British Journal of Educational Technology *Vol 34 No 5 2003* 625–638

Software that assists learning within a complex abstract domain: the use of constraint and consequentiality as learning mechanisms

James Aczel, Pat Fung, Richard Bornat, Martin Oliver, Tim O'Shea and Bernard Sufrin

James Aczel is a Lecturer at the Institute of Educational Technology (IET) at the Open University. His research interests include mathematics education, focusing on students learning algebra and logic, and Popperian psychology. Address for correspondence: Institute of Educational Technology, The Open University, Walton Hall, Milton Keynes, England, MK7 6AA, Tel: +44 1908 652953, email: j.c.aczel@open.ac.uk. Pat Fung was director of research in IET, and has now retired. Richard Bornat's research interests included interface design and research into logic; he has also left his academic post. Martin Oliver researches the use of computers in Higher Education, focusing particularly on curriculum design, evaluation and the impact of technology on staff practice. Address for correspondence: Education and Professional Development, UCL, 1-19 Torrington Place, London, England, WC1E 6BT, Tel: +44 207 679 1905, email: martin.oliver@ucl.ac.uk. Professor Tim O'Shea has published widely within the field of Artificial Intelligence in Education, and he is past Chair of AISB, the Society for the Study of Artificial Intelligence and the Simulation of Behaviour and past President of the Psychology section of the British Association for the Advancement of Science. He has recently taken up the post of Principal of Edinburgh University. Address for correspondence: The University of Edinburgh, Old College, South Bridge, Edinburgh, Scotland, EH8 9YL, Tel: +44 131 650 2150, email: principal@ed.ac.uk. Bernard Sufrin is a Lecturer, Fellow and Tutor in Computation at Worcester College , Oxford University. His research interests include computer-based support for refinement and proof; formal methods, abstraction, and patterns in system design; programming language design and implementation; and intentional programming. Address for correspondence: Oxford University Computing Laboratory, Wolfson Building, Parks Road, Oxford, OX1 3QD, England, Tel: +44 1865 273828, email: Bernard.Sufrin@comlab.ox.ac.uk.

Abstract

This paper describes a research project into undergraduates' use of a software tool to learn symbolic logic—a complex abstract domain that has been shown to be intimidating for students. The software allows the students to manipulate proofs in certain ways and then calculates the consequences of their actions. A research method has been developed that allowed students' use of this tool to be modelled, and this model was then used to identify, refine and create visual cues that provide support for students' reasoning. The focus of this paper is the role of the software as an artefact to aid students' visualisation of reasoning processes rather than the logic itself. The main mechanisms by which this visualisation is supported are the imposition of *constraints* on the actions available and the demonstration to students of the *consequences* of their actions. The study shows that the software encouraged experimentation with different routes to a proof, and constituted a challenge to fixated reasoning.

Published by Blackwell Publishing Ltd., 9600 Garsington Road, Oxford OX4 2DQ, UK and 350 Main Street, Malden, MA 02148, USA.

Introduction

Many students find formal reasoning to be extremely abstract (Cheng *et al.*, 1986) and even "frightening" (Fung *et al.*, 1993, 1994). Such difficulties seem to be particularly acute for students who lack prior experience with formal languages such as algebra (Fung and O'Shea, 1992).

In response to these difficulties, a number of computer-based tools have been developed to support students studying logic (eg, Goldson *et al.*, 1993; Fung *et al.*, 1996). Many of these support learning by automating elements of proof construction, allowing proofs to be saved and reviewed, and using feedback to prompt reflection. In addition, several of these tools incorporate graphical representations in order to help students interpret the formal language.

Research has demonstrated that using visual, concrete examples as illustrations to complement formal notation can be an effective strategy for supporting learning (eg, van der Pal, 1995). However, it is possible to distinguish (Oliver, 1997) the employment of representations as entities to be reasoned about (that is: "content representations") from the employment of representations as supports for the visualisation of reasoning processes (that is: "process representations"). It has been argued that software tools that incorporate both content and process representations will be more effective at supporting learners.

However, to date, there has been little research into the means by which useful process representations can be developed. This paper describes a research project into undergraduates' use of a formal reasoning tool to learn symbolic logic. This software tool—*Jape*—allows the students to manipulate proofs in certain ways and then calculates the consequences of their actions. A research method has been developed that allowed students' use of this tool to be modelled, and this model was then used to identify, refine and create visual cues that provide support for students' reasoning.

The context

The subjects of this research were 170 computer science undergraduates taking an introductory course in propositional and predicate logic. The students were expected to translate natural language statements into a given formal representation, to evaluate formal representations semantically, and to prove conjectures using formal rules.

A program called *Tarski's World* (Barwise and Etchemendy, 1992) was used to help students with the first two of these tasks—translation and evaluation. Previous research (Fung and O'Shea, 1992; Fung *et al*, 1996) has shown that this software effectively assists students by providing concrete, visual referents for symbols.

However, the essence of the third task—formal proof of conjectures—is that its objects and operations are purely syntactic; supporting proof construction by means of concrete referents would defeat the point of the activity. This domain of knowledge is therefore quintessentially *abstract*. It is also *complex* in the sense that each proof consists of

an unpredictable number of lines; each line has to be justified by one of a set of basic proof rules; each rule has a different structure; and each structure might apply to the partially-completed proof script in several ways. Given this abstractness and complexity, it is perhaps unsurprising then that the notion of formal proof is notoriously poorly appreciated by students at high school or university.

ItL Jape

A program called *Jape* (Bornat and Sufrin, 1999) is provided for the students to support learning. It allows them to manipulate proofs using a mouse; when users apply a rule to a line of the proof, the software calculates the consequences. Jape can be configured by an educator to use a variety of logics, to present proofs in different ways, and to allow various user actions. This research looks at a particular implementation, called *ItL Jape*, which has been configured with the "natural deduction" style of logic and pre-loaded with around 70 conjectures to be proven. Van Ditmarsch (1998) concluded that ItL Jape is the most visually "appealing" of the five proof assistants he examined.

In the course described above, *Jape* was introduced to help students with the third task (formal proof of conjectures). The software was made available on the local computer network and for downloading, and lab sessions with teaching assistants were scheduled as part of the course. Within these, students were set proofs to complete and were able to use the software to aid them in this process (see, eg, Figure 1). The conjectures that were set as coursework were pre-loaded within the software; students who completed a proof were then able to save this and return to it at a later date. Students could also enter their own conjectures if they so wished.

By focusing on a specific implementation of the software, this research is unable to evaluate Jape's flexibility in the logics it can handle and the interfaces it can present. But it was felt that it was only possible to understand the strengths and pitfalls of such tools in supporting visualisation by examining students' specific interactions.

This paper does not consider the details of the logic—see Aczel (2000) for fuller results. However, it may be instructive to observe how what the student sees on the screen changes during the course of a typical proof (Figure 1). In particular, notice how the ellipsis ("...") directs attention to missing steps.

Research method

Three overlapping studies were conducted—the "Measurement Study", the "Observational Study" and the "Reflection Study".

In the Measurement Study, data was gathered on students' backgrounds, their usage of Jape, and their success in the course. The use of control groups was not possible—on ethical and pragmatic grounds—and so a conventional experimental design to compare interventions proved impractical. Instead, comparisons were drawn *within* the context of the course (rather than between this intervention and some alternative). For example, data were gathered that enabled an exploration of whether different student

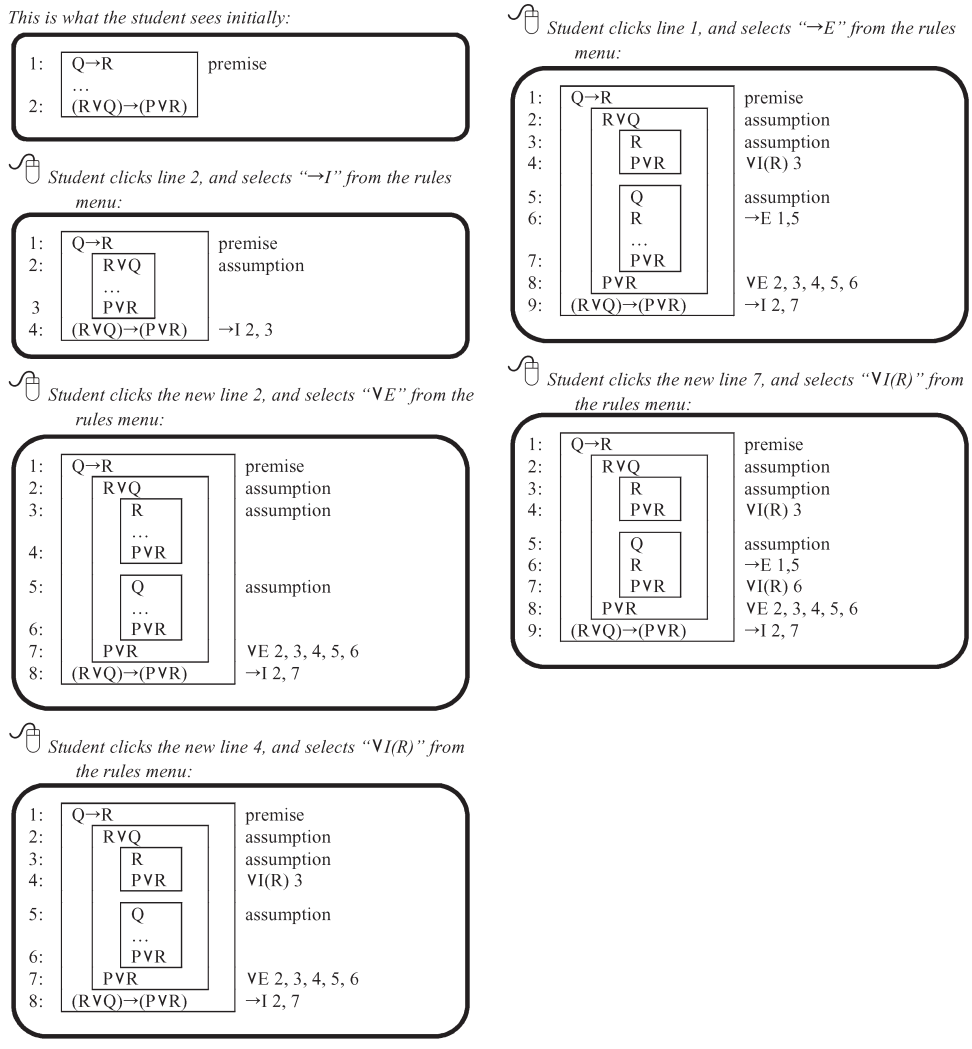

Figure 1: Example Proof

groups were more likely to use Jape, and of whether students who used Jape more did better in the course. The main data collection methods were written tests, questionnaires, and automatic logs of program usage. Full details of the instruments, samples, and assumptions are in Aczel (2000).

In the Observational Study, the work of students was observed over the course, following a largely naturalistic approach (Guba and Lincoln, 1981) in order to understand how ItL Jape fitted into the learning context. The bulk of this observation took place in

eleven weekly small-group hour-long workshops, although the teaching also included twice-weekly hour-long lectures, comprehensive course notes, and tutorial classes. The workshops provided an opportunity to observe students actively engaged with the subject matter, and to talk informally to them about their understandings and difficulties. Jape was used during three workshops, the first of which began with a half-hour introduction to the program. Usage varied from nothing (16% of the sample) to more than 2 hours (19%). Four volunteer students were videotaped as they used Jape during each of the workshops.

In the Reflection Study, ten students—six individuals and two pairs—were videotaped using the program in task-based interviews. The main aim of the study was to test the findings of the Observational Study by presenting to students situations similar to those in which important incidents had occurred, seeing if the incidents were replicated, and, if so, obtaining students' interpretations of the incident. By careful questioning, it was hoped to discover the visual cues to which students were attending in order to arrive at their decisions. Such interventions by the researcher were not part of the Observational Study because they would have disturbed the natural flow of the students' work and directed attention to aspects of the situation that might not otherwise be noticed. By tackling proofs on paper first and then using the software, interface difficulties were highlighted when paper attempts had been successful; and when paper attempts had been invalid or had stalled, it was possible to identify the features of the software that enabled progress in visualising the reasoning process.

Results

The relationships between student background, Jape usage and test scores

A quantitative analysis of tests, surveys and logfiles suggests that students' backgrounds appeared to have little effect either on how much the program was used, or on how much progress was made with the conjectures in the program.

However, on average, the more a student used Jape the higher was his or her score in course outcome measures. Moreover, males, programming novices, and the financially-motivated tended to get lower test scores, *unless* they used Jape for more than an hour.

Fixated reasoning is challenged

Each task in Jape followed a similar format: you are given some premises and a conclusion; now find the proof that the conclusion follows from the premises.

One might expect students to start from the given premises and write down a successive series of lines (justified by the appropriate rules) that ends up at the given conclusion. Many students did this. This is referred to as "reasoning forwards", in that it involved reasoning from the premises *forwards* to the conclusion.

However, some students realised in addition that, very often, they could also work out a line or two that must precede the conclusion. We call this strategy "reasoning backwards", in that it involved reasoning from the conclusion *back* to the premises. On paper,

there was clear evidence of a distinction between behaviour that was *fixated on reasoning forwards* and behaviour that was *flexible about reasoning forwards or backwards*. So, for example, a forward-fixated reasoner would want to *make* assumptions when a flexible reasoner would *calculate* assumptions. In Jape, however, although forward-fixated reasoners felt some frustrations about not being able to apply certain rules in the way they could on paper (eg, the rule "$\wedge$I"), many quickly discovered the usefulness of applying a certain rule backwards (eg, the rule "$\rightarrow$I"), and this skill often transferred back to paper.

Interface difficulties

Difficulties with the interface mostly arose out of one situation in particular: if students failed to select a line before choosing a rule, the software guessed the line to which the rule was to be applied. There were two common difficulties with such incompletely-specified steps. Firstly, the direction of application may have been opposite to the one intended. Secondly, placeholder characters ("unknowns") were often produced to represent missing information. This feature is useful to experts; but the novices involved in this study did not know what these characters were for. In both scenarios, students typically misinterpreted unexpected output as due to a faulty choice of rule rather than as due to a failure to click a line; and they would then typically be misled into trying an incorrect rule.

Such were the difficulties created by this situation that some students even went so far as to say that they would prefer to type in their proofs line-by-line for checking, rather than have Jape generate the lines in response to mouse clicks.

Learning from exploration is encouraged

On paper, errors in proofs were frequently undetected by students; and explorations of different routes to a proof were rarely deep. Lab assistants, lecture notes, fellow students, and tutors were relied upon both to validate the correctness of proofs and to provide hints as to how to proceed. The use of the lecture notes as a key authority was problematic for conjectures that were superficially similar to conjectures in the notes or that did not appear.

Meanwhile, in Jape, students trusted the program not to make illegal steps; and students who overcame forward-fixated reasoning successfully used the fast feedback and the undo facility to explore how to proceed. This exploratory behaviour often transferred back to paper. On the other hand, there were also examples of students attempting to over-generalise from one rule to another.

On paper, many students appeared to write down a line and then try to think of the rule that justified it. The empty spaces where lines were needed were therefore a focus for attention, and when these spaces became small (because the number of intermediate lines was unpredictable), students' work tended to become either untidy or invalid.

In Jape, students seemed to focus on a particular line on the screen and then try to find a rule to apply to that line. The choice of line upon which to focus generally seemed to be either the most complex line or the lines either side of the ellipsis (the "…" symbol). The ellipsis acted as an important visual cue of "work to be done" and its disappearance provided a satisfying feeling of completion when the proof was finished. There is little evidence of attention to the justifications in deciding what rules to apply and where.

On paper, proving was slowed down by the need to draw boxes (as in figure 1), and it also seemed almost as if the untidiness created by box-drawing tended to diminish students' appreciation of their work. In addition, incorrectly positioned boxes often led students astray, even when their basic plan for constructing the proof was initially sound.

In Jape, box-drawing was automated (and hence there were no positioning errors) and many students said that they valued how much faster this made the process of proving. However, on returning to paper, students were not noticeably more skilful in positioning the boxes.

Interviews and observations suggest that students tended to tackle a far greater number of conjectures in Jape than on paper. Many students put this down to the point-and-click interface and the automation of proof-drawing. However, there is clear evidence of some students struggling to recall how to reproduce proofs they had successfully constructed the week before.

The fact that Jape calculates the effect of actions meant that students often had to deal with unexpected changes in the proof script, a phenomenon which does not arise on paper. There were several consequences. Firstly, steps that created sudden enlargements in the size of the proof, that bifurcated the proof, or that introduced unfamiliar signs were often misdiagnosed as erroneous. Secondly, students frequently failed to notice when unhelpful lines were introduced—checking for the impossibility of proving a conclusion from preceding lines (by informally interpreting the logical connectors) was not a common strategy, in spite of much prior instruction in the semantics of formal logic. Thirdly, the software carried out rule applications so fast that it was sometimes difficult to grasp what had just happened; however, the undo and redo facilities were then often used to replay the step.

Changed perceptions

Analysis of the discourse used in interviews and proof episodes suggests that the typical paper-based perception of a proof as a written, linear sequence of logical formulae contrasts with an alternative perception in Jape that a proof is a set of simplifications of the conclusion and premises. Rules feature as technical warrants for lines during paper proofs, but they are "applied" in Jape to generate lines and justifications automatically.

Both on paper and in Jape, little attention was paid to the structure of conjectures, or to the similarities between proofs. Indeed, sometimes students had more difficulty the second time they proved a proof segment than the first time. Interestingly, it was noticeable that at various points in some complex proofs, students *suddenly* became confident (not always rightly) about the success or otherwise of the particular approach taken. Both on paper and in Jape, students saw some rules as harder than other rules. Implication and Conjunction tended to be seen as the easiest topics; Disjunction was next; Negation and Quantifiers were held in equal dread. This order matches the order in which the rules were introduced in the lectures, the order in which the rules were practised on paper, the order in which the conjectures were presented in ItL Jape, and (roughly) the order in which most students believed rules should be applied. It was also observed that conjectures without premises seem to be viewed as potentially harder a priori.

Analysis: modelling the learning mechanisms

This section outlines a theoretical model that is used in the discussion (below) to illuminate these results.

In order to explain the reasons for students' general success in learning from Jape, and for their failure to learn from feedback in some situations, a model of students' knowledge of the domain has been developed. See Aczel (2000) for details of the development and testing of the model. This model posits a set of "proof strategies". Such strategies vary in how they are combined and in how they are articulated. *Rule-specific strategies* enable students to decide what rule should be applied in given situations, to implement that rule, and to predict the effects of the rule. For example:

«If the principal operator in the conclusion is →, break up that conclusion by using →I backwards—it creates an assumption box that often allows forward reasoning»

«Use ∀I backwards before ∀E forwards».

In contrast to rule-specific strategies, *global strategies* help students to plan how they will attempt a proof, and to debug that plan. For example:

«Look for the most complex line in order to decide on a focus for attention»

«When reasoning backwards, check if the lines produced are provable from the premises».

In addition, users have been categorised by their prior knowledge of the rules. Group 1 users know the name of the rule they want to apply but are not necessarily aware of the precise effects of the rule; Group 2 users know how they want the transformed proof to look, but are less sure about the name of the rule that achieves this transformation; Group 3 users have a partial understanding of the rules and are trying to understand the rules from the output of the program; and Group 4 users have never met the rules before and so are not target Jape users.

These proof strategies and groups of users help to explain much observed behaviour in this domain (Aczel, 2000). However, a model of reasoning processes is not enough in

Figure 2: Paper proof—Group 1 students

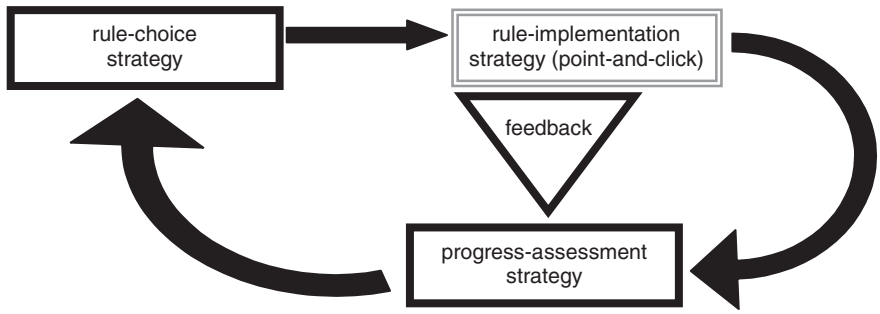

Figure 3: Jape proof—Group 1 students

Figure 4: Paper proof—Group 2 students

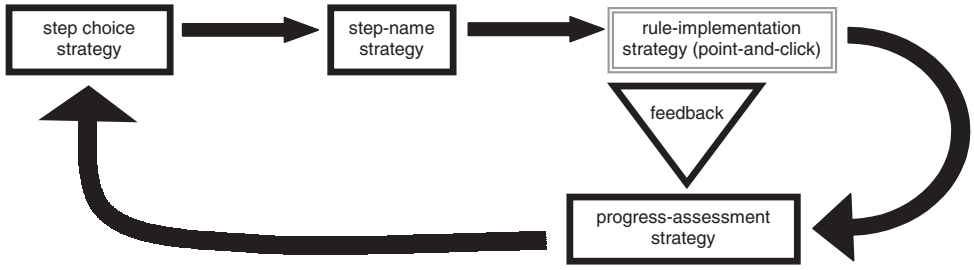

Figure 5: Jape proof—Group 2 students

itself to understand how representations are supporting the *learning* of these processes—it is also necessary to model the mechanisms by which embryonic processes are refined.

Figures 2 to 7 summarise some of the detailed analysis (Aczel, 2000). The important thing to note is that the learning mechanisms are transformed by the facts of the software making it easier to implement a rule and providing feedback that can be used in

Figure 6: Paper proof—Group 3 students

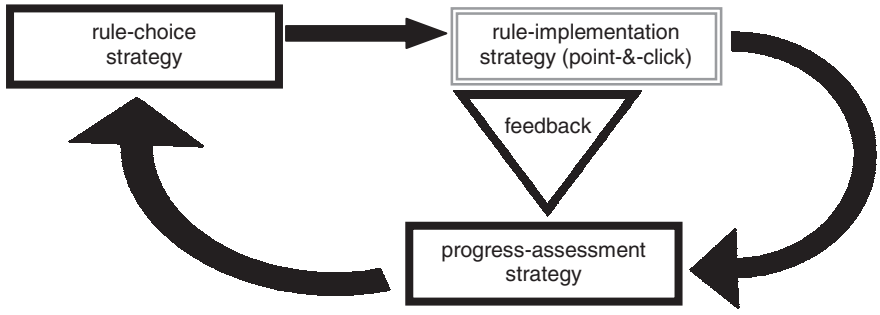

Figure 7: Jape proof—Group 3 students

an assessment of whether the rule constitutes a movement in the direction of proof completion.

Discussion

Explaining the challenge to fixated reasoning

With this model of prior knowledge and learning mechanisms, many of the empirical results can be explained. We consider here just two results: the challenge to forward-fixated reasoning, and the encouragement of exploratory learning.

Forward-fixated reasoning was challenged because the software *constrained* the strategies available. For example, the forward-reasoning strategy «make an assumption» was not possible, but the backward-reasoning strategy «apply →I backwards» was. Nevertheless, just being *forced* to carry out a backwards step rather than a forwards step does not necessarily demonstrate its superiority (and hence encourage a transfer of such a strategy to paper), as evidenced by the frustration of Group 1 and Group 2 students about not being able to find in Jape certain forward-reasoning rules that would produce output they knew was legitimate.

The superiority of the strategy was demonstrated for some students when they saw that certain anticipated structures were generated automatically. But those Group 1 and Group 3 students who were not anticipating these structures were unlikely to appreciate the value of the backward-reasoning strategy. Moreover, Group 2 students who failed to realize that the only way to generate structure was by applying rules struggled to make much progress.

Explaining the encouragement of learning from exploration
The poor progress-assessment strategies exhibited by many students suggest that they are Group 3 students. Consequently, features of the software that contributed to the systematic testing of embryonic rule-choice and progress-assessment strategies would be appreciated—such as the similarity of the display to the written version, the familiar point-and-click interface, the list of rules, the accurate calculation, the fast feedback, the automatic box-drawing, the undo facility, the gradual increase in the complexity of proofs, the indication of progress via the list of completed conjectures, and the ellipsis.

Features of the software that were *irrelevant* to this testing would be ignored—such as justifications, structural aspects of proofs, efficiency and semantic considerations. Features of the software that *inhibited* this testing would be problematic—such as troublesome inputs (misclicking, ambiguous labels, an uncomfortable menu structure, parameters, special selection modes), complex outputs (dramatic changes in proof size, bifurcations, variables), and unfamiliar outputs (placeholder unknowns, side conditions, opaque dialogue messages). These features that inhibit strategy testing also go some way to explain the relative perceived difficulty of the topics.

Students could tackle many more proofs in Jape than on paper not just because of the speed with which proofs could be constructed using the mouse, and because the messy task of drawing the proof (drawing boxes, leaving spaces) was handled automatically, but also because illegal moves were challenged instantly—which eliminated the need for repeated checking of one's work—and because the ellipsis constituted a visual cue to focus attention.

Group 3 students have to develop a large number of rule-specific strategies in a relatively short time. It is therefore unsurprising that they would have difficulty in remembering them a week later. Moreover, they started work with the conjectures that they had found difficult even when the strategies were freshly formed—this would not aid recall. Students appeared to attempt a variety of means to cope with the memory demands of these strategies; for example, reviewing already completed proofs, tackling proofs from the previous session to recapitulate the strategies; and replaying steps that had a dramatic effect on the proof. The most prominently unhelpful way to cope with memory demands was to overgeneralise strategies between rules.

What interface changes might promote better process visualisation?
Given the empirical results and the theoretical model of learning mechanisms, it is possible to analyse changes to the software that might enhance learning. Many of these have already been incorporated in the next version of ItL Jape. In order to help dislodge forward-fixated reasoning, to create greater awareness of the structure of the rules, and to aid recall—particularly among Group 2 and Group 3 students—the rules menus could include mnemonics (text, icons, sounds, animations) that suggest the structure of the rule without bypassing its name. Moreover, some rule applications surprised students by the sudden contraction or expansion of the proof; perhaps

there could be an option to animate changes slowly, or to show changes in a different colour.

The automatic drawing of boxes and the automatic insertion of justifications left the students free to focus on debugging their rule-choice and progress-assessment strategies. But if, once these strategies have been developed, the teacher then wished students to develop rule-implementation and justification strategies, there could be options to turn off automatic boxes and justifications.

A student could spend a great deal of wasted time on a proof script in which he or she had previously generated a line that could not be proven. Although it is not possible in principle to detect all such blind-alley situations, the software could check for blind-alleys that a tutor would notice by informally interpreting the logical connectors, and then alert the user to it if he or she failed to notice it after a certain length of time, after a certain number of steps, or after the proof had reached a certain size. If there were an option to turn this off in time, perhaps students would then begin to check for themselves.

The students did not find the dialogue messages comprehensible, which was unsurprising as the messages were intended to provide feedback to logic designers rather than to logic learners. Consequently, students tended to ignore the text of the dialogue box and treat the appearance of the box merely as an indication that they had attempted an illegal step. Those messages that *were* informative were therefore not read properly. All the messages have now been tailored to specific common problem situations, indicating what action the user attempted (thus helping the detection of misclicks) and suggesting possible resolutions. The next version of ItL Jape also allows students to obtain assistance on terminology that features in the dialogue messages, such as "hypothesis", "conclusion" and "unproved conclusion". Another difficulty with the dialogue box was that students were unable to continue working on the proof until the dialogues were dismissed, which distracted attention away from the body of the proof and meant that the message was not visible while students attempted any suggested actions. A permanently-visible advice panel might help here, and could also let students peruse a "history" of attempts, thus minimising repetitive failed attempts.

This research found that letting novices specify steps incompletely could actually undermine their tentative theories and strategies. In the light of these findings, the next version of ItL Jape has implemented "training wheels"—a feature that requires students to specify the formula to be simplified and to choose the direction of application, and puts on the menus only those steps that novices are likely to use (thus removing many of the opportunities for incompletely-specified steps).

The misclicking of items on the rules menu, while not frequent, tended to cause confusion when students were unaware that they had misclicked; and if students were not sure if they had misclicked, the only way to tell was to undo and reapply steps, and errors in reapplication compounded the confusion. Improving the readability and

structure of the rule menu may help reduce misclicking; while detection of misclicks might be aided by showing the action just taken next to the "undo" menu item, in dialogue messages or in a history.

One commonly observed phenomenon was the rapid yet inefficient traversal of the rules menu by the mouse pointer. An analysis in terms of strategies suggests that if this indicates difficulty in finding the rules then it might be helpful for some user groups to have the rules menu divided into forward and backward rules rather than into introduction and elimination rules.

Jape could offer a feature that enabled students to simplify formulae by double-clicking rather than selecting a rule. An analysis of the effects of this feature in term of proof strategies suggests that there are advantages for Group 1 students who are already competent with the rules. However, although double-clicking simplification may help forward-fixated Group 2 and Group 3 students to reason more flexibly, they might not gain competence that transfers to paper.

Group 2 students would appear to have the least productive experience of Jape. If these students could be identified accurately, one way of making the program more productive for them might be to turn off at least some of the forward-reasoning restrictions that inhibit sub-optimal steps but that consequently jar with these users. However, if this were done, the program would need to encourage flexible reasoning by other mechanisms, such as through context-related hints.

Conclusion

Although students find formal reasoning to be difficult, previous research has suggested that visualisation tools could help with this abstract domain. This study has supported the findings of such research.

The evaluation of Jape has improved our understanding of how students really construct proofs. The main mechanisms by which students' visualisation of formal reasoning processes is supported are the imposition of *constraints* on the actions available and the demonstration to students of the *consequences* of their actions. The software also allowed students to consider many more examples than would be possible on paper, it encouraged experimentation with different routes to a proof, it challenged fixated reasoning, and it lessened reliance on other people as authorities for correctness.

Moreover, a variety of visual cues have been identified to support learning, in particular the ellipsis, the rules menu, and the graphical display of successive versions of the proof script. However, the success of these cues depends crucially on the prior knowledge of students. Not all visual cues are equally effective in supporting the reasoning process; in fact, some are detrimental to it. The software has been redesigned as a result of this research, particularly the menu structure, the annotation of rules, the treatment of missing information and the dialogue messages. Assessing whether

these changes enhance students' experience would assist in the corroboration of this analysis.

In this method, a better understanding of the process that the students are attempting to visualise has been crucial to developing a better educational technology. We have seen here that modelling learning mechanisms in detail using *strategies* pays rich rewards in terms of the ability to explain and predict interactions with educational software. Such detailed analysis is vital for the identification of learning-supportive features and for the evolutionary development of educational software.

References

Aczel J C (2000) *The evaluation of a computer program for learning logic: the role of students' formal reasoning strategies in visualising proofs* CALRG report, The Open University.

Barwise J and Etchemendy J (1992) *The language of first-order logic* Third Edition, Cambridge University Press.

Bornat R and Sufrin B (1999) Animating formal proof at the surface: the Jape proof calculator *The Computer Journal* **42**, 3, 177–192.

Cheng P, Holyoak K, Nisbett R and Oliver L (1986) Pragmatic versus syntactic approaches to training deductive reasoning *Cognitive Psychology* **18**, 293–328.

Fung P and O'Shea T (1992) *Using software tools to learn formal reasoning: a first assessment* CITE Report No 168, The Open University, UK.

Fung P, O'Shea T, Goldson D, Reeves S and Bornat R (1993) Computer science students' perceptions of learning formal reasoning methods *International Journal of Mathematical Education in Science and Technology* **24**, 5, 749–760.

—— (1994) Why computer science students find formal reasoning frightening *Journal of Computer Assisted Learning* **10**, 240–250.

—— (1996) Computer tools to teach formal reasoning *Computers in Education* **27**, 1, 59–69.

Goldson D, Reeves S and Bornat R (1993) A review of several programs for the teaching of logic *The Computer Journal* **36**, 4, 373–386.

Guba E and Lincoln Y (1981) *Effective evaluation: improving the usefulness of evaluation results through responsive and naturalistic approaches* Jossey-Bass, London.

Oliver M (1997) *Visualisation and manipulation tools for modal logic* unpublished PhD thesis, Open University.

van der Pal J (1995) *The balance of situated action and formal instruction for learning conditional reasoning* unpublished PhD thesis, University of Twente.

van Ditmarsch H (1998) User interfaces in natural deduction programs *in* Backhouse R C (ed) Proceedings of the 4th International workshop on user interface design for theorem proving systems, Eindhoven University of Technology.