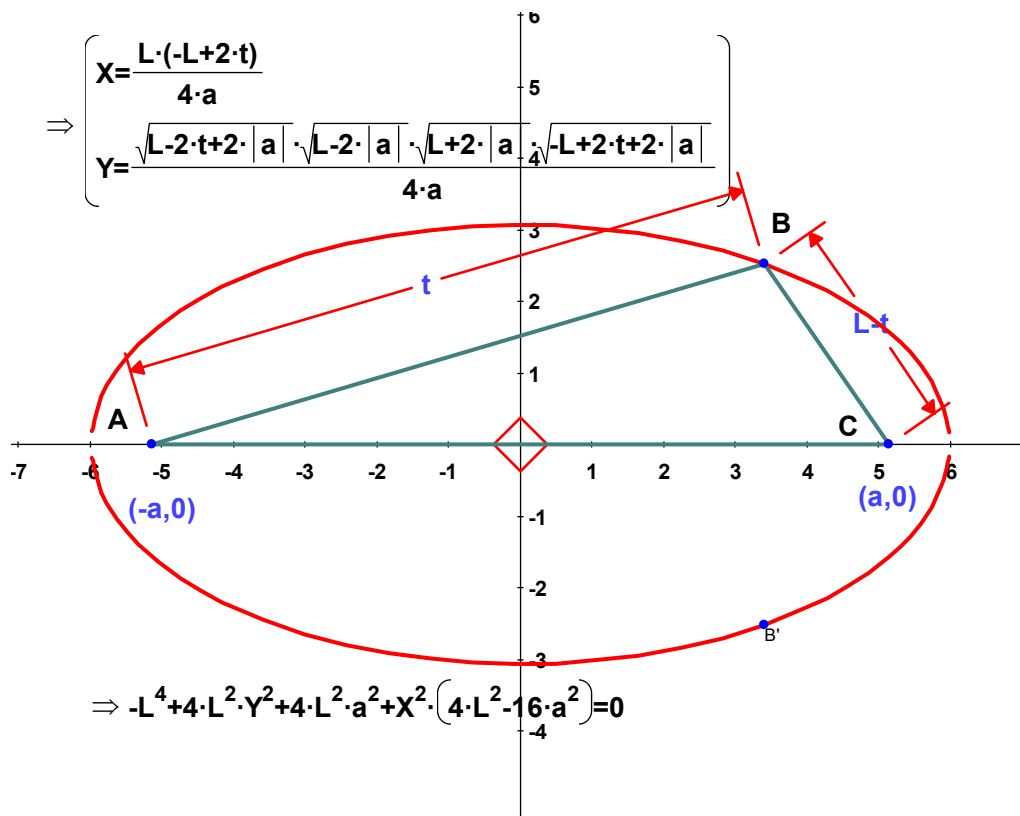
1. Select the locus. In the Calculate (Output) palette, chose Parametric equation. The resulting expressions are the formulas for  $x$  and  $y$  as a function of  $t$ . Move the expression wherever you wish on the drawing.
2. Select the locus again and, this time, request the Implicit equation. The resulting expression gives the equation of the curve in terms of  $x$  and  $y$ .

### Make the locus reflection.

1. Select the line segment tool again and make a line from A to C. This line lies on the X-axis.
2. Select the curve.
3. In the Construct palette, choose Reflection. The status bar now shows a message prompting you to choose the axis about which to reflect the curve.



4. Select the line segment AC. This will reflect the curve about the X-axis.



B is also reflected, its reflection appearing on the lower curve as B'.

5. Select B and drag it; B' follows. But, though you can select B', you can't drag it; it's locked as the reflection of B.

### ***Compare locus and reflection equations.***

1. Select the reflected curve and request its parametric equation (from the Calculate (Output) palette).
2. Compare them side-by-side and you'll see that the x-parameterizations are the same and the y-parameterizations differ only in sign (which is, of course, as it should be).
3. If you repeat the experiment for the implicit equation, you'll see they are identical.

### ***Explore on your own.***

Having completed the introductory tutorials, we hope you'll be comfortable exploring on your own. If you wish, further interesting examples can be found on our website, [www.geometryexpressions.com/explorations/](http://www.geometryexpressions.com/explorations/)

# Index

Absolute Values .....	21	Construction-based Drawing .....	6
ambiguities .....	9	constructions.....	4
Ambiguity		ellipse .....	44
Resolving .....	12	Intermediate Variables .....	23
animation.....	27	Locking Variables.....	26
Animation .....	48	Using the lock .....	26
Application-added Variables .....	10	Locus .....	44
Assumptions.....	21	output	
Constraint-based Drawing.....	5	calculations .....	21
constraints .....	4, 10, 14	Reflection .....	49
conflicts .....	13, 39	Variables	
system added.....	10	Assignment .....	26
Construction Sequence.....	9	Controlling.....	26