

Improving critical thinking using web-based argument mapping exercises with automated feedback

Sam Butchart, John Bigelow, Graham Oppy
School of Philosophy and Bioethics
Monash University

Kevin Korb
School of Information Technology
Monash University

Ian Gold
Philosophy Department
McGill University

Abstract. In this paper we describe a simple software system that allows students to practice their critical thinking skills by constructing argument maps of natural language arguments. As the student constructs their map of the argument, the system provides automatic, real-time feedback on their progress. We outline the background and theoretical framework that led to the development of the system and then give a detailed example of how a student would work through a particular argument mapping exercise using the software. We then describe how the system was used in a single semester undergraduate critical thinking course. We evaluated the course using a standardised critical thinking test and measured an improvement in critical thinking skills of 0.45 standard deviations from pre-test to post-test; a modest, but encouraging result for a single semester course. We conclude the paper with some comments on the limitations of the system and ways in which it might be improved and extended.

1. Background

Critical thinking

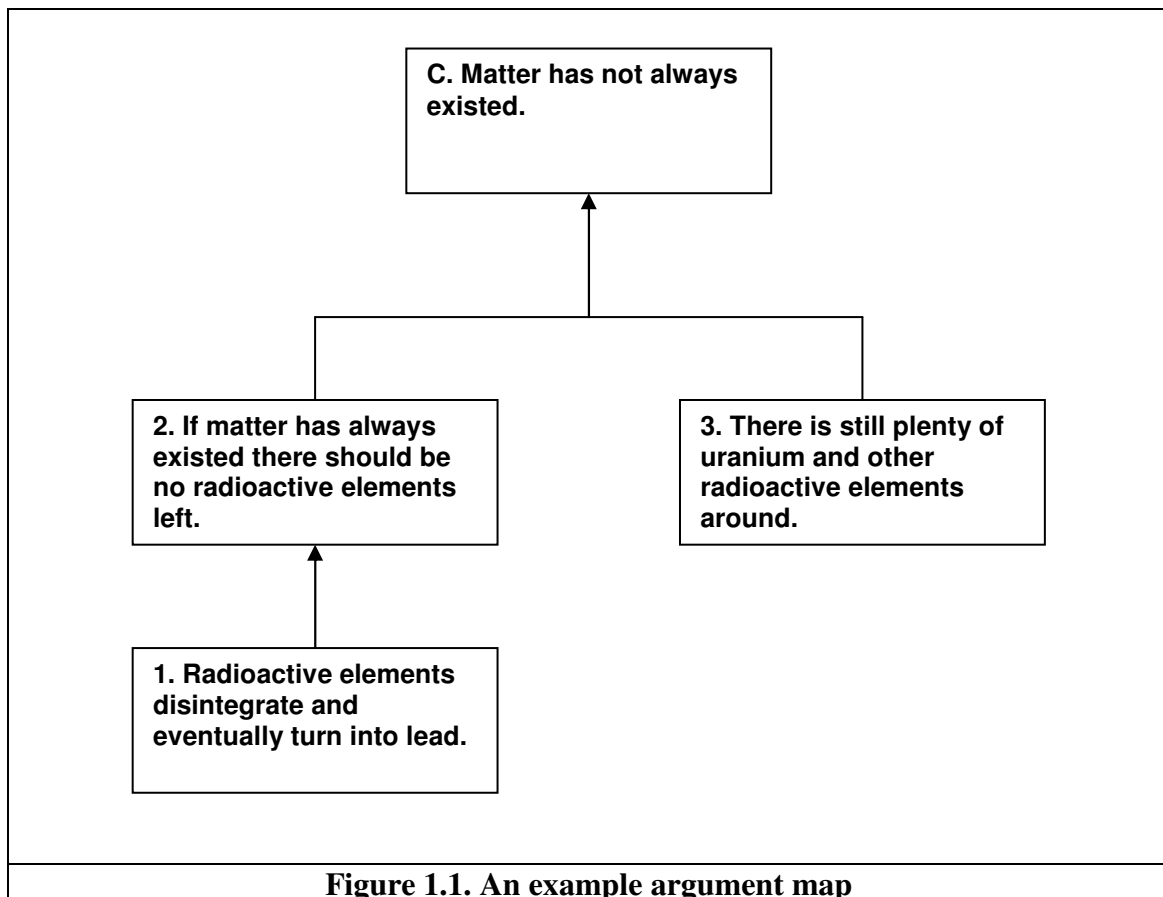
Many universities in Australia and elsewhere now offer undergraduate courses in *critical thinking* (also known as critical reasoning, informal logic and so on). The stated aim of such courses is to improve students reasoning skills. Typically, the focus is on the analysis and evaluation of real arguments taken from articles, books, opinion pieces, editorials, letters to the editor and so on. Students are taught to identify the components of an argument (premises, main conclusion, intermediate conclusions, unstated premises), analyse their structure (how the components fit together) and evaluate them using a variety of informal methods (for example, comparing them to lists of known fallacies). Students are also often encouraged to construct and evaluate their own arguments. The underlying assumption is that critical thinking – the ability and disposition to analyse and evaluate arguments in a variety of real-life contexts – is a skill that can be taught and improved. (See Ennis 1987 for a detailed analysis of the skills and dispositions commonly supposed to be involved in critical thinking).

Argument maps

A device that is often used in teaching critical thinking is the *argument map*. An argument map is a graphical representation of the logical structure of an argument – the ways in which premises, intermediate steps and the final conclusion all fit together. Consider, for example, the following argument (from Fisher, 2001):

Radioactive elements disintegrate and eventually turn into lead. If matter has always existed there should be no radioactive elements left. But there is still plenty of uranium and other radioactive elements around. This is scientific proof that matter has not always existed.

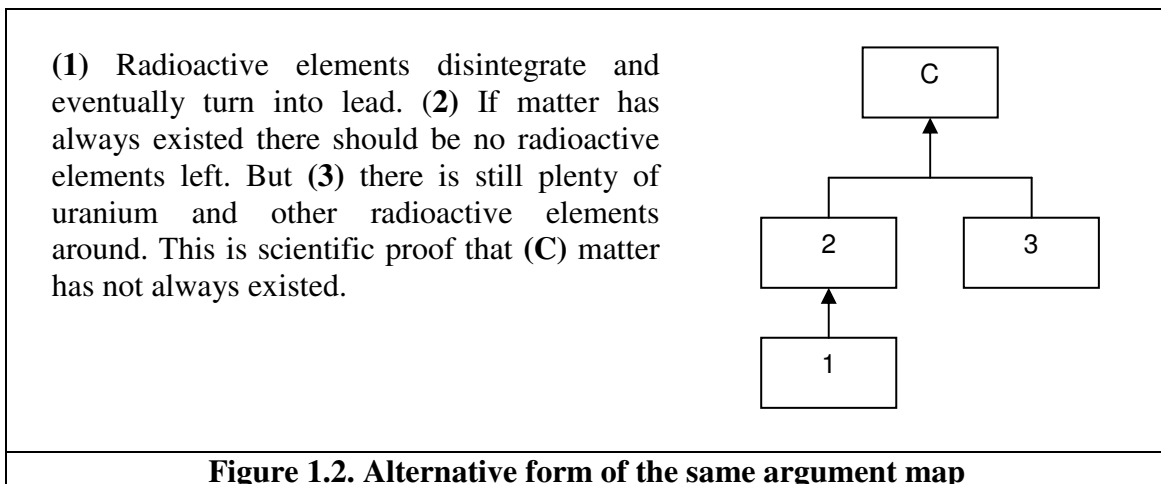
This argument can be concisely represented as an argument map as follows:



In these diagrams boxes represent the premises and conclusions and arrows indicate logical support. So in this case, premise 1 supports the intermediate conclusion 2, which in conjunction with premise 3 (shown by the line connecting 2 and 3) supports the main conclusion C.

An alternative format, which is often used because it requires less writing and takes up less space on the page, is to insert numbers into the text itself to indicate the

components of the argument and then use these numbers inside the boxes. The above argument, mapped in this way, would look like this:



Such diagrams have been used in critical thinking courses and textbooks since at least the 1950s. (Toulmin 1958, Mackie 1958, Scriven 1976, Geach 1976, Fisher 1988, LeBlanc 1998). Students are given an argumentative text to analyse and must create an argument map to represent its structure. This may sound like a trivial task, but it is not. Many students initially find it very difficult. It is in fact no easy task, when faced with even a short passage of ordinary argumentative text, to ‘extract out’ the core components of the argument. Even when students are able to do this, they can find it hard to see how the components fit together. Asking them to produce an argument map can provide the teacher with valuable insights into the student’s ‘mental model’ of the argument in question.

The most common mistake is to misidentify the relationship between premises like 2 and 3 in the above argument, which jointly support the main conclusion. Typically, students suppose instead that one premise supports the other (as is the case with premises 1 and 2 in the above argument), so they will place premise 3 underneath premise 2, or vice versa – a good indication that they have not properly understood the logical relationships holding between the components of the argument. With practice, however, most students improve.

Computer-assisted argument mapping software and deliberate practice

A recent innovation has been the use of computer software such as *Araucaria*, *Athena* and *Reason!able*, which assist students in the creation and manipulation of argument map diagrams. (See Kirschner et. al. 2002 for an overview). Such software allows students to easily create, manipulate and share complex argument maps – something which is not so easy using pencil and paper methods. Text can be typed into boxes and edited, supporting premises can be added, deleted or moved around and so on. In some systems, evaluations of the argument (assessments of the truth of premises and strength of inferential connections) can also be incorporated into the argument map. The results can be saved, printed out, shared online or pasted into an word-processing document.

Assuming that critical thinking is a skill that can be taught and improved, how can argument mapping software help achieve that goal? One view about how critical thinking skills can be improved is represented by the *Quality Practice Hypothesis* (van Gelder 2001a,b, 2004, 2005). According to this theory, acquiring expertise in critical thinking, as in many other areas, requires large amounts of *deliberate practice*. The concept of deliberate practice is based on research in cognitive science on how expertise is acquired in a variety of cognitive domains (see Ericsson and Charness 1994 and references cited in van Gelder et. al. 2004).

Deliberate practice must be *motivated* (the student should be deliberately practicing in order to improve their skills), *guided* (the student should have access to help about what to do next), *scaffolded* (in the early stages, it should be impossible for the student to make certain kinds of mistake) and *graduated* (exercises gradually increase in difficulty and complexity). In addition, for practice to be effective, sufficient *feedback* must be provided – the student should have some way of knowing whether they are doing the right thing or not. The quality practice hypothesis states that critical thinking practice must be deliberate in this sense to have any chance of improving students' performance (van Gelder 2004 et. al., p.143).

The use of computer assisted argument mapping exercises can help support extensive deliberate practice without expensive one-on-one tutoring. This is the fundamental idea underlying the design of the *Reason!able* argument mapping software. Students are provided with many ordinary language arguments to analyse and evaluate by creating a map of the argument using the software. Alternatively, students may be asked to construct an argument of their own on a given topic and produce a map of their argument.

The software itself supports the creation of these argument maps in a way that is both guided and scaffolded. (Gradually increasing difficulty and complexity is arranged for by the designer of the exercises). Scaffolding is provided by building certain constraints into the kind of diagram that can be produced. For example, every argument must have one and only one main conclusion and a strictly hierarchical structure of supporting premises (premises cannot support more than one conclusion). This is all built into the software, which simply does not allow students to create a map which violates these constraints. Guidance is provided by context-sensitive help – when the student selects a component of the argument map they are constructing, text appears providing some advice as to what to do next. (van Gelder 2001a,b. The latest version of the software is called *Rationale*. See Austhink 2006).

Using this approach to teaching critical thinking, van Gelder and others at the University of Melbourne have achieved impressive results. Over a single semester, 12 week course, they have recorded significant improvements in critical thinking, as measured by a standardised multiple-choice test (the California Critical Thinking Skills Test). Effect sizes for the mean gain from pre-test to post-test range from 0.8 to 0.89 standard deviations (van Gelder et. al. 2004).

The problem of feedback

A significant problem remains however – that of providing appropriate *feedback* to the student. Marking and grading argument maps produced by students is a time consuming process and requires a fair amount of expertise. In class, tutors can provide feedback to students on whether their argument maps are correct or not. With large classes of course, this can be difficult – there may not be enough time for tutors to give every student the feedback they need. One solution (adopted by van Gelder) is to provide model answers, so that students can assess themselves. Students compare their maps of a given argument to completed maps provided by the instructors. One problem with this is that students may not be able to work out *why* their answer is wrong and the model answer correct. Another problem arises from the fact that there is often more than one correct way to map a given argument. That being so, students may not be able to tell when a difference between their map and the model answer is an *important* difference.

Can feedback be automated? If it could, this would represent a substantial advance in the use of argument mapping software, by allowing for deliberate practice in a way that does not require a very low student-tutor ratio or hours of difficult marking.

In what follows, we describe one solution to this problem – a simple argument mapping software system which is able to automatically provide instant feedback to the student as they construct a map of a given argument. A prototype of the system was implemented in 2004 and incorporated into the Monash University critical thinking course. The system was implemented as a *Java Applet*, so that the argument mapping exercises can be provided to students over the web. We used the WebCT site for the course, but the system could be used to embed argument mapping exercises in any web page.

We will first describe the system, giving an example of how a student would work through one argument mapping exercise. We will then report on how the system was used in the critical thinking course. Finally, we comment on some of the limitations of the system and some possibilities for future enhancements.

2. An example automated argument mapping exercise

Figure 2.1 shows a simple example of an automated argument mapping exercise. The window in the top left hand corner contains the text of the following simple argument:

Large inequalities in wealth always threaten the viability of true democracy, since wealth is the basis of political power and true democracy depends on the equal distribution of political power among all citizens.

(LSAC 2002)

The student's task is to construct a map of the argument, using the mouse to select the appropriate segment of text and then clicking on the buttons below. The argument map gradually appears in the larger pane to the right.

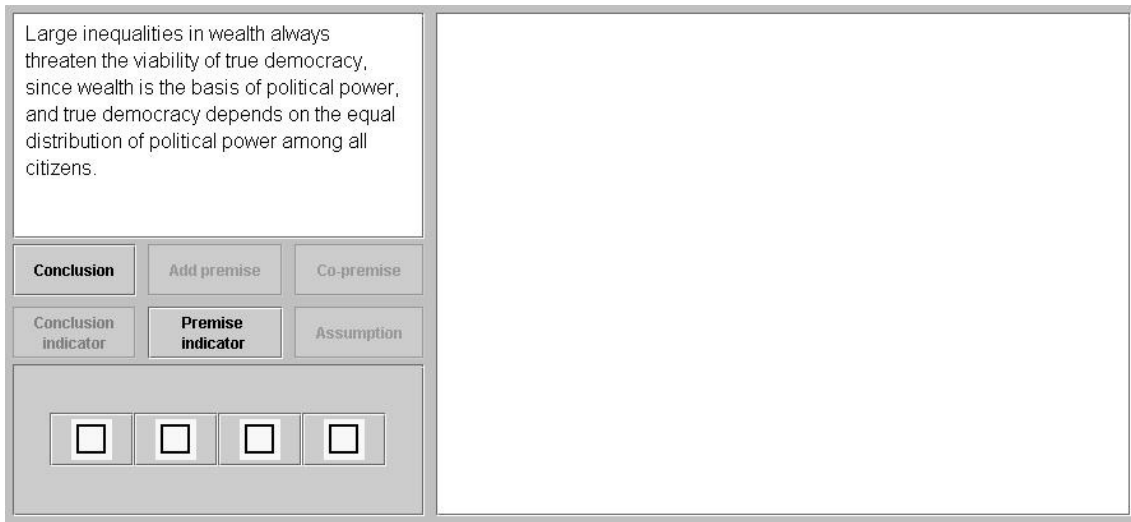


Figure 2.1. An argument ready for mapping

In figure 2.2, the student has selected the segment of text that they take to represent the conclusion of the argument. The student then clicks on the button labelled 'Conclusion' to indicate their choice. In this case, the student's identification of the conclusion is incorrect, so a small red cross appears in the box underneath the buttons. The student knows they must think again.

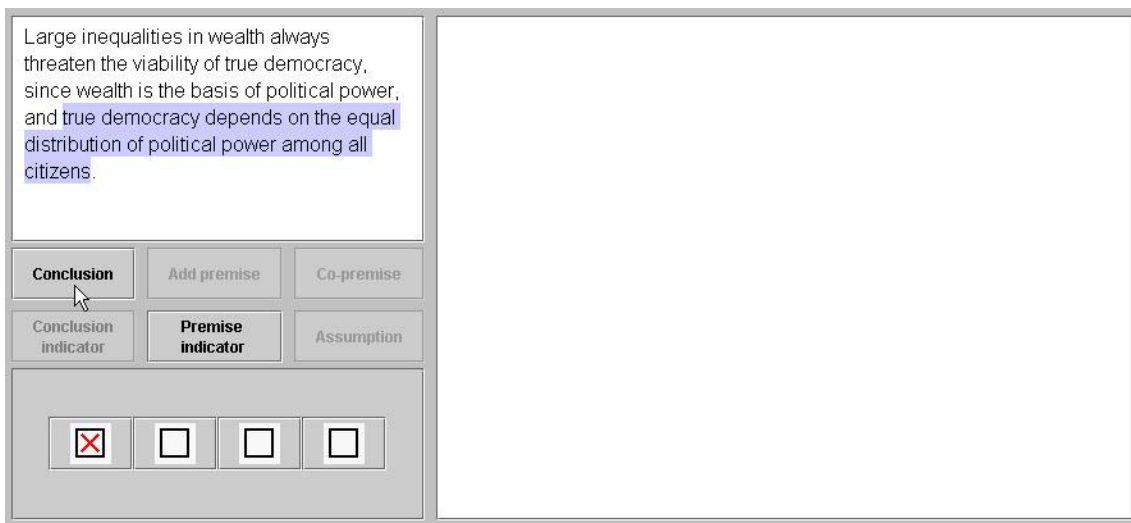


Figure 2.2. Conclusion misidentified

After re-reading the argument, the student selects the correct conclusion and clicks on the 'Conclusion button' again (Figure 2.3). The result is shown Figure 2.4. This time a green tick replaces the cross, so that the student knows they have correctly identified the conclusion. A box representing the conclusion appears in the right hand argument map pane and the text of the conclusion is highlighted in red.

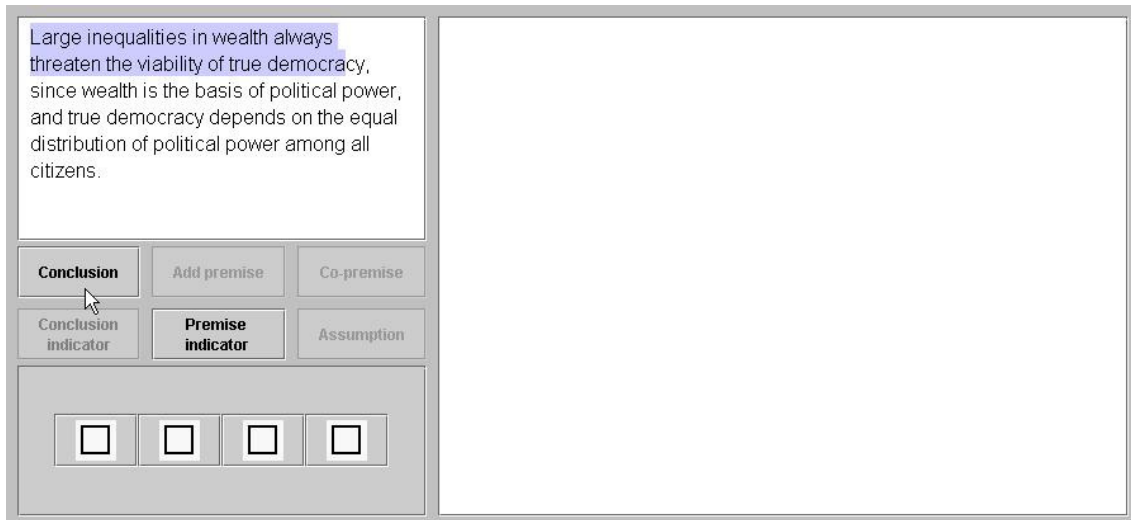


Figure 2.3. Correct conclusion selected

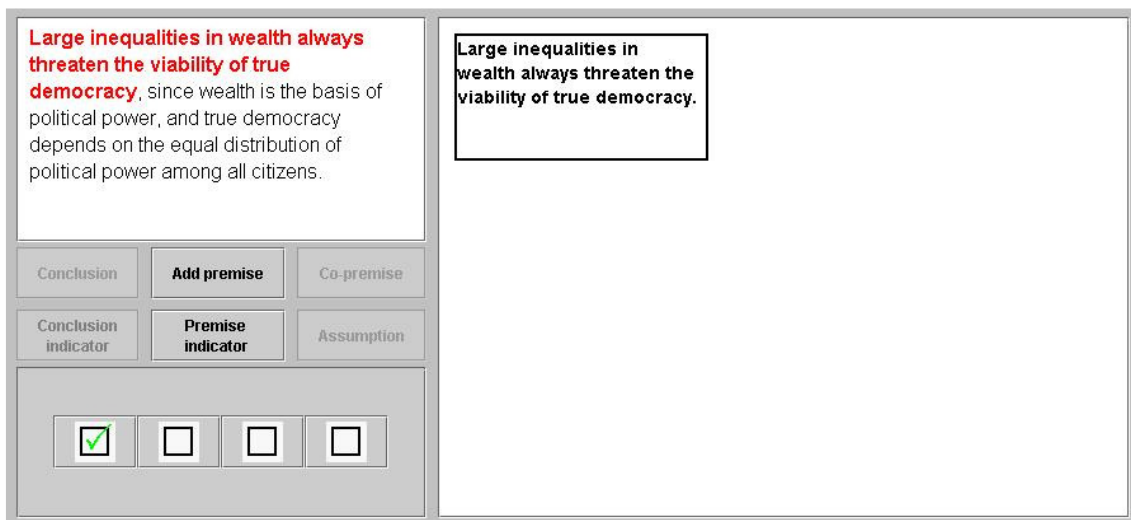


Figure 2.4. Conclusion correctly identified

This illustrates the mechanism for providing automated feedback to the student. The system knows which segment of the text represents each component of the argument, so it can instantly inform the student whether they have correctly identified that component or not. To avoid unnecessary frustration, the system allows for a fair amount of leeway in the selection of text. Notice, for example, that in Figure 2.3, the student has not selected all of the word ‘democracy’. The system automatically expands selections to the nearest word. If the student had only selected up to the word ‘true’ however, this would have been marked as incorrect – a premise or conclusion must always be a complete sentence.

Notice also that the ‘Conclusion’ button has now been disabled, since every argument has only one main conclusion. The previously disabled ‘Add Premise’ button is now enabled. This is one mechanism used to provide scaffolding and guidance – by selectively enabling and disabling buttons the student is guided as to what to do next and prevented from making mistakes.

The next step is for the student is to identify the premises supporting the main conclusion of the argument. In figure 2.5, the student has correctly identified the word ‘since’ as a *premise indicator*. Again, a green tick has appeared in the box to the right of the first tick, to indicate that this identification is correct (the word is then also underlined). The identification of the premise indicator provides a clue that that the text immediately following is a premise. The use of premise and conclusion indicators (words like ‘since’, ‘because’, ‘therefore’, ‘hence’, ‘it follows that ..’ and so on) to help students identify the components of an argument is a standard part of most critical thinking textbooks and courses. (See for example Fisher 2001, pp. 22-32, LeBlanc 1998, pp. 2-3).

Figure 2.5. A premise indicator identified

In figure 2.6 the student has selected all of the text following the word ‘since’ and clicked on the ‘Add premise’ button. This is not correct however – there are actually two separate premises in this example and the student’s selection has covered both of them. To map this argument correctly, each individual premise must be separately represented on the map. A red cross appears to indicate that the student has not correctly identified a premise of the argument.

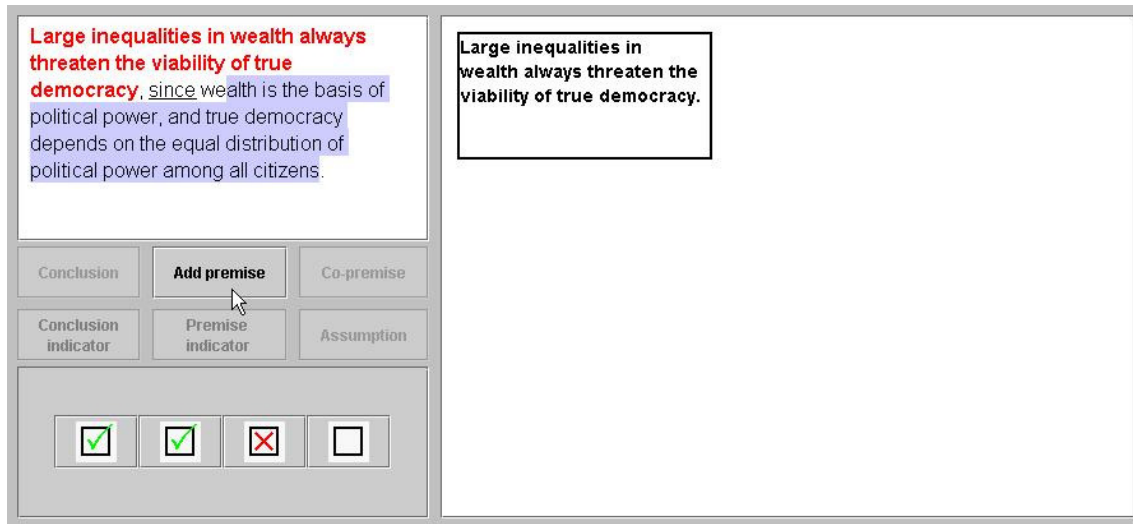


Figure 2.6. Premise misidentified

In figure 2.7, the student has now correctly identified one of the premises, by selecting the appropriate text and clicking on the 'Add premise' button. The function of this button is to add a supporting premise underneath the currently selected box in the argument map. A tick appears to indicate that this is correct and a box representing that premise is added to the argument map.

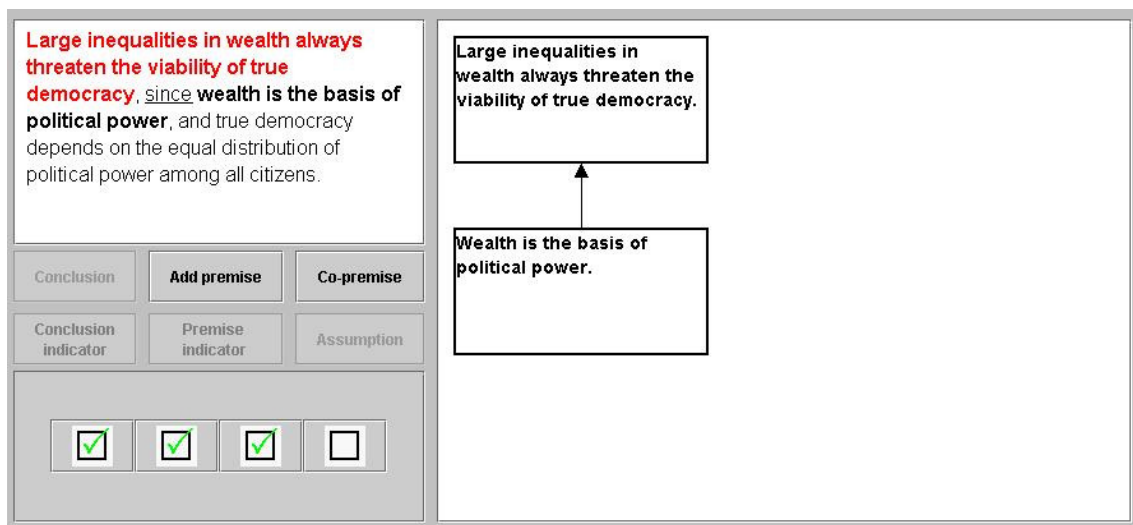


Figure 2.7. One premise correctly identified

The fact that one box has yet to be ticked off tells the student that they have one more component of this argument to identify. Since only two buttons remain enabled, the student knows that the remaining item is either a co-premise or a supporting premise.

In figure 2.8, the student has selected the appropriate segment of text and clicked on the 'Add premise' button. The function of this button is to add the selected text as a *supporting* premise below the currently selected box in the argument map pane (in this case, the premise 'Wealth is the basis of political power'). This choice is incorrect however; the selected text does not support that premise, but rather acts as a co-premise

supporting the main conclusion. So a red cross appears in the final box, to indicate that the student has made a mistake.

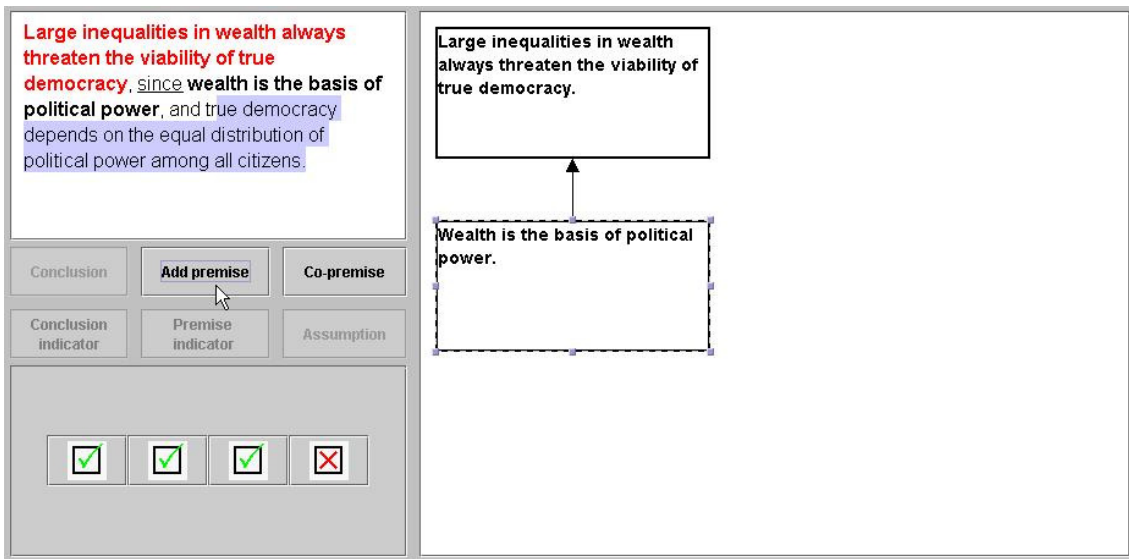


Figure 2.8. Final premise misidentified as supporting premise

Finally, in figure 2.9, the student has correctly identified the selected text as a co-premise. The co-premise is added to the argument map and a green tick appears in the final box, informing the student that they have completed this exercise and can go on to the next one.

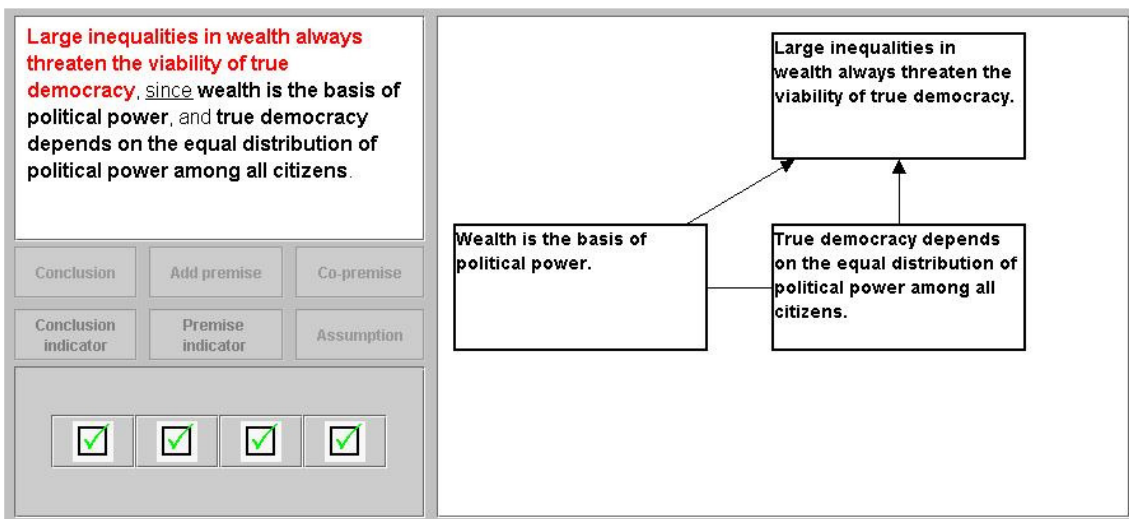


Figure 2.9. The argument fully mapped

It is worth noting that there is a fair amount of flexibility in the order in which components of the argument are identified. In this example, the student could have selected the text 'true democracy depends on the equal distribution of power ...' first and then added the co-premise 'wealth is the basis of political power'. However, the system does enforce a 'top-down' method of working – claims must always be

identified before any premise that supports them. But this may actually be an advantage, since it is one way of providing scaffolding and guidance to the student.

The system can handle maps of any degree of complexity, so that exercises can become more complex as students progress. An example argument map with a more complex structure is shown in figure 2.10. This example also shows how the distinction between linked premises (co-premises) and independent or ‘convergent’ premises is represented.

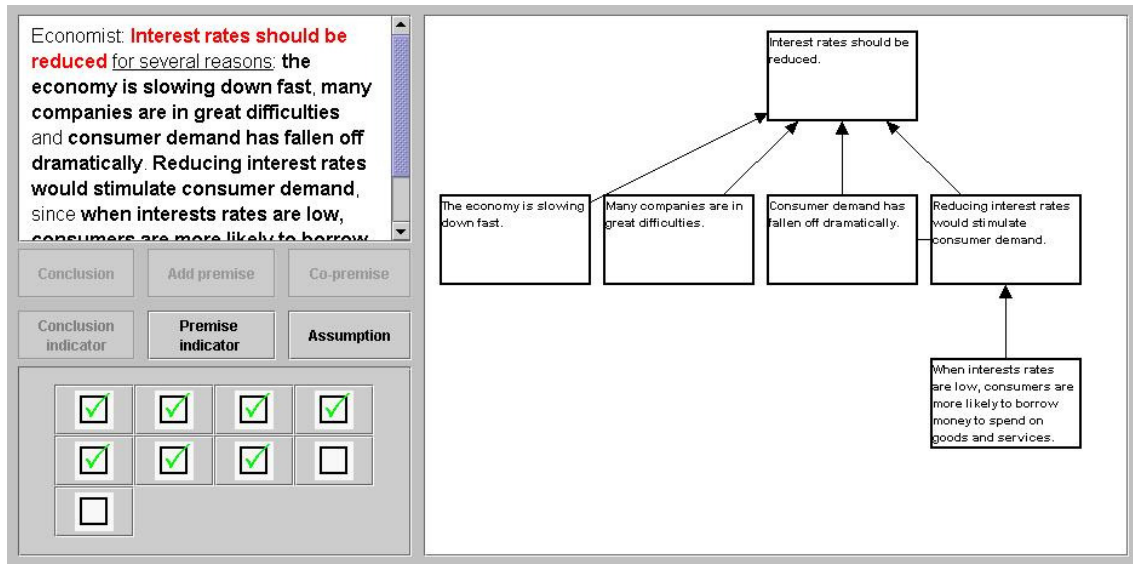


Figure 2.10. A more complex argument map

An additional feature is the ability to incorporate unstated premises (‘assumptions’) into the argument map. This is done by clicking on the ‘Assumption’ button and selecting the assumption from a multiple-choice list. Figures 2.11 and 2.12 show an example. (This example is taken from an LSAT ‘Logical Reasoning’ question, LSAC 2002).

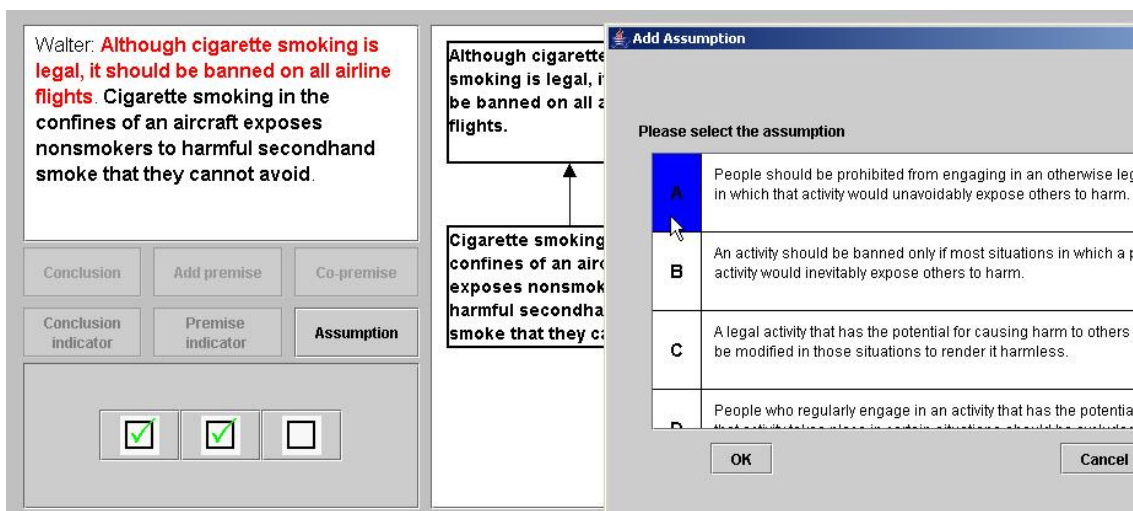


Figure 2.11. Adding an unstated premise (assumption)

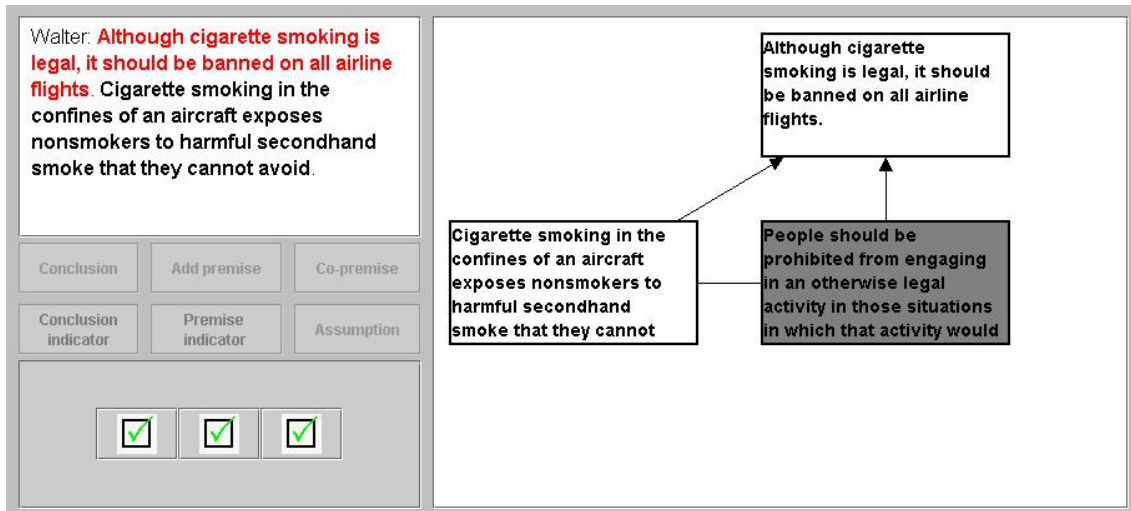
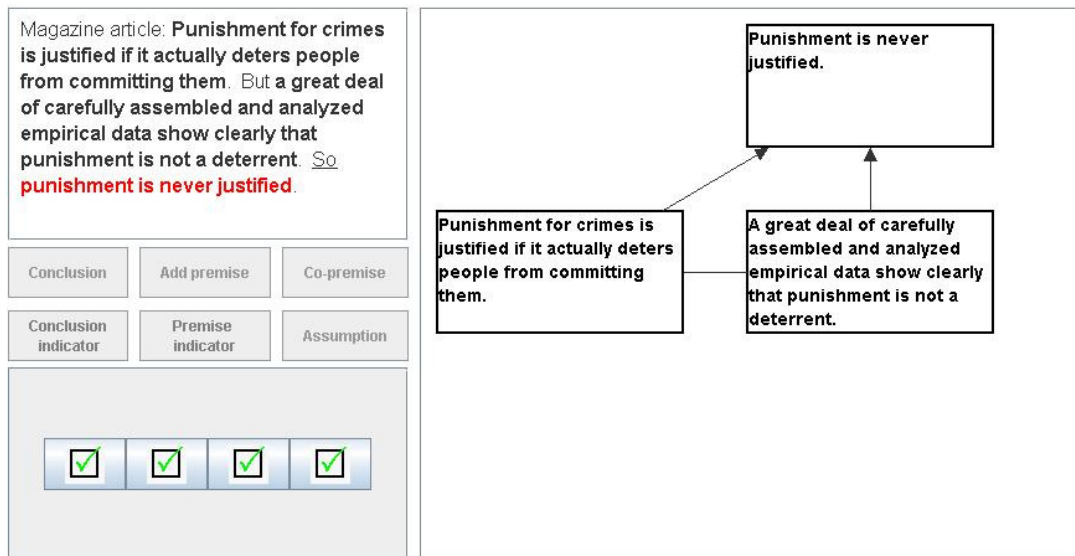


Figure 2.12. The assumption added to the map

It is also possible to include a multiple-choice question as part of the exercise. Students must first create a map of the argument and then answer a question about it. For example, the question might ask the student to identify a flaw in the argument. Again, instant feedback is given on whether the answer is correct. Figure 2.13 shows an example. (LSAC 2002).



The reasoning in the magazine article's argument is flawed because the argument

- (A) depends on data that there is reason to suspect may be biased
- (B) mistakenly allows the key term 'punishment' to shift in meaning
- (C) mistakes being sufficient to justify punishment for being required to justify it
- (D) ignores the problem of mistakenly punishing the innocent
- (E) attempts to be more precise than its subject matter properly allows

Incorrect.

Figure 2.13. An argument map exercise with multiple-choice question

The interested reader may care to try out some sample exercises using the system. These are available at:

<http://www.arts.monash.edu.au/phil/research/thinking/argumentmaps/>

3. Using the system in a critical thinking course

In semester one 2004, we incorporated these argument mapping exercises into a single semester critical thinking course, run by the School of Philosophy and Bioethics at Monash University (PHL1030. *Thinking: Analysing Arguments*).

3.1 Overview of the course

The unit is a single semester course, with 12 weeks of instruction, consisting of one 60 minute lecture and one 2-hour tutorial session each week. The course falls into two main parts. First, there is a section on argument *analysis* (identifying conclusions, premises, unstated assumptions and structure of arguments). This is followed by a section on argument *evaluation* (evaluating arguments, criticising arguments, identifying fallacies).

Assessment took the form of eight homework exercises. There was no final exam. Each homework exercises consisted of two sections: a set of LSAT multiple-choice logical reasoning and reading comprehension questions (LSAC 2002) followed by written analysis and evaluation of a short argumentative text.

Tutorials for the course took place in computer labs. Students spent the first half hour of each tutorial working on argument mapping exercises using the software. The remaining tutorial time was spent on further practice at analysing and evaluating example arguments. For this purpose, students were required to read one fairly short passage (essay or book extract) each week. The passages were on a wide variety of topics: philosophy, evolutionary theory, psychology, law, politics and so on. There were four tutorial groups, taught by two different tutors. Class sizes ranged from 12 to 17 students. For a week by week outline of the course, see Table 3.1.

3.2 Argument mapping exercises

Ten sets of exercises, consisting of 5-10 arguments for analysis were provided. These were made available on the course website, provided by WebCT. Students worked at their own pace and none of the exercises were graded. Students worked on the exercises by themselves, rather than in pairs or groups, but students were not prevented from discussing their work with their neighbour. On average, approximately 30-40 minutes each week were spent working on these exercises. The tutor was present to offer help if required. Since all the exercises were available online, students were able to complete the exercises at home if they did not finish them in class. Several students took advantage of this opportunity.

Table 3.1 lists the type of exercise used in each tutorial for the course. Note that the first exercise did not involve any argument mapping. In this exercise, students were

given a collection of passages and had to say which ones were arguments and which ones were not. The pre-test and post-test are explained in the following section.

Table 3.1. Course outline and argument mapping exercises used

Week	Lecture Topic	Tutorial Exercise
1	Why study argument?	Pre-test
2	Introduction to reason and argument	1. Identify arguments Distinguish arguments from non-arguments. 10 multiple-choice questions (no argument mapping exercises).
3	Argument analysis 1	2. Identify conclusions Identify the main conclusion of the argument. 12 argument mapping exercises.
4	Argument analysis 2	3. Map simple arguments Create a map of the argument. 11 exercises.
5	Argument evaluation: truth, justification	4. Map complex arguments Create a map of the argument. 14 exercises.
6	Argument evaluation: clarity, relevance, strength	5. Argument structure Map the argument and identify the role played by a particular statement or the argumentative strategy used. 14 exercises with multiple-choice questions.
7	Criticism: Objections and replies	6. Argument structure As above. 14 exercises with multiple-choice questions.
8	Criticism: Assumptions	7. Assumptions Map the argument, selecting the correct assumption from a list. 8 exercises.
9	Fallacies (Ambiguity)	8. Fallacies of ambiguity Map the given argument, then answer the question to identify the flaw. 8 exercises with multiple choice questions.
10	Fallacies (Relevance)	9. Fallacies of relevance Map the given argument, then answer the question to identify the flaw. 8 exercises with multiple-choice questions.
11	Fallacies (Truth, ambiguity)	10. Fallacies of truth Map the given argument, then answer the question to identify the flaw. 8 exercises with multiple-choice questions.
12	Fallacies (Strength)	11. Fallacies of strength Map the given argument, then answer the question to identify the flaw. 8 exercises with multiple-choice questions.
13	Reason and happiness	Post-test

4. Gains in critical thinking skills

As part of a continuing attempt to investigate and compare alternative teaching methods for improving critical thinking, students enrolled in this course were pre- and post-tested using the California Critical Thinking Skills Test (CCTST). This is a timed (45

minute) multiple-choice test, coming in two equivalent forms A and B. Each form consists of 34 items which test students' ability to clarify the meaning of claims, analyse and evaluate arguments and draw correct conclusions from given information. (Facione and Facione 1992).

4.1 Sample and procedures

The final sample consisted of 43 undergraduate students (16 men and 27 women). Ages ranged from 17 (1 student) to 55 (1 student). The mean age was 21.5 years, while the mode was 18 years (30.2%). All students taking the course were required to complete the pre- and post-test. They were informed about the purpose of the study and asked to sign a consent form giving permission for their test scores to be used. Test scores did not count towards the students' final grade.

Out of an initial enrolment of 63 students, 52 (82.5%) consented to be part of the study. Of the students who consented 50 (96%) completed the pre-test and 43 (83%) also completed the post-test. Of the 9 students who failed to complete both tests, 7 completed the pre-test but not the post-test and 2 completed neither test.

Students completed the CCTST during the first half of the scheduled 2-hour tutorials for the course. The pre-test was completed in the first tutorial (week 1) and the post-test in the final tutorial (week 13). The tests were completed under examination conditions, as outlined in the test manual. Students were not informed of their test scores until after the end of the course. All students were given form A of the CCTST for the pre-test and form B for the post-test.

There is some evidence that form B of the CCTST is somewhat harder than form A (Jacobs 1995, Hitchcock 2004). For example, randomly distributing forms A and B to incoming freshmen, Jacobs measured a form A mean of 16.01 (n=684) and form B mean of 15.36 (n=692). This pattern is supported by our own studies, carried out subsequent to that reported here, in which we distributed test forms randomly. Given that form B is slight harder, using it as the post-test would tend to provide a more conservative underestimate of the magnitude of any gains in critical thinking ability, rather than an overestimate.

4.2 Results

Table 4.1 shows the pre- and post-test means for the 43 students who completed both tests. The pre-test mean was 18.209 (54%) while the post-test mean was 20.233 (60%). A standardised effect size for gains in critical thinking ability was calculated by dividing the mean gain in raw CCTST score (2.02) by the estimated population standard deviation of 4.45. (This is the value used in other studies using the CCTST, such as van Gelder et. al. 2004). Students showed a statistically significant improvement in critical thinking of 0.45 standard deviations (95% confidence interval = 0.17, 0.74). This gain is significantly greater than zero ($t = 3.202, p < 0.05$, two-tailed).

Table 4.1. Improvement in critical thinking scores on the CCTST

	Mean <i>n</i> = 43	95% confidence interval	Standard deviation
Pre-test	18.209 (54%)	[16.73, 19.69]	4.8
Post-test	20.233 (60%)	[18.6, 21.87]	5.32
Gain	2.02	[0.74, 3.29]	4.14
Effect size	0.45	[0.17, 0.74]	

4.3 Comparison with other studies

To get an idea of how these results compare with other single-semester critical thinking courses, Figure 4.1 shows results from a number of studies, all of which used the CCTST as a pre- and post-test measure of critical thinking ability. Study 1 can be used as an estimate of the amount one might expect students to improve over a single semester of university study, without any specific critical thinking instruction. These are the results from 90 students enrolled in a single semester introductory philosophy course at California State University at Fullerton (Facione 1990). The mean gain for these students was 0.14 standard deviations on the CCTST (although this gain fails to be significantly different from zero at the 0.05 level). The mean gain for the Monash argument mapping group (Study 4) was more than three times higher, at 0.45 standard deviations. Although this difference in mean gain scores of 0.31 standard deviations fails to be significant at 0.05 level (95% confidence interval for the difference = -0.03, 0.65), it is significant at the 0.1 level (90% confidence interval for the difference = 0.03, 0.6).

More directly relevant are the results from the following semester at Monash University (Study 2). The following semester of the same year, we taught the course again, without using argument mapping software. Students were pre- and post-tested using the same procedures as described above. The mean gain for that semester was 0.19 standard deviations ($n = 65$, 95% confidence interval = -0.01, 0.39). The difference of 0.26 standard deviations between this result and the gain measured using our argument mapping software was not significant at the 0.05 level (95% confidence interval for the difference = -0.07, 0.6).

At California State University at Fullerton, a single semester critical thinking course lead to a gain of 0.32 standard deviations (Study 3, Facione 1990). At McMaster university Hitchcock measured a similar gain to ours of 0.49 standard deviations for a 12-week course using the LEMUR software (Study 5, Hitchcock 2004). The LEMUR software accompanies Jill LeBlanc's textbook (Le Blanc 1998) and incorporates numerous multiple-choice quizzes and exercises. The software also includes some simple argument mapping exercises in which students can drag component sentences of a text into a pre-structured argument map diagram (this idea is also used in the software accompanying Hurley 2005).

The gains in critical thinking obtained using extensive practice with the *Reason!able* argument mapping software are significantly higher, at 0.8 standard deviations (Study 6, van Gelder et. al. 2004). The difference between this result and the 0.45 gain measured at Monash is statistically significant (95% confidence interval for the difference = 0.026, 0.675).

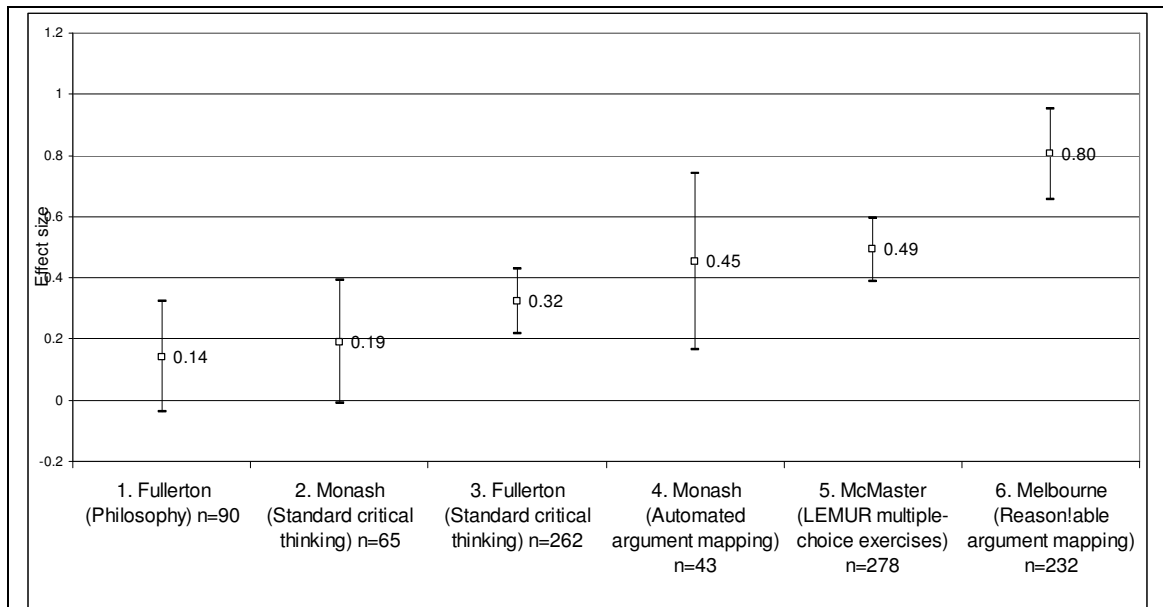


Figure 4.1. Gains in critical thinking ability for single-semester critical thinking courses. (Bars show 95% confidence intervals).

1, 3: Facione 1990, 5. Hitchcock 2004, 6. van Gelder et. al. 2004.

4.4 Discussion

Although there was a statistically significant improvement in students' critical thinking test scores, caution is clearly required in interpreting this result. It is unclear how much of that improvement was due to the use of the automated argument mapping exercises and how much is due to other aspects of the course or to factors not directly related to teaching such as maturation.

A comparison with the studies described above is suggestive however. Firstly, the improvement in critical thinking scores obtained using automated argument mapping exercises (0.45 standard deviations) is more than three times as great as the gain of 0.14 standard deviations measured for a single semester philosophy course with no explicit critical thinking instruction. This suggests that the improvement was due to the instruction students received, rather than simply to maturation. Furthermore, the improvement is more than twice as large as that obtained the following semester, teaching the same course without the automated argument mapping exercises (0.19 standard deviations), suggesting that most of the improvement is in fact due to the argument mapping exercises. These comparisons are no more than suggestive however;

due to the small sample sizes involved, no statistically reliable inference can be drawn from these comparisons.

Nonetheless, the gains in critical thinking for the course as a whole are encouraging and compare favourably to gains obtained in similar studies. Informal feedback from students on the use of the system was fairly positive: several students reported finding the argument mapping exercises useful and a few took up the opportunity of completing the exercises from home.

The result certainly seems *consistent* with the Quality Practice Hypothesis, even if it does not strongly support it. Students using the system certainly got a lot more guided and scaffolded practice than students the following semester and the students who had more practice showed a statistically significant improvement in critical thinking scores, while the latter group did not. Again however, no reliable inference to the conclusion that the practice caused the improvement can be drawn.

The mean gain in critical thinking scores obtained using our automated argument mapping system is significantly lower than that obtained using the *Reason!able* argument mapping software at the University of Melbourne – a difference of 0.35 standard deviations (95% CI = 0.026, 0.675). How can this difference be explained? We can do no more here than offer some possible explanations.

Firstly, the *Reason!able* software is much more flexible than our system in several ways. Most obviously, it allows students to put any text at all into argument map boxes, rather than requiring them to select a segment of text. It also allows students to edit the structure of their maps as they build them, by moving and dragging boxes into new positions. Students can also build up their argument maps in any order they like; they can start with the conclusion and work down to the premises, or they can start with the premises and build up to the conclusion. This extra flexibility may well make a big difference to how much students can learn from argument mapping exercises.

Secondly, *Reason!able* has an ‘argument evaluation’ mode, as well as an ‘argument mapping’ mode. This allows students to add information to their maps representing their judgements about the plausibility of premises and the strength of inferential connections. (In our system, argument evaluation practice was provided using multiple-choice questions, which ask the student to identify flaws in the argument). It may be that this ability to incorporate argument evaluations into the argument maps constructed by students plays a significant role in the success of the *Reason!able* system.

Thirdly, there are additional factors, not directly related to the software itself, which may account for the difference between the Melbourne results and our own. One difference is that at Melbourne, students work together in groups of two on their argument maps. In the present study, students worked on the argument mapping exercises on their own. It may be that the collaboration and peer interaction that become possible when students work together provides a significant boost to the power of the argument mapping method.

Finally, the sheer number of hours of practice with argument mapping exercises is

somewhat greater for the Melbourne university course. Students at Melbourne spent (on average) a total of 10 hours on argument mapping using the *Reason!able* software over a 12-week period (van Gelder et. al. 2004, p. 147, Table 1). The Monash students, on the other hand, spent a total of 5-7 hours working on automated argument mapping exercises over the same period (30-40 minutes on argument mapping exercises for each of ten tutorials). It seems quite possible then that the difference between the Melbourne University result and the Monash result is due to the amount of quality practice provided. Indeed, van Gelder’s study found a positive correlation ($r = 0.31$) between the number of hours of practice (as measured and logged by the software itself) and gains in critical thinking as measured by the CCTST (van Gelder et. al. 2004, pp. 147-8).

5. Limitations and future development

We conclude with some brief comments on the limitations of the system and possibilities for future extensions and improvements.

5.1 Making the feedback more informative

One obvious way in which the system could be improved would be to make the feedback to the student more informative than a simple ‘correct’ or ‘incorrect’. In particular, since the system has a complete representation of the correct argument map, it would be possible to distinguish between various kinds of common mistake that students make when constructing argument maps. For example, students will often make the mistake of placing a claim *underneath* a given premise, when the claim should be a co-premise. The opposite kind of mistake – representing a claim as a co-premise, when it should be a supporting premise – is also possible. All these sorts of mistakes and others could be captured by the system quite easily and a more informative message delivered to the student. One way to do this is shown in Figure 5.1

The screenshot shows the Reason!able software interface. On the left, a text box contains a paragraph about radioactive elements. Below it are buttons for 'Conclusion', 'Add premise', and 'Co-premise'. A mouse cursor is over 'Add premise'. Below these are buttons for 'Conclusion indicator', 'Premise indicator', and 'Assumption'. At the bottom left, a feedback message reads: 'Incorrect. That premise does not support the currently selected claim. Hint: Could it be a co-premise? Score: 2 out of 6. (Attempt: 3)'. On the right, an argument map is shown with a box containing 'Matter has not always existed.' and a dashed box containing 'There is still plenty of uranium and other radioactive elements around.' with an arrow pointing up to the first box.

Figure 5.1. More informative feedback

Instead of a sequence of ticks and crosses, we now have a text pane which displays information to the student on their progress with the argument map. In this example, the student has correctly identified the main conclusion and one of the premises. But the student has then misidentified the claim 'If matter has always existed there should be no radioactive elements left' as providing support for that premise. The system has identified the mistake and displays an appropriate message and a hint.

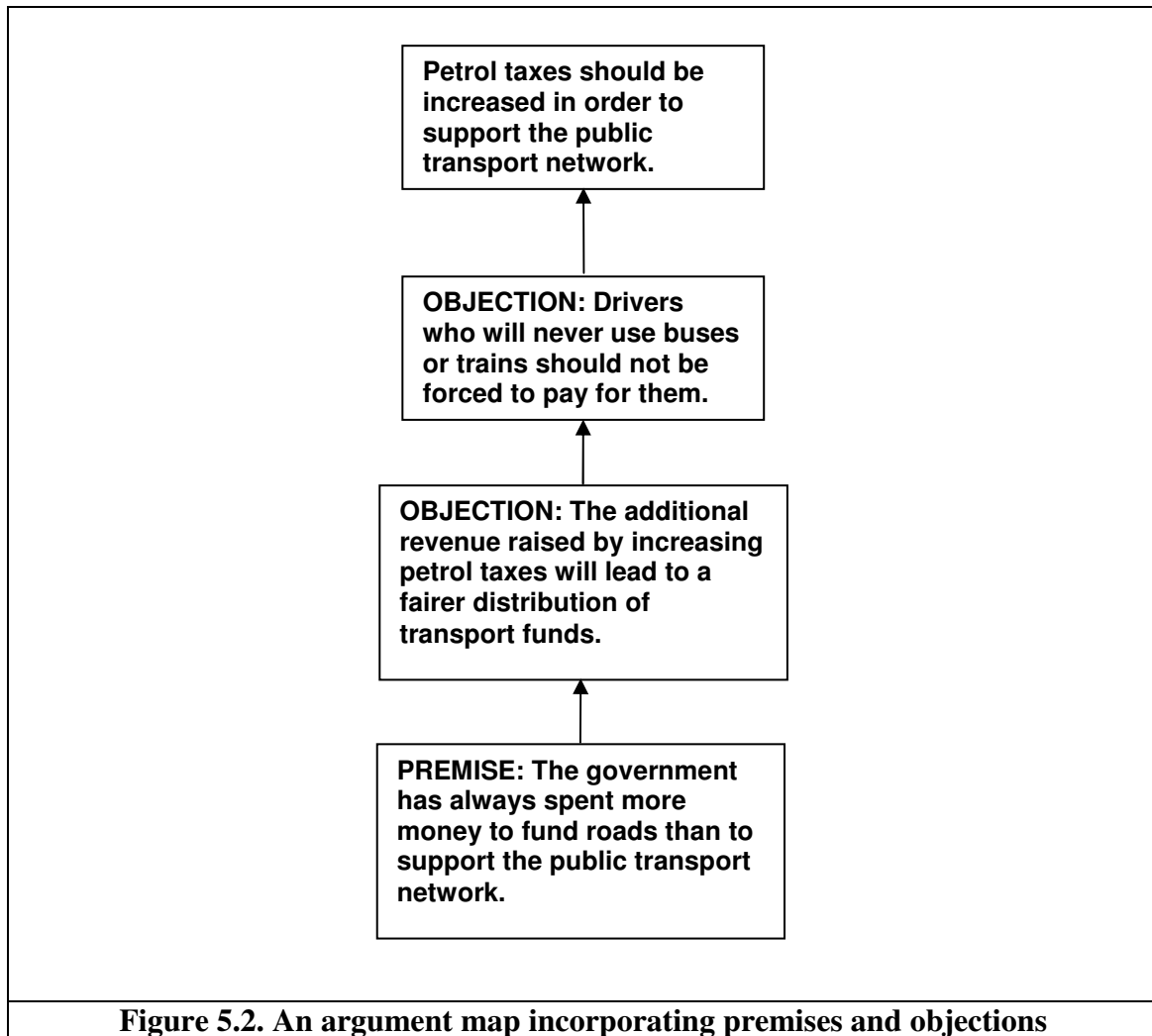
In addition to giving more informative feedback about the student's mistakes, this text pane could also be used to display context sensitive help about what to do next as the student constructs their argument map. There could also be an option to turn off the hints and the help once students feel they no longer needed it. In this way, a further aspect of quality practice would be incorporated – gradually removing scaffolding as the student becomes more expert.

5.2 More flexibility in constructing maps

The current system enforces a fairly strict ordering on the way student's construct argument maps. The conclusion must be identified first, followed by premises that directly support it, followed by premises that support those premises and so on. Sometimes however, it is more natural to construct an argument map from the bottom up, starting with the premises and building up to the main conclusion. The system could be made more flexible by allowing students to construct their maps in either of these two ways.

5.3 Objections

In addition to premises that *support* a claim, argument mapping systems often include the concept of an *objection* to a claim. This is especially useful when mapping a complex debate, consisting of arguments, counter-arguments, rebuttals and so on. (See Figure 5.2). It would be straightforward to add objections to argument maps in the current system. An 'Add objection' button could be used to add an objection, rather than a supporting premise under the currently selected node in the argument map pane. Objections could then be distinguished from supporting premises in the argument map using different colours or by adding a simple prefix to the text (as in figure 5.2).



5.4 Evaluations

Another limitation of the system described here is that it does not allow students to include evaluations of arguments directly in their maps. (Instead, we included multiple-choice questions asking students to identify flaws in the argument). There are two aspects to argument evaluation; evaluation of premises and evaluation of inferences. Both aspects can be incorporated into argument maps. One way to do this is to ask students to rate premises and inference on a scale. For premises, the scale might range from 'definitely false' to 'probably false' to 'probably true' to 'definitely true'. For inferences, the scale might range from 'definitely does not support', to 'weak support' to 'strong support' to 'conclusive support'. These judgements could then be shown in the argument map diagram itself. (See Figure 5.3).

This capability could be added to the system described here by adding 'evaluate premise' and 'evaluate inference' buttons. The student selects a box in the argument map pane and then clicks on the 'evaluate premise' button. A dialog box then appears from which the student can select one of range of possible evaluations from a fixed

scale. If appropriate the system can then provide feedback on the student's selection. Evaluating inferences would work in a similar way.

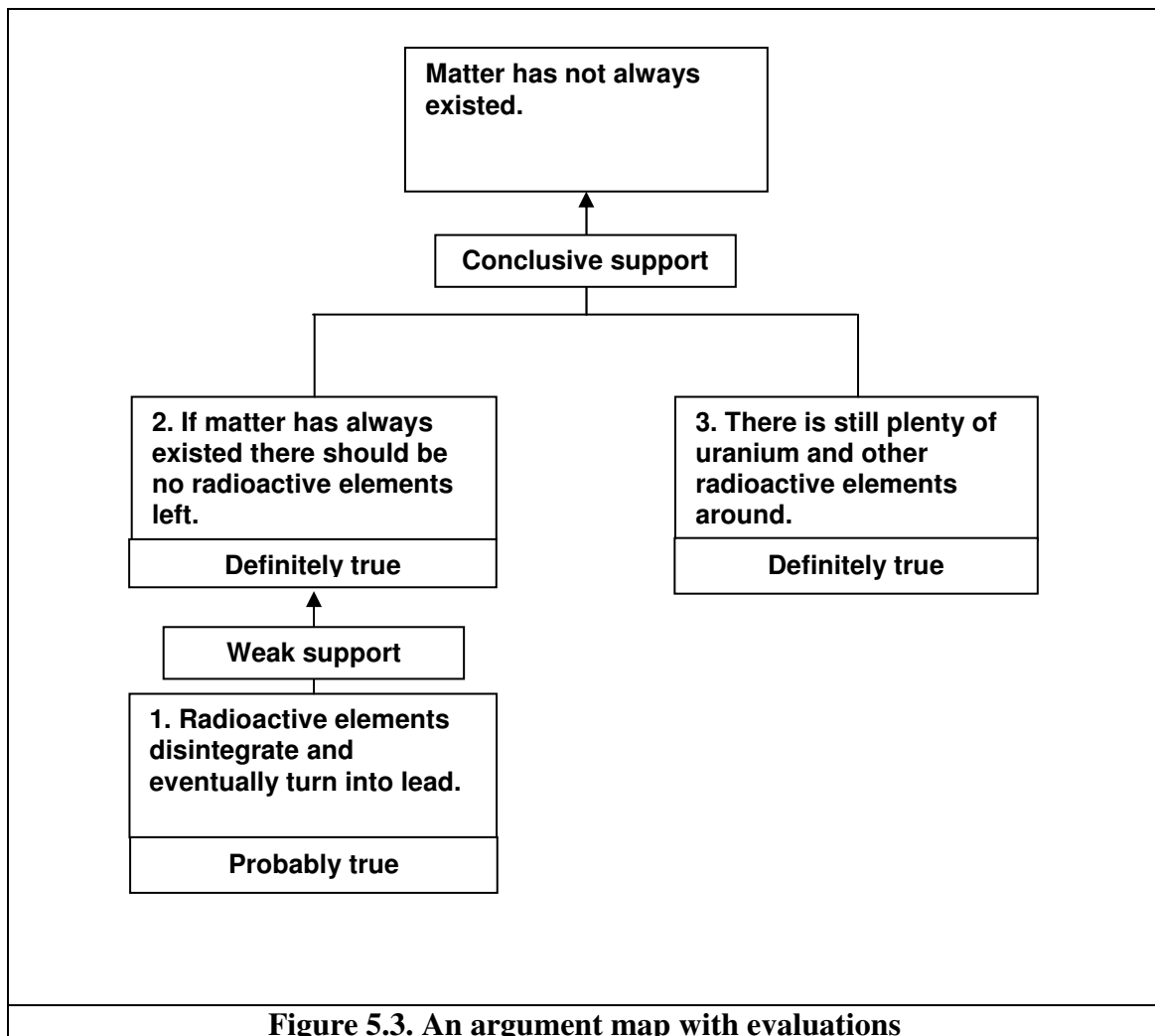


Figure 5.3. An argument map with evaluations

The system might also enforce rules relating to the internal consistency of students' evaluations. One such rule might be that a conclusion cannot be more any more acceptable than the premises used to support it. This rule has been violated in the example shown in Figure 5.3; the intermediate conclusion 2 has been evaluated as 'definitely true', even though the evidence that supports it has been evaluated as only 'probably true' (and the inference itself has been classified as 'weak'). The system could automatically pick up on mistakes like this, and provide the student with appropriate feedback.

5.5 Throwing away the scaffolding: free form maps with a model answer

Finally, we will mention one further limitation and possible extension of the system. Building automated feedback into the system in the way we have done does place some constraints on the text of arguments that can be mapped. Firstly, conclusions and premises must be represented by complete sentences that can be selected as a continuous block of text using the mouse. Secondly, the system does not give students

practice at an important skill of argument analysis; that of *paraphrasing* claims made in an argumentative text.

One way in which the system might be made more flexible than is to allow exercises where the automated feedback is turned off. Instead of selecting a block of text for the conclusion of the argument, the student clicks on the ‘Conclusion’ button and an empty box would appear in the argument map. The student then types the text of the conclusion into the empty box. Adding premises and assumptions would work in the same way. This would allow students to paraphrase conclusions and premises that appear in the text, ensuring that they are represented by complete sentences. Instead of selecting unstated assumptions from a multiple-choice list, students would be able to enter them directly onto their maps.

Feedback could then be provided by means of the traditional ‘model answer’. One way this could be incorporated into the current system is shown in Figure 5.4. Here the student is typing in the text of a premise of an argument. When they have finished constructing their map they click on the ‘Model answer’ tab, which shows a completed model map for the argument. The student can then compare their own map to the model. The system might even keep track of how many elements (conclusion, premises and assumptions) the student has added to their map and only allow the model answer to be viewed when a sufficient number of elements have been added.

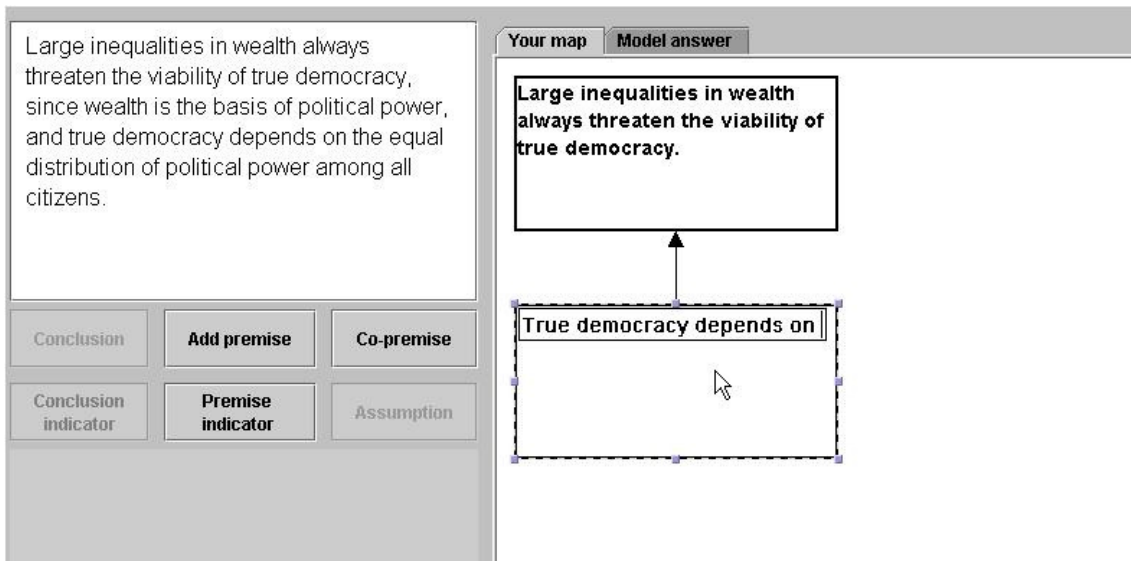


Figure 5.4. A free-form argument map under construction. Clicking on the ‘Model answer’ tab displays a completed map for the argument.

The idea is that the system could operate in two different modes. In one mode, automated feedback is turned on and the student receives instant feedback on their progress as they construct their map. In the second mode, automated feedback is turned off and students construct their maps by typing directly into the boxes, rather than selecting text. Feedback is then provided by allowing students to compare their map to a

model answer.

Automated feedback and 'free-form' argument mapping exercises could then both be incorporated into a critical thinking course. Early in the course, automated feedback is turned on, providing students with scaffolding and feedback. As the course progresses, the argument mapping exercises should become gradually more complex. Gradually, the scaffolding is removed by turning off the automated feedback and using more free-form exercises. Here again, quality practice would be further supported by allowing scaffolding to be removed as the student becomes more expert.

6. Summary

We have described a simple system for creating argument mapping exercises incorporating automated feedback. A prototype of the system was implemented and used in a single semester undergraduate critical thinking course at Monash University. Students spent approximately 30-40 minutes each week working on argument mapping exercises using the system. The results were encouraging; students showed an average improvement of 0.45 standard deviations on a standardised test of critical thinking over the course of the semester. Although limited in various ways, the system shows great potential for supporting and enhancing quality deliberate practice in an important critical thinking skill.

Acknowledgments

The research described here was funded by an Australian Research Council Linkage Project grant, (LP0346935; Effective Pedagogy for Improving Critical Thinking) and by a grant from the Australian Government Department of Education, Science and Training (DEST). The research was carried out in collaboration with DEST and the Australian Council for Education Research (ACER).

Appendix: Notes on implementation

The mechanism for providing automated feedback is fairly straightforward. The system represents the correct map of the given argument text using a tree structure, where each node in the tree represents a segment of the argument text. Each node stores just two index numbers - the point in the argument text where the segment begins and the point where it ends. The student's argument map, as it gets built up, is represented in exactly the same way.

When the student selects a segment of text using the mouse and then clicks on the 'Conclusion' button, the system simply checks to see if the start and end points of the student's selection matches the start and end points of the main conclusion (the root node) of the correct argument map. If it does, a node representing the main conclusion is added to the student's argument map. If not, the mistake is reported to the student.

From that stage on, there is always a particular node in both maps that is designated as the *current node*. The current node in the correct map is always set to be the same as the current node selected in the student's map. The student can change the current node by clicking on the appropriate box in the argument map window.

When the student selects some text and clicks on 'Add Premise' the system checks all the nodes under the current node in the correct map to see if the student's selection matches any of the nodes there. If not, the mistake is fed back to the student. If a match is found a new node representing the premise is added to the student's map. Matching is simply a matter of checking that the start and end points of the two segments coincide.

Similarly, when the student selects some text and clicks on 'Co-premise', the system simply checks all the siblings of the current node in the correct map, to see if there's a match. If there is no match, the mistake is fed back to the student. If there is a match, a new node representing the co-premise is added to the student's map.

Premise and conclusion indicators are not stored as part of the argument map tree structure, but in separate lists. Again, they are stored as a pair of index numbers into the argument text. Checking a student's selection as a premise or conclusion indicator is then simply a matter of searching through the appropriate list for a match.

This process continues until all the required nodes have been added to the student map

and all the premise and conclusion indicators have been correctly identified. Since the system knows how many nodes there are in the complete, correct map for the argument (and how many premise and conclusion indicators there are) it knows when the student's Omap has been completed.

The whole system is implemented in the Java programming language, as a so-called *Java Applet*. An applet is a java program that runs inside a web-browser. When a user loads a web-page containing an applet, the program is downloaded from the server and then runs locally, inside the user's web-browser.

Each argument map exercise then consists of a web-page incorporating an instance of the argument mapping applet. A few lines of html are all that is necessary to include an applet in a web page. The text of the argument itself is included in the html for the web page and is passed as an input parameter to the applet. The structure of the correct map of the argument is stored as a string of characters, using a simple text format. This string is also passed to the applet, via the html for the web-page. Once the applet has loaded and a representation of the correct map has been constructed from the input string, the student can begin using the system.

References

Austhink (2006). Austhink organisation website. *Rationale* argument mapping software. <http://www.austhink.com>.

Ericsson, K.A. and Charness, N. (1994). "Expert Performance". *American Psychologist*. 49, pp. 725-747.

Ennis, R. H. (1987). "A taxonomy of Critical Thinking Dispositions and Abilities". *Teaching thinking skills: theory and practice*. J. B. Baron and R. J. Sternberg. New York, Freeman: 9-26.

Facione, P. A. (1990). *The California Critical Thinking Skills Test. College Level. Technical Report #1. Experimental Validation and Content Validity*. Millbrae, California, California Academic Press. ED 327549: 21.

Facione, P. A. and N. C. Facione (1992). *The California Critical Thinking Skills Test and Manual*. Millbrae, CA, California Academic Press. Available from: <http://www.insightassessment.com/>

Fisher, A. (1988). *The logic of real arguments*. Cambridge; New York, Cambridge University Press.

Fisher, A. (2001). *Critical thinking: an introduction*. Cambridge, Cambridge University Press.

Geach, P. T. (1976). *Reason and argument*. Oxford, Blackwell.

Hatcher, D. L. (1999). "Why Critical Thinking Should be Combined with Written Composition." *Informal Logic*. 19 (2-3), pp. 171-183.

Hatcher, D. L. (2001). "Why Percy can't think: A response to Bailin." *Informal Logic*. 21 (2), pp. 171-181.

Hitchcock, David (2004) "The effectiveness of computer-assisted instruction in critical thinking", *Unpublished draft manuscript*.

<http://www.humanities.mcmaster.ca/~hitchckd/effectiveness.pdf>

Hurley, P. J. (2005). *A Concise Introduction to Logic*, Wadsworth Publishing.

Jacobs, S. S. (1995). "Technical Characteristics and Some Correlates of the California Critical Thinking Skills Test, Forms A and B." *Research in Higher Education*. 36 (1), pp. 89-108.

Kirschner, P. J., Buckingham Shum, S. J. and Carr, C. S. (eds). (2002). *Visualizing Argumentation: Software Tools for Collaborative and Educational Sense-Making*. London. Springer-Verlag.

LeBlanc, J. (1998). *Thinking clearly : a guide to critical reasoning*. New York, W.W. Norton.

LSAC (2002). *The Official LSAT PrepTest (nos. 7-38)*. Newton, PA, Law School Admission Council (LSAC).

Mackie, J. (1958). 'Notes on Informal Logic'. Unpublished lecture notes. University of Sydney.

McKeachie, W, P. Pintrich, Y. Lin and D. Smith (1986). *Teaching and learning in the college classroom: A review of the research literature*. Ann Arbor: University of Michigan, National Centre for Research to Improve Post-Secondary Teaching and Learning.

Pascarella, Ernest T. and Patrick T Terenzini (1991) *How college affects students: findings and insights from twenty years of research*. San Francisco, Oxford: Jossey-Bass Publishers.

Pascarella, Ernest T. and Patrick T. Terenzini (2005) *How college affects students volume 2: a third decade of research*. San Francisco: Jossey-Bass.

Scriven, M. (1976). *Reasoning*. New York, McGraw-Hill.

Toulmin, S. E. (1958). *The uses of argument*. Cambridge, Cambridge University Press.

van Gelder, T. (2001a) "How To Improve Critical Thinking Using Educational Technology." In *Meeting at the crossroads: Proceedings of the Annual Conference of the Australasian Society for Computers in Learning in Tertiary Education*. Australia,

Victoria: EDRS, ED. 467972.

van Gelder, T. and A. Rizzom (2001b). "Reason!Able Across the Curriculum. in 2001: Is IT an Odyssey in Learning?" *Proceedings of the 2001 Conference of ICT in Education Victoria*.

van Gelder, T. Bissett M. and Cumming G. (2004) "Cultivating Expertise in Informal Reasoning." *Canadian Journal of Experimental Psychology*. 58 (2), pp. 142-152.

van Gelder, T. 2005. "Teaching Critical Thinking: Some Lessons from Cognitive Science". *College Teaching*. 35 (1), pp. 41-6.

Author contact details

Sam Butchart, John Bigelow, Graham Oppy
School of Philosophy and Bioethics
Monash University, Clayton
Melbourne
Victoria 3800
Australia
Sam.Butchart@arts.monash.edu.au
John.Bigelow@arts.monash.edu.au
Graham.Oppy@arts.monash.edu.au

Kevin Korb
School of Information Technology
Monash University, Clayton
Melbourne
Victoria 3800
Australia
Kevin.Korb@infotech.monash.edu.au

Ian Gold
Department of Philosophy
McGill University
Leacock Building
855 Sherbrooke St. W.
Montreal, Quebec
H3A 2T7
Canada
Ian.Gold@McGill.ca