Using Argument Diagramming Software in the Classroom

MARALEE HARRELL
Carnegie Mellon University

Introduction

After teaching a wide variety of philosophy courses at the undergraduate level, I came to believe that the fundamental barrier students faced in my classes was an inability to read philosophy, especially primary sources. Most students, it seemed, could get the general idea of a text, but could not follow the argument. Indeed, most students did not recognize that the text presented an argument; rather, they would read the text as a story. This type of reading was plain when, for example, students were asked to give presentations; generally the student merely highlighted the points the author made, in the sequence the author made them.

This type of reading of philosophy texts should not be unexpected, however. Before college, students have years of practice reading and analyzing stories, but they receive little instruction on recognizing, analyzing or creating arguments. In fact, even in college, unless a student takes a critical reasoning course in which argument analysis is the focus, most students learn how to understand, analyze and create arguments only incidentally. In our department, for example, students enrolled in upper division philosophy courses who have taken our introductory philosophy course are generally no better at these skills than those who did not.

In discussing this situation with a colleague, it occurred to us that one of the things students lack is a template for reading philosophy. Most students have a sort of template for reading stories, developed from years of English courses. The template tells them what to look for (main characters, conflict, climax, conflict resolution, dénouement, etc.). What we need to do in our introductory philosophy course, we decided, is give the students a template—a sort of map they can fill in, as they find the relevant parts of the text.

An argument diagram is just this sort of map. In such a diagram, text boxes are used to represent claims, and arrows and lines are used to represent connections. Thus, if we consider an argument to be a series of statements in which one is the conclusion, and the others are premises supporting this conclusion, then an argument diagram is a visual representation of these statements and the inferential connections between them.

For example, at the end of *Meno*, Plato (1976) argues through the character of Socrates that virtue is a gift from the gods (89d-100b). While the English translations of Plato's works are among the more readable philosophical texts, it is still the case not only that the text contains many more sentences than just the propositions that are part of the argument, but also that, proceeding necessarily linearly, the prose obscures the inferential structure of the argument. Thus anyone who wishes to understand and evaluate the argument may reasonably be confused. If, on the other hand, we are able to extract just the statements Plato uses to support his conclusion, and visually represent the connections between these statements (as shown in Figure 1), the structure of the argument is immediately clear, as are the places where we may critique or applaud it.

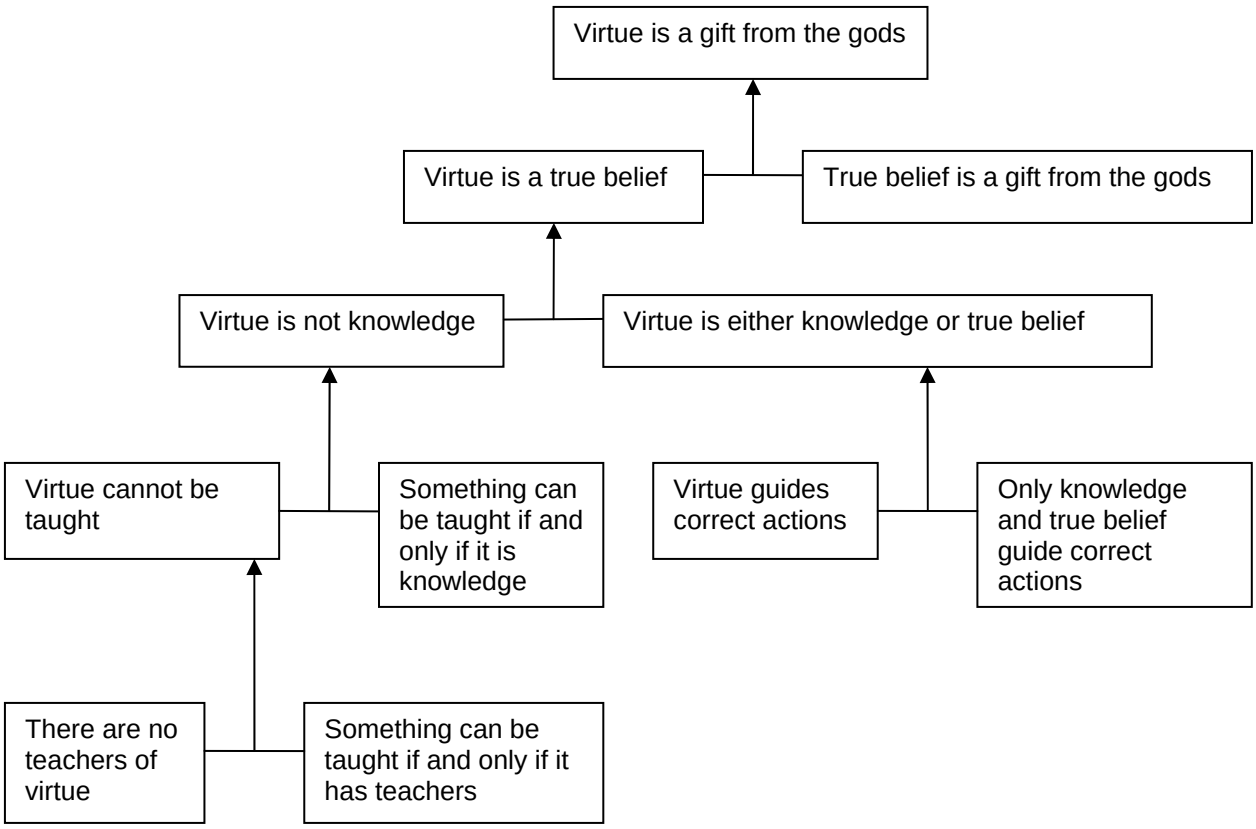

Figure 1 An argument diagram representing one of the arguments in Plato's *Meno*.

Having to create an argument diagram prompts students to look for claims an author makes, as well as the connections between these claims. Indicator words (e.g. "and," "because," "hence," "therefore," "since," etc.) can be used to determine these connections.

Argument Diagramming in the Classroom

In the introductory philosophy class I teach every semester, I use argument diagramming mainly for assistance in the analysis and critique of arguments presented in primary source texts (as with the example from Plato's *Meno* above). At the start of the semester, I teach the students the basics of constructing argument diagrams to accurately reflect the arguments presented in the text. Then, the students are given short passages to diagram as homework throughout the semester.

When I explain the answers to these homework assignments, I like to take the students through the process I used to create the diagrams. We go step by step, singling out the statements, and using indicator words and context to determine the inferential connections between these statements.

The first time I used argument diagramming in the classroom this way, I did not use any argument diagramming software; in the first half of the semester I used the chalkboard, and, since this proved to be messy and time consuming, in the second half I presented fully formed diagrams on overhead slides. For their part, most of the students submitted diagrams drawn by

hand, while some used the drawing tools in Word to produce their diagrams. Even this static presentation and the cumbersome drawing of the diagrams, though, helped the students tremendously. Nearly all of my students learned how to construct a diagram that accurately represents the argument given in text, and these students improved their critical thinking skills significantly more than the students who did not learn this skill, both in my section and in the other sections of the introductory course (Harrell 2004, 2005).

Even with these great results, I still want the students to experience more tangibly the way that I construct diagrams: how I move boxes around, create and erase arrows, set aside smaller sub-arguments to be connected to the overall argument later. Up until now I had done all this with a pencil and paper, and showing the exact steps of *that* process to the students would, I believe, be at best unhelpful.

My Ideal Argument Diagramming Software

Figure 1 was created in Microsoft Word, using the drawing tools. As I said above, this is very cumbersome and time consuming, not to mention static. With this sort of tool, I must work out the argument diagram ahead of time, construct it in Word, and then present it to my students fully formed. The advantage of software systems is the potential to construct the diagram "on the fly" both for the user, and for presentation to the students. It is especially important to be able to present a diagram dynamically to my students so they can see, as well as participate in, its construction. This kind of practice allows the students to construct argument diagrams themselves that represent both arguments in texts and their own arguments, and it has been shown that the ability to construct argument diagrams improves a student's critical thinking skills (see, e.g. van Gelder, 2001, 2003; Twardy 2004; Harrell 2004, 2005).

There are several additional important features of a software program used to dynamically present the construction of an argument diagram:

- Having different representations for (a) premises that need to be combined in order to support a (sub-) conclusion (a linked argument) and (b) premises that provide separate support for a (sub-) conclusion (a convergent argument).
- Having different representations for supporting statements, objections, and replies to objections.
- Moving the text boxes around (e.g., to make room for more of the argument, or to highlight different parts of the argument) and having the connections between boxes remain intact while moving.
- Determining the layout of the diagram on the screen (e.g., to indicate the order that the premises appear in the text by the left to right order of the text boxes on the screen).
- Entering an arbitrarily long segment of text into a text box (e.g., to be able to use the author's exact, or near to exact, words).
- Changing the size of the font in the textboxes (for personal use 12 pt may be fine, but for classroom presentation it is helpful to have 18pt or larger).

Argument Diagramming Software on the Market

With the rise in awareness of both how visualization can aid learning and how computers can aid visualization, there are a number of software packages now available that facilitate argument diagramming. Some of the more prominent are Araucaria,[1] Argutect,[2] Athena Standard,[3]

Inspiration,[4] and Reason!Able.[5] I teach the use of argument diagrams in all of my classes for the purposes of both understanding and evaluating texts and for creating novel arguments. It is my intent in this section to give an overview of the advantages and drawbacks of each of these packages from this perspective.

Araucaria

Araucaria can only be used for analyzing pre-existing arguments saved as plain text. You must upload the text first; when this is done, the text appears in the blue box next to the workspace. To create the "nodes" of the argument, you highlight a proposition in the text area, and then click on the workspace; a yellow circle with a letter inside appears in the bottom left corner of the screen. Once there are at least two nodes in the workspace, the process of building an argument diagram can begin. Clicking, for example, on the "B" node, and then dragging the arrow from it to the "A" node, produces an arrow from B to A indicating that B is a premise that supports A. Iterating this process of creating nodes and drawing arrows results in a full argument diagram, as shown in Figure 2.

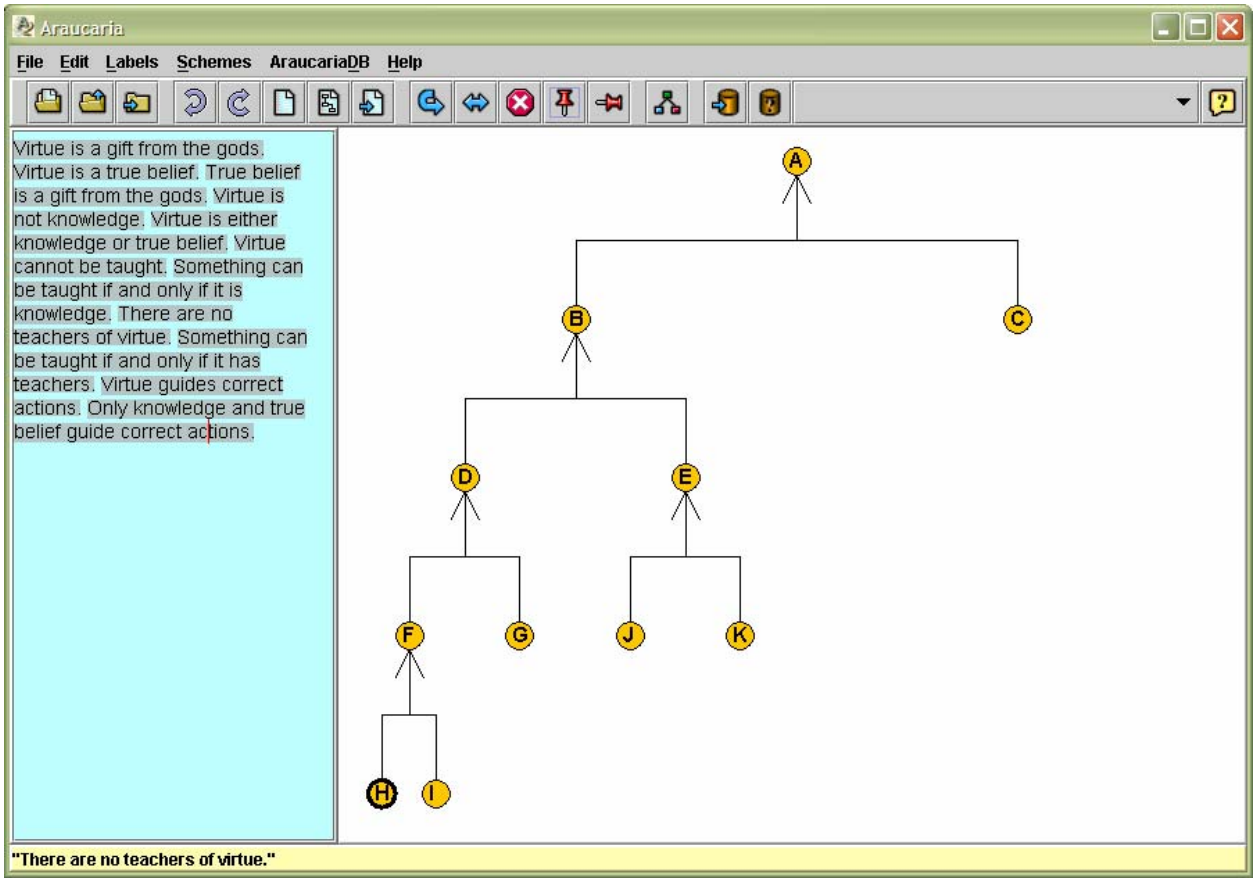

Figure 2 A screen shot of an argument diagram produced by Araucaria, representing the same argument from Plato's *Meno* that is represented in Figure 1.

If you wish to view the argument diagram with text boxes instead of lettered nodes, you can click on File → Full Text to create a .jpeg image as shown in Figure 3.

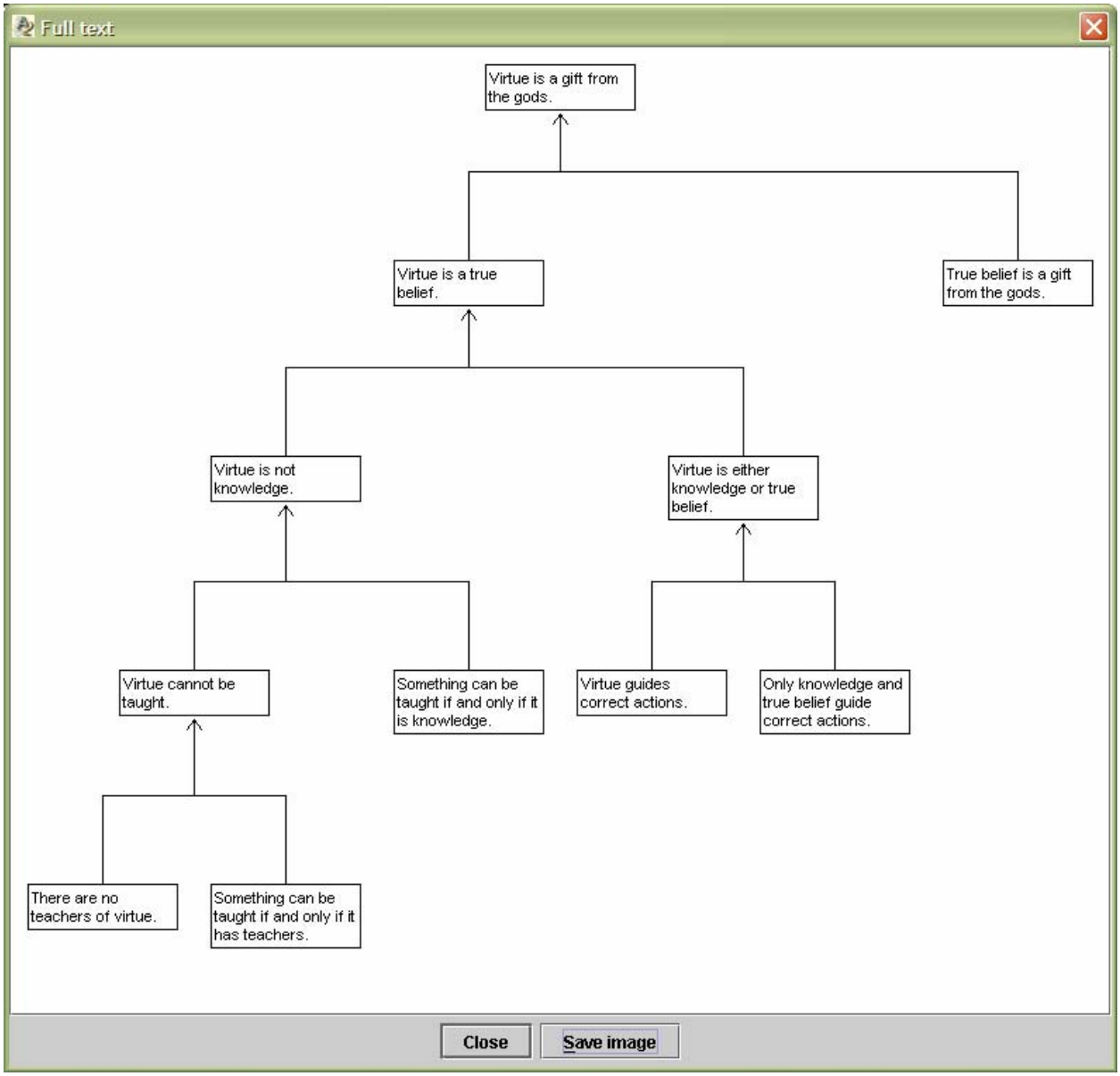

Figure 3 The .jpeg image of the full text version of an argument diagram produced by Araucaria, representing the same argument from Plato's *Meno* that is represented in Figure 1.

Despite its highly stylized appearance, Araucaria has some very good features: (1) you can easily represent both linked and convergent arguments, (2) given an uploaded file of text, you can easily create a diagram on the fly for a demonstration, (3) there is a straightforward way to represent objections to propositions in the argument and reasons for these objections, and (4) there are tools available to represent an evaluation of the argument as well as the argument itself (these tools include indications of the strength of premises as well as strength of inferential connections).

Unfortunately, Araucaria has several drawbacks. (1) Although you can represent objections to propositions, there is no way to represent a reply to that objection other than as an objection to an objection. (2) The requirement that text must be uploaded is a severe limitation. This type of text is not always readily available, especially for students (and professors!) who are trying to analyze an argument from a printed text. Copying the text to be analyzed into a text file becomes an overwhelming obstacle when the text is Descartes' *Meditations*. (3) It is difficult to rearrange the nodes; you can't drag the nodes around the workspace. The nodes supporting a conclusion node are ordered across the screen from left to right in the order that they were attached to the conclusion; thus, to change the left to right ordering, one must redo the attachment of the nodes. (4) For a complicated argument, it is difficult to keep track of which nodes represent which propositions. While it is true that you can click on a node and see the corresponding proposition in the yellow space at the bottom of the screen, you can only do this for one node at a time, and thus, you can never "see the whole board." This makes constructing diagrams you have not previously constructed on paper very difficult. (5) Text cannot be modified once it is uploaded. When constructing an argument diagram, you want to balance faithfulness to the text with displaying propositions that can be independently understood. Thus, one often needs to modify slightly the original text, so that a proposition can stand on its own. (6) You can't have more than one screen up at any given time. This is a severe limitation for presenting a comparison of argument diagrams in class, as well as comparing alternate diagrams during the construction process. (7) You can't create diagrams of your own arguments spontaneously; the requirement of uploading text limits you to working with previously constructed statements.

Argutect

Argutect is a software package that was originally intended to help people in the business world better focus their thinking and give more informative presentations. To be fair, despite its name, it is not intended to be used for argument diagramming. Rather, it is intended to represent the process of thinking about an issue, where thinking is described as a set of questions and answers that may branch nonlinearly from the original question. This is what is called a "thought tree." Nonetheless, it does make the claim that any argument diagram can be turned into a thought tree, and this software does have similar appearances and functions with many packages that are specifically intended for constructing argument diagrams. An example of an Argutect thought tree is given in Figure 4.

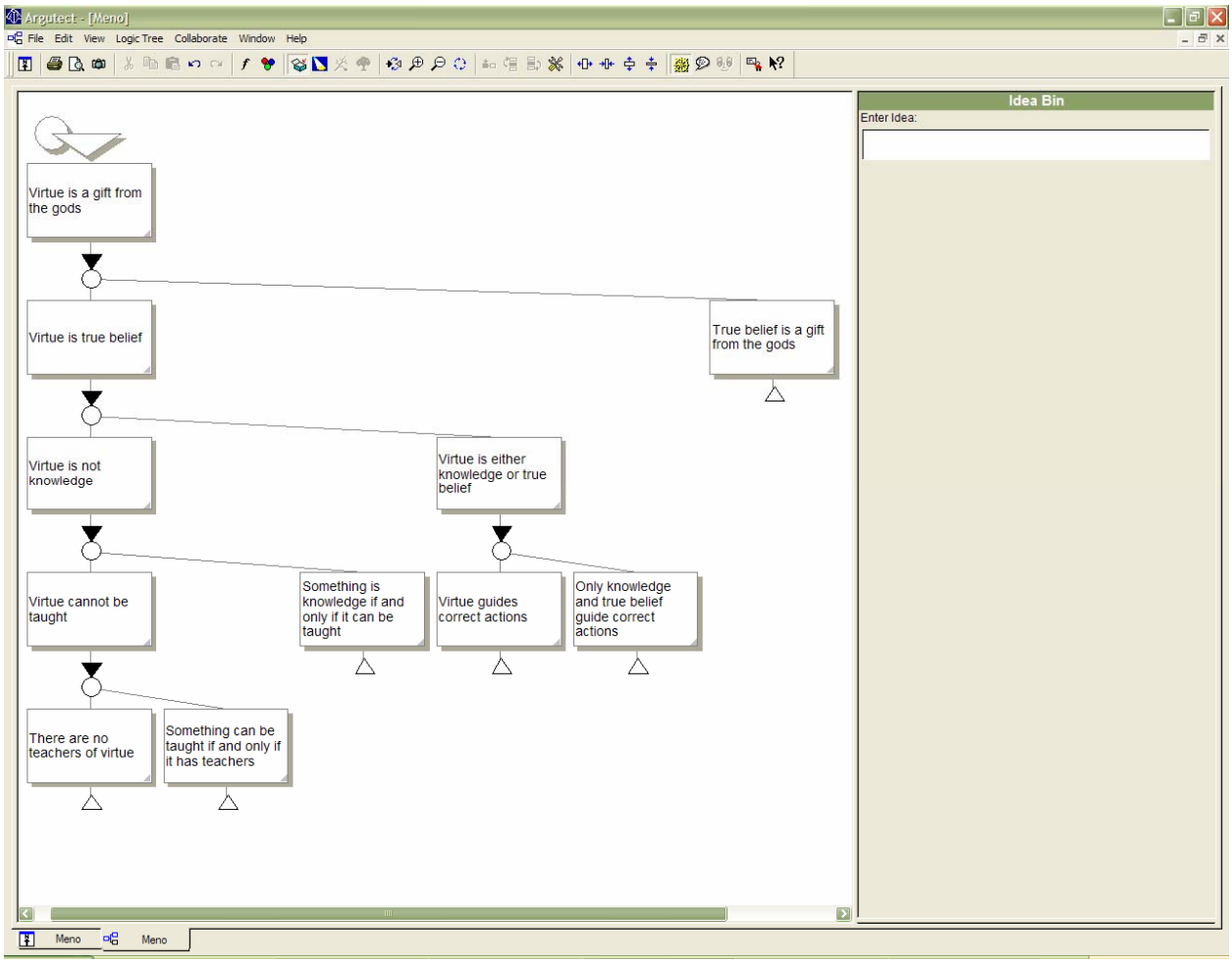

Figure 4 A screen shot of a thought tree produced by Argutect, representing the same argument from Plato's *Meno* that is represented in Figure 1.

The most prominent advantage of Argutect is that you can enter text into the diagram in two ways. You can create nodes of the argument, and then enter text directly into the box, or you can type claims into the "Idea Bin" to store them for later use (or both). This latter option is good for those who aren't sure how the propositions fit together and who want to lay them out before making the connections. Another good feature is that you can have more than one screen up at a time, so you can freely switch back and forth. In addition, each box can be independently colored, so you could have a coloring scheme that indicates main conclusion, reasons, objections, and replies.

The drawbacks of using Argutect for constructing argument diagrams are many. (1) The arrows point in the wrong direction. This is because what you are really creating are thought trees, and the arrows are supposed to point in the direction of the flow of question-and-answer thought. In an argument diagram, though, arrows are supposed to indicate the flow of support, so you want arrows pointing from premises to conclusions. (2) There is no way to represent a difference between linked and convergent arguments. Indeed, the diagram is not even clearly representing one or the other. (3) The boxes in the workspace are not independently expandable, so every box

has to be the same size. This could make for a very unwieldy diagram if you have just one box with a large amount of text. (4) In addition, although a box can be moved from supporting one box to supporting another, the boxes in the workspace are not generally moveable, so there is no flexibility in the layout of the diagram. The boxes supporting a conclusion are ordered across the screen from left to right in the order that they were created; thus, to change the left to right ordering, one must redo the creation of the premise boxes.

Athena Standard

Unlike Argutect, Athena Standard was developed specifically to create argument diagrams representing arguments found in philosophical texts. This package has considerable more flexibility in many ways than both Araucaria and Argutect, but it still has considerable drawbacks (see Figure 5).

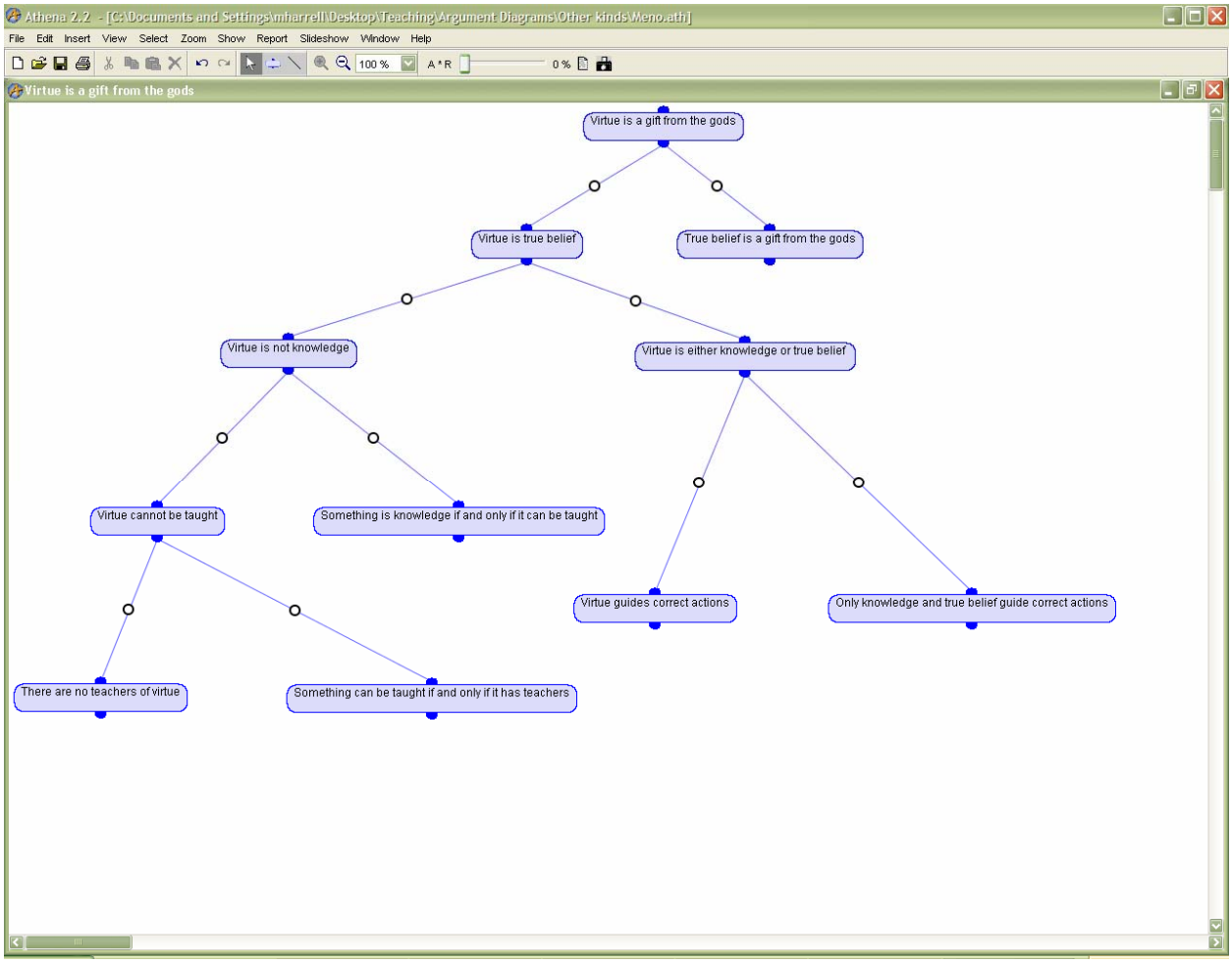

Figure 5 A screen shot of an argument diagram produced by Athena Standard, representing the same argument from Plato's *Meno* that is represented in Figure 1.

Athena Standard has several nice features. (1) You can create text boxes for claims independently on the workspace and move them around as you wish. This allows you to view propositions before deciding what are the connections between them. After creating two or more

text boxes, they can be linked by dragging a line from one to the other. The boxes can then be moved freely around the workspace while maintaining the connection you created. (2) The connecting lines can be the default color (indicating support) or changed to be red (indicating an objection). (3) You can have multiple windows open at once, so you can freely switch back and forth between different diagrams. (4) Athena Standard also offers ways to represent evaluations of the argument. For the text boxes, the default mode for Athena standard is to have, in addition to the text, indications of the acceptability of the claim (both as a percentage, and using color), as well as the relevant position of the box in the structure; for the connections the default is to have an indication of the relevance of one box to the box to which it has been connected. Additionally, these indicators can all be disabled, which gives a simpler, cleaner look to the diagram.

The drawbacks of Athena Standard, however, are significant. (1) The connecting lines are not arrows, so one can only assume which way the support is supposed to go based on the layout of the diagram. This limits the flexibility one has with moving the text boxes around the workspace. (2) As with Argutect, there is no way to represent a difference between linked and convergent arguments and the diagram is not even clearly representing one or the other. (3) While you can use red connections to indicate objections, you can only represent replies as further objections to the objections. In addition, the colors of the text boxes cannot be changed and so recognizing an objection in the diagram can be a very subtle task. (4) The visible text one can enter into a box is extremely limited. The boxes only expand sideways, so long propositions just lead to very wide boxes. This feature is due to the design the authors had in mind. When entering text, there is a space to enter long quotes or other propositions (but this text is not visible on the diagram), and there is a space to enter a “summary” of that proposition. This puts to onus on the you to be able to interpret the text as well as place it correctly in the structure of the argument. I would rather these be two separate tasks.

Inspiration

Inspiration is undoubtedly the most versatile software package of the group. And this is completely intentional; Inspiration was created to aid students (mostly), teachers, and professionals in their thinking in an extraordinarily wide array of possible projects. Because of this, there are very many template choices and you can just fill in the boxes with the appropriate text. Unfortunately, even with so many, there is no template for argument diagrams. The closest I found is the “Supporting Idea” template, but this has several drawbacks. You can, of course, create your own template, but I found it much easier just to use the basic workspace instead, as it allows you to create as many text boxes as you want and then draw the links between them, with the links being arrows by default (see Figure 6).

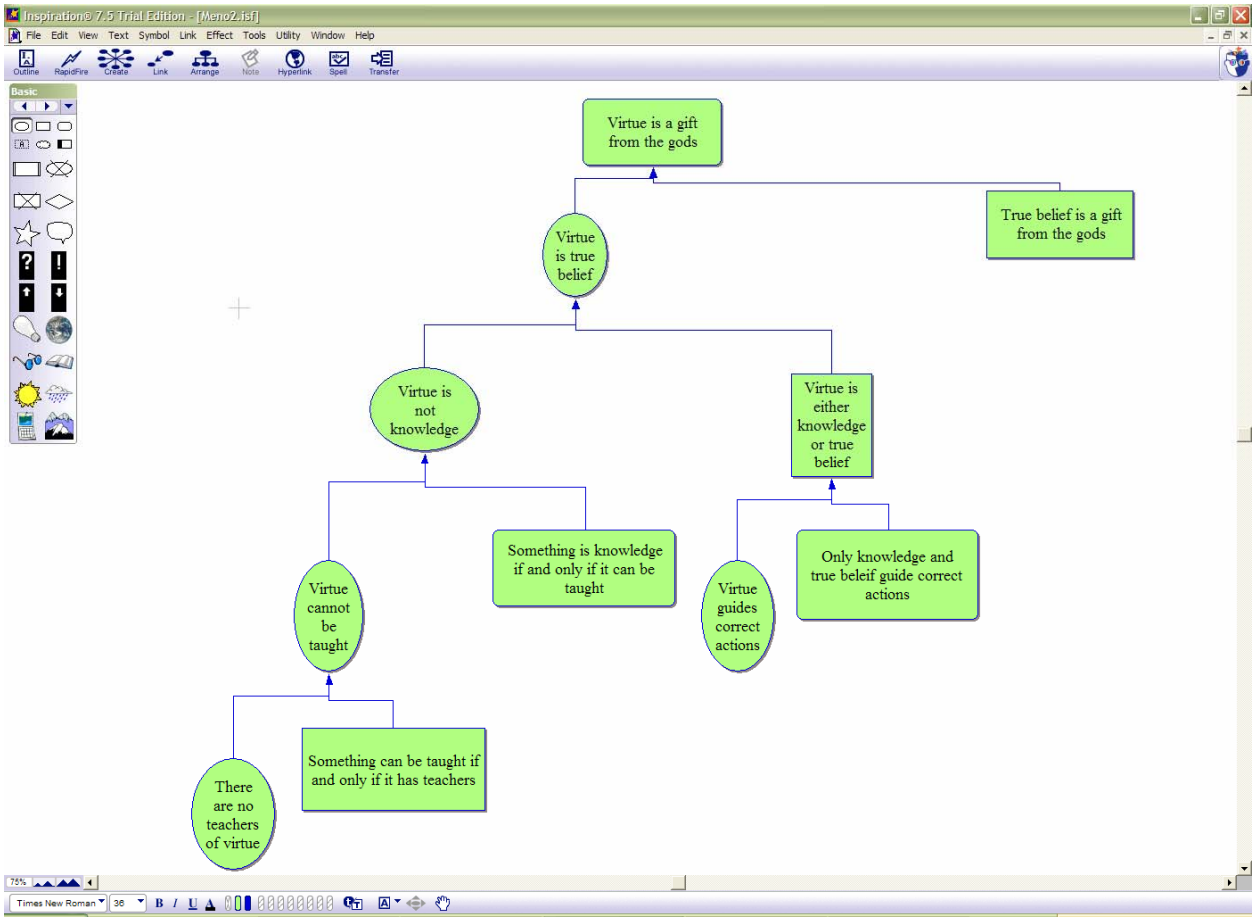

Figure 6 A screen shot of an argument diagram produced by Inspiration, representing the same argument from Plato's *Meno* that is represented in Figure 1.

There are many advantages to using Inspiration to create argument diagrams. (1) There is a wide variety of box shapes and fill colors, so reasons, objections, and replies, as well as the main conclusion, or even different kinds of premises can be differentiated. (2) In addition, the connections between boxes can be labeled and colored, so reasons, objections, and replies can be differentiated in an alternate or complimentary way. (3) You can change the shape and size of each box individually, or you can change them all at once. (4) There is a "Zoom" feature that allows you to keep the font constant, but make the diagram appear larger or smaller. (5) There is a spell checker, and at least in the "Supporting Idea" template, the "Arrange" function works nicely to "clean up the diagram." (9) You can add headers and footers to your file and print directly from the program, or you can transfer an image of the diagram directly into a Word document (although for large diagrams there is a little trouble with resizing to fit the margins; see Figure 7).

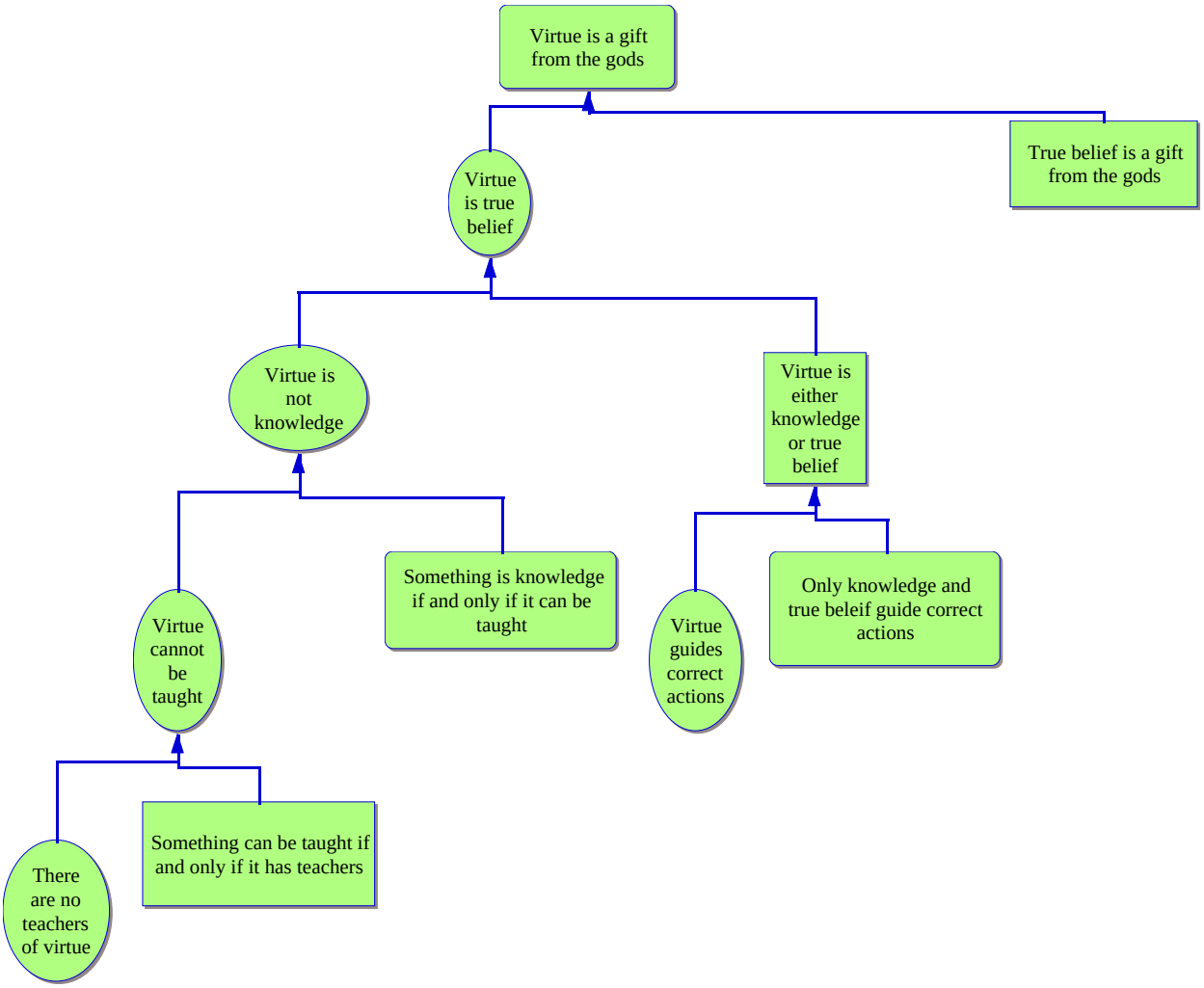

Figure 7 The image that is transferred from the Inspiration workspace into a Word document.

For all of these advantages, however, Inspiration has quite a number of drawbacks for my specific purposes. (1) When changing font sizes, the boxes don't automatically resize to accommodate the text. (2) You can manipulate the connections to represent both linked and convergent arguments, but this is not straightforward. (3) In the basic mode the "Arrange" function can destroy the structure you have created by moving boxes to odd places (however, there is an "Align" function that will allow you to "clean up" the diagram). (4) In the "Supporting Idea" template, creating links with arrows is a multi-step process. (5) The word document creating by using the "Transfer" function contains not only the diagram but also an outline, and for the purposes of an argument diagram, the outline generally does not make sense (it can readily be deleted, however). (6) In the basic mode, the program thinks that the box that that is pointed to by the arrow is the subordinate box, unlike in a regular argument diagram in which we consider premises to be subordinate to their conclusions; this only matters if you want to use the function that "hides subtopics," for this function will actually hide everything above the premise selected, instead of everything below.

Reason!Able

For the purposes of constructing and evaluating argument diagrams, Reason!Able is easily the best package I've encountered. Bram van Heuveln (2004) has already sung its praises on the pages of this journal, and I agree with him on all of these points, and so I will not go on at length about all its advantages. Suffice it to say that creating diagrams is simple and straightforward, as is adding objections and their reasons. There are also very good tools for representing the evaluation of the argument that are easy and helpful to use; you can evaluate each premise in the argument (both those that have support in the argument as well as those that do not), and you can evaluate the inferential connection between propositions. In addition, there is a printing option for the diagrams which can scale the diagram to fit the page if desired, as well as a way to put a label on these prints. In fact, I have been using Reason!Able for quite some time to keep an electronic record of all of the argument diagrams I have constructed (see Figure 8).

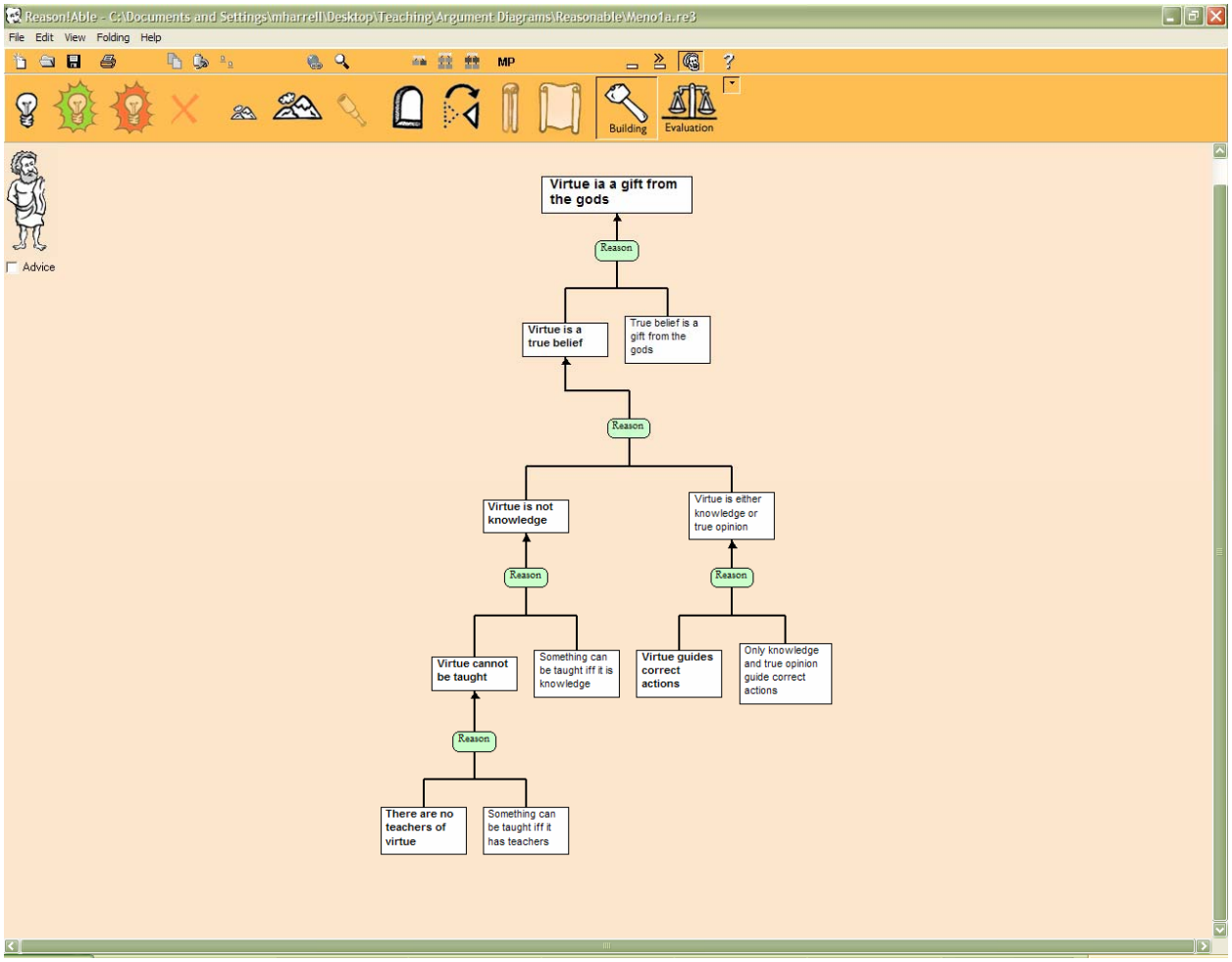

Figure 8 A screen shot of an argument diagram produced by Reason!Able, representing the same argument from Plato's *Meno* that is represented in Figure 1.

Despite all of its wonderful features, however, Reason!Able does have some significant drawbacks which prevent me from using it for demonstration in class. (1) It is slightly unstable;

there are certain keystroke combinations which do not do what they should, and I have lost many a diagram when trying to enter text. (2) In an effort to scale the text box to the size of the sentence inside, the program often breaks up words in strange places, making the statements difficult to comprehend at first glance. (3) The program automatically rearranges the text boxes of the diagram to minimize the space the diagram takes up, but this is not always desirable. As I noted above, many times, for both myself and my classroom demonstrations, I want the relationship of the premises in the diagram to roughly reflect the order in which they occur in the text. (4) The program also automatically bends the arrows connecting propositions as it is rearranging the text boxes to minimize the size of the diagram. This often leads to a very visually confusing diagram, especially for complicated arguments. (5) As with Araucaria, you can't have more than one screen up at any given time. As I said above, this is a severe limitation for presenting a comparison of argument diagrams in class, as well as comparing alternate diagrams during the construction process. (6) Also as with Araucaria, although you can represent objections to propositions, there is no way to represent a reply to that objection other than as an objection to an objection. (7) While it is possible to populate the workspace with independent text boxes with the relevant propositions, and then draw the arrows in later, this procedure is cumbersome, and it still requires you to know in advance which proposition is the main conclusion. I found it much easier, when working through a text, to construct the argument diagram on paper, and then create the resulting diagram in Reason!Able afterward. (8) Unfolding the reasons for a (sub-) conclusion so that you can include a "helping" premise is a good idea when creating a diagram of one's own arguments (especially for students), but this feature is not as helpful for representing arguments from a text. This is because the text often fails to include the helping premise, and to be faithful, we should not write it in; thus an argument diagram can be visually confusing with some green boxes and some white boxes all of which are "reasons." In addition, the program automatically makes the text of the "main" premise bolder than the text of the helping premises. This may make sense if the "helping" premise is truly merely helping, but more often, it is the case that two premises that need to be combined in order to support a conclusion are equally important.

Recommendations

I should emphasize that each of these software packages was created with a slightly different objective, one that I may not share. As such, none suit my purposes exactly, but this is to be expected; I very have specific ideas about the use of argument diagramming both for my demonstrations and for student use. These ideas, though, did not develop independently of my exploration of available software. On the contrary, it was through this exploration that I was able to reflect the most deeply on my own teaching practices and ultimately understand the most clearly what kind of software I can and cannot put to good use.

All of the software programs I have considered here can be found on the web, and downloads either of the complete package or of trial versions are available for free. Thus, it is relatively easy to experiment with any or all of the software. I recommend this both for teachers who are already using argument diagrams and want to investigate better ways to construct and present them as well as for teachers who have not thus far considered argument diagrams as a pedagogical tool. I have discovered tremendous value in using argument diagramming not only for my own thinking process, but also (and more importantly) for my students' thinking both about arguments they

read in primary source material, in newspaper articles, or online and about arguments they construct for themselves in essays and class discussions.

REFERENCES

Harrell, Maralee. 2004. "Using Argument Diagrams to Improve Critical Thinking Skills in What Philosophy Is" Technical Report CMU-PHIL-158. July 30, 2004.

Harrell, Maralee. 2005. "Using Argument Diagrams to Improve Critical Thinking Skills in Introductory Philosophy" submitted to *The Journal of the Learning Sciences*.

Plato (1976) *Meno*. Translated by G.M.A. Grube. Indianapolis: Hackett.

Twardy, C.R. (2204) Argument Maps Improve Critical Thinking. *Teaching Philosophy*, 27, 95-116.

van Gelder, Tim. 2001. "How to improve critical thinking using educational technology." In G. Kennedy, M. Keppell, C. McNaught, & T. Petrovic (Eeds.), *Meeting at the crossroads: proceedings of the 18th annual conference of the Australian Society for computers in learning in tertiary education* (pp. 539-548). Melbourne: Biomedical Multimedia Uni, The University of Melbourne.

van Gelder, Tim. 2003. "Enhancing deliberation through computer supported visualization." In P.A. Kirschner, S.J.B. Shum, & C.S. Carr (Eds.), *Visualizing argumentation: Software tools for collaborative and educational sense-making* (pp. 97-115). New York: Springer.

van Heuveln, Bram. 2004. "Reason!Able: An Argument Diagramming Software Package." *Teaching Philosophy* 27:2 (June): 167-172.

NOTES

I am grateful to Robert Cavalier for first introducing me to argument diagramming software, urging me to carry out this project, and providing very useful comments on earlier drafts.

[1] Araucaria v. 2.0, © Chris Reed and Glen Rowe, http://www.computing.dundee.ac.uk/staff/creed/araucaria/

[2] Argutect v. 4.0, © Knosis, http://www.knosis.com/argutect.html

[3] Athena Standard v. 2.2, © Bertil Rolf and Charlotte Magnusson, http://www.athenasoft.org/

[4] Inspiration v. 7.5, © Donald Helfgott and Mona Westhaver, http://www.inspiration.com/

[5] Reason!Able v. 1.1, © Tim van Gelder and Andrew Bulka, http://www.goreason.com/