Using computers to learn logic: undergraduates' experiences

James Aczel, P. Fung, Richard Bornat, Martin Oliver, Tim O'Shea, Bernard Sufrin

To cite this version:

James Aczel, P. Fung, Richard Bornat, Martin Oliver, Tim O'Shea, et al.. Using computers to learn logic: undergraduates' experiences. Cumming, G., Okamoto, T. & Gomez, L. Advanced Research in Computers and Communications in Education, Amsterdam, IOS Press, pp.875-882, 1999. hal-00190200

HAL Id: hal-00190200

https://telearn.archives-ouvertes.fr/hal-00190200

Submitted on 23 Nov 2007

Using computers to learn logic: undergraduates' experiences

Aczel, J. C.[1], Fung, P.[1], Bornat, R.[2], Oliver, M.[3], O'Shea, T[4], & Sufrin, B.[5]

[1] *Institute of Educational Technology, The Open University, UK*
[2] *Department of Computer Science, Queen Mary & Westfield College, London*
[3] *Learning and Teaching Innovation and Development, University of North London*
[4] *Birkbeck College, London*
[5] *Oxford Computing Laboratory, University of Oxford*
Address for correspondence: James Aczel, Institute of Educational Technology, The Open University, Milton Keynes, MK7 6AA, UK, Tel: + 44 1908 652953, Email: j.c.aczel@open.ac.uk

Learning formal logic can be difficult for many students. This paper describes some ongoing research into a computer program designed to help computer science undergraduates learn the natural deduction style of formal reasoning. Data collection methods included observation and videotaping of workshops, interviews, written tests, surveys, and logging of program usage.

The paper focuses on students' experiences using the program to assist proof construction. It was found that videotaping students provided interesting insights into the effectiveness of the program as a learning tool. In particular, it is noted that students made use of a number of *rule-specific* and *global* strategies to help them construct proofs; and that these were, for the most part, developed by students themselves. It is suggested that for those students adopting a reflective approach, the program was more effective than pencil-and-paper in encouraging the progressive refinement of proof strategies, not just because more proofs could be considered but also because inadequate proof attempts and incorrect rule applications were immediately challenged. The findings from the workshop videotapes inform the design of the next phase of the research.

Keywords: **Interactive Learning Environments**; **Teaching & Learning Strategies**; **Logic Software**

1 Introduction

There is evidence that many students can find formal reasoning difficult [8][9]. Yet it is also clear that there are programs - such as Tarski's World [1] - that are considered by students to be useful and enjoyable for learning the syntax and semantics of first order logic [7][10]. Nevertheless, the notion of *proof* is notoriously not well appreciated by students at high school or university.

Can students be helped by software tools to understand the nature of formal proof? *How* exactly can students use a software tool to learn to construct proofs? What aspects of the interface and functionality might be vital for non-superficial understanding?

The program Jape [2] allows interactive, step-by-step construction of proofs for a variety of logics. It allows a teacher some degree of control over the rules that can be used, the display of the proof on-screen, and the effects of mouse clicks. By researching students' experiences with Jape and by trying to discern the cognitive processes at work when students work on proofs, it is hoped to increase understanding of the effectiveness of such tools in supporting the learning of formal reasoning for software development. The implementation of Jape used in this research shares some similarities with MacLogic [4] and the Carnegie Mellon University Proof Tutor [14], but the style of the graphical interface is innovative in its “quietness” [3].

This paper outlines ongoing research into Jape’s effectiveness in supporting an undergraduate course in first-order logic using natural deduction. The paper focuses on a particular phase of the research, in which students were videotaped as they used the program to assist them in constructing proofs. After an outline of the specific implementation of the program used and the approach to data collection adopted, some results of this phase of the research are described. The insights gained from the videotaping are being used to inform the next phase, data from which might help to illuminate a domain of great interest to educators in computer science, mathematics and philosophy.

2 The Specific Implementation of Jape Used in the Research

Jape takes a description of a particular logic as a system of inference rules. The program has been applied to several logics, including predicate calculus, Hindley-Milner type assignment, axiomatic set theory and a functional programming logic. The proof tree in any logic can be directly manipulated simply by clicking a formula with the mouse. Jape has a tactic language in which actions may be bound to mouse clicks, menu items and keystrokes.

Among recent attempts to evaluate the educational potential of logic software, Kadoda [13] compared features available across a wide range of theorem-provers (some of which might be used in educational settings), using a standard questionnaire given to program users and developers, and based on the vocabulary of “cognitive dimensions” [11]. Van Ditmarsch [5], meanwhile, compared five natural deduction proof assistants using issues such as how proofs are displayed, bias towards either forward or backward reasoning, and the availability of help.

However, in order to address the question of how precisely Jape might assist learning, it was clearly necessary for us to examine students’ experiences with logic at a level of detail that enables conclusions to be drawn from specific interactions with the software. A particular implementation of Jape was therefore investigated. This implementation (“ItL Jape”) was used by a cohort of first-year computer science undergraduates as they undertook an introductory course in propositional and predicate logic. The software was pre-loaded with the “natural deduction” style of reasoning; it was configured so as to present students with a sequence of conjectures to prove; and it was also configured to act more like a logic calculator than a theorem-prover, in that in order to construct each proof, the student had to indicate, step-by-step, which rules they wished to apply to which lines.

So, in proving the conjecture $Q\to R \vdash (R\lor Q)\to(P\lor R)$, ItL Jape would initially display:

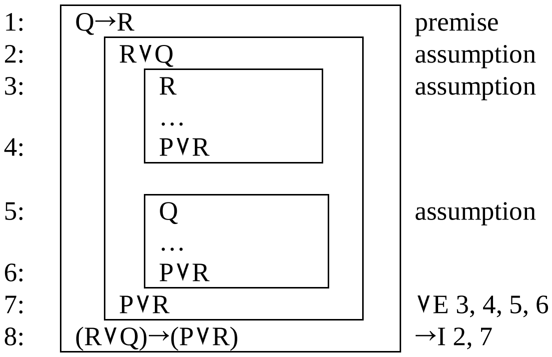
1:	$Q\to R$	premise
	…	
2:	$(R\lor Q)\to(P\lor R)$	

Clicking line 2 and selecting “→I” from the rules menu changes the display:

1:	$Q\to R$	premise
2:	$R\lor Q$	assumption
	…	
3	$P\lor R$	
4:	$(R\lor Q)\to(P\lor R)$	→I 2, 3

In effect, the “implication-introduction rule” has been applied.

A student might proceed from here by using ∨I(L) (“or-introduction left”) on line 3. This would lead ultimately to a dead-end. Applying ∨E (or-elimination) to line 2 is more useful:



1:	Q→R	premise
2:	R∨Q	assumption
3:	R	assumption
	…	
4:	P∨R	
5:	Q	assumption
	…	
6:	P∨R	
7:	P∨R	∨E 3, 4, 5, 6
8:	(R∨Q)→(P∨R)	→I 2, 7

And so on, until the proof is complete (the ellipsis symbol “…” would disappear).

ItL Jape is the implementation considered by van Ditmarsch [5] to be the most visually “appealing” of the five proof assistants he scrutinised, although he notes that “There is no proof help, apart from ‘debugging’ help: helpful warning messages when wrongly applying rules.” and “It is not possible to submit entire proofs. It is not even possible to make an incorrect proof step, as rule execution is automatic given a formula and a rule. This makes Jape less fit for teaching natural deduction to ‘absolute beginners’.”.

The course lecturer decided not to provide ItL Jape with direct manipulation rules (so, for example, double-clicking on a formula would apply the most “obvious” rule) because it was suspected that novice logicians would learn more about natural deduction if they had to choose the rule for themselves. But Jape could obviously be configured with different users in mind - more experienced logicians might want different aspects automated. Note also that Fitch boxes [6] were chosen for displaying proofs, rather than proof trees.

Clearly, then, by focusing on a specific implementation of the software, this research is mostly unable to evaluate Jape’s versatility with respect to logic system or interface design. On the other hand, by examining in detail students’ interactions with a specific implementation, it may be possible to indicate something of the potential strengths and pitfalls that might apply to other proof assistants.

3 The Approach to Data Collection

The first phase of the research made use of a naturalistic evaluation approach [12]. About 170 first-year computer science undergraduates were observed as they followed a course in introductory first-order logic in London. This was in order to understand how ItL Jape fitted into the learning context.

The students were observed working in small groups on course tasks - sometimes using pencil-and-paper and sometimes using computers. Use of ItL Jape was automatically logged, so that it would be possible to calculate which conjectures were attempted by each student, how long each student spent on a conjecture, and whether the duration and frequency of program usage is associated with better test results. Results and sample scripts from a number of tests given to the students as part of their degree course have been obtained – these include a pre-course mathematics test; a mid-course test on propositional logic; and an end-of-course test on predicate logic.

However, it should be noted that some students may be using Jape at home, for which timings would not be available. It has not been possible for this research to obtain more detailed records of the particular rules applied by students during proving - ideally, a complete record of each student's interactions with the program would be captured.

On the whole, most students ended up using ItL Jape for just two hours. However, since some did not use it at all (because of absence from those workshops) and some used it much more than most (because they accessed the program in their own time), there is a spread of usage time that will be useful in interpreting course outcome measures. Four students (individually or in pairs) were videoed as they used ItL Jape during the course, and they were interviewed about their experiences. Two students will be referred to below - Kusi and Lewis. They worked independently and were videoed for about three hours each.

4 Results from this Phase

At the time of writing, several findings are already evident from the analysis of the video data, including models of students' strategies and the ways in which these are developed.

4.1 Strategic knowledge

The first result is that some, if not all, students are making use of more or less readily identifiable strategies to help them construct proofs. Some of these strategies are *rule-specific*, such as «If there is an arrow as the principal operator in the conclusion, break up the conclusion using →I.».

Clearly not all students would, if asked, inevitably come up with the same linguistic formulation for this →I strategy, but it is necessary to attempt to capture, for the purposes of this paper, the flavour of a strategy that appears to account for very many student actions and that is repeatedly articulated by students in similar terms to these. Note that there are likely to be subtle variations. For example, it might be that algebraically-inclined students think of applying →E to P→Q as "substituting" P into the function-machine formula (a common metaphor in school mathematics) to get Q. →E is seen as a functional operator rather than as a relational rule. Or it could be that students are operating purely syntactically (something like «Given that P is a line of the proof and that P→Q is a line of the proof, the rule →E allows the line Q to be written.»). Or it could be that students are using an informal notion of existential proof (something like «If a proof of P exists, and a proof of P→Q exists, then that is sufficient to prove Q, justified by the axiom →E.»). Or it could be that they are using a notion of truth (something like «P→Q tells me that if P is true, then Q is true. But P *is* true, so Q is true. →E is the instruction to point this out.»).

Typically it seems that there are few indications from students' talk in this phase of the research of which variants might be being used. Therefore, in the next phase of the research, an interview-based study is planned that is hoped will yield more clues in this regard. By tackling selected proofs, and being asked to describe their interpretations of what they are doing, the aim is to explore more closely students' strategies for constructing proofs. However, one must then be aware of the likelihood of obtaining post hoc rationalisations rather than definitive cognitive mechanisms.

Some of these rule-specific strategies help students to choose between rules; for example, «If there is a choice between ∨E forwards and ∨I backwards, try ∨E first.». The student Kusi described this as the "precedence" of ∨E over ∨I. Other rule-specific strategies seem attempts to evaluate the success of a particular course of action before it is attempted; for example, «Proof by contradiction $\neg E \neg I$ may be useful in the case $\neg A \vdash B$ (where A and B represent complicated formulas) if A would be easier to break up than B.».

In contrast to rule-specific strategies, there appear to be strategies that might be called *global strategies*. Two important examples would be «When reasoning forwards, check if the lines produced are useful in obtaining the conclusion.»; and «When reasoning backwards, check if the lines produced are provable from the premises.». A conjectured strategy for which we have as yet seen no evidence is «If stuck on a conjecture with no premises, try to think of a previously-proved theorem that could be applied, so as to allow forward reasoning.».

4.2 Development of the Strategies

It could be argued that these strategies represent no more than an *ad hoc* collection of “rules of thumb” that demonstrate little of the deep understanding that an experienced logician might have, and show little regard for the circumstances in which they might fail. Moreover, it might be asserted, using a proof assistant program actually *encourages* a “blind”, purely syntactical, pattern-matching trial-and-error approach - for example, «Look for the ‘main symbol’ in the most complicated line. Find the same symbol in the list of rules. Try one of the matching rules. Undo the rule if the display doesn’t look right (criteria for which might include any large, unexpected increases in the length of the proof, the number of boxes, the number of gaps in the proof, the appearance of unfamiliar symbols); and try another.».

This argument is supported by video evidence that the student Lewis had some difficulty in recalling proof strategies that were apparently strongly founded a week before. The strategies that students are constructing in order to meet the short-term demands of a particular proof just do not appear to last, for some reason; and it is possible that the reason is superficial understanding. Could learning through trial-and-error of rules be a faster route to *ad hoc* tactics? It is also possible that even where understanding is thorough some sort of instructional intervention is desirable in order to support later recall.

On the other hand, while it has to be admitted that many of the simpler conjectures succumb to such a primitive trial-and-error strategy, it would have to be asked, then, where the more sophisticated strategies described above come from. There is little evidence in this study that they come from instruction; where students are using pattern-matching trial-and-error at the start of a set of exercises they often end up articulating or at least demonstrating the more sophisticated strategies. We would argue that students are *developing these strategies for themselves*, and, furthermore, they are developing them *precisely because pattern-matching trial-and-error is progressively found to be inadequate*.

Moreover, portraying these strategies as purely *mechanical* responses to a limited set of straightforward syntactical inputs seems undeserved. Not only do the strategies to be applied at a particular point in a proof have to be selected with care - particularly in conjectures involving negation or quantification - and not only do the strategies have to be adapted progressively as counter-examples are encountered, but the strategies also clearly incorporate expectations about what a proof should look like, about why a particular rule might be applicable in certain circumstances, about what might or might not be provable, and so on.

Finally, the role of the software in this is not so simple as encouraging pattern-matching trial-and-error. We did find evidence of the latter, and it was even noted by one student “He doesn’t know what he’s doing. … He’s proving them but he doesn’t know what’s going on.”; but early indications from the analysis are that success with mechanical methods was short-lived. Whether a more reflective approach is more or less likely when using software or using pencil-and-paper is at this stage uncertain; but what does seem clear is that for many students, using ItL Jape allowed them to consider many more examples than would be possible using pencil-and-paper (because the program takes on the task of drawing the proof) and it also guaranteed that inadequate proof attempts and incorrect rule applications were immediately challenged. It would therefore be reasonable to suggest that for those students adopting a reflective approach, the development of sophisticated proof strategies would be more effective when using the software than when using pencil-and-paper.

Indeed, several students expressed the view that, although it was not easy to work out from the program what the “difficult” rules did, or how or when they might be useful without at least some sort of initial “grasp” of the rules, the main advantage of ItL Jape was that it allowed experimentation in order to work out proof strategies. It would be interesting to be able to identify what precisely this “grasp” might be and why the feedback provided by the program was insufficient to provide it.

In order to probe the development of strategies more closely in the in-depth task-based interviews in the reflection study, students will be presented on paper with 5-10 partially-completed proofs from that topic; and asked about what the next steps might be. Students are to be asked to explain their rule decisions. Then these same conjectures will be tackled using ItL Jape. The logic conjectures have been devised to test hypotheses arising from the analysis of the video data. Where paper-and-pencil attempts are successful, this will illuminate interface issues; where paper-and-pencil attempts are invalid or stall, this will clarify the role of ItL Jape in enabling the further development of strategies.

4.3 Some Strategies are Harder than Others

It seems to be harder to construct useful strategies for some proof rules than others, given the particular implementation of the natural deduction system. For example, the →I and →E rules were apparently easily handled, in that the relevant strategies were constructed and recalled with little difficulty. The main difficulty with the ∧ rules was doubt about whether the rule “∧-E(L)” selected or removed the left-hand-side of the formula. But suspicion of the ∨E rule was widespread, to such a degree that students would often prefer to attempt ∨I backwards rather than attempt ∨E forwards. A typical case of this would occur in a conjecture such as $(P \vee Q) \vdash (Q \vee P)$.

In addition, it was noticeable for the student Kusi that ultimately even the most sophisticated syntactic rule-specific strategies had to be supplemented eventually by the use of semantics. For example, the conjecture $\forall x\ (P(x) \vee Q(x)) \vdash \forall x\ P(x) \vee \forall x\ Q(x)$ is easier to handle if one is able to interpret predicate notation (because time will not be wasted trying to prove it); and the conjecture $P \vee Q \vdash ((P \rightarrow \neg Q) \wedge (Q \rightarrow P)) \vee (Q \vee P)$ is easier to prove knowing that a large chunk of it can be safely ignored. But we do not yet know whether semantic checking (for usefulness, for provability and for simplification) is widespread. Certainly it was uncommon during students’ first two hours of using ItL Jape, in which they were largely dealing with →, ∧ and ∨ rules; and this is in spite of much instruction in the semantics of formal logic. The links between, on the one hand, the abstract symbols and rules of natural deduction, and, on the other, the extensive work on informal reasoning, truth-tables and set theoretic proofs of propositions, were just not apparent in the discussions of students working at the computer.

Whether semantic checking became more common as the conjectures become harder to prove is not yet clear, and this is something that needs to be explored further. The conjectures presented in the reflection study - and the order in which they are presented - need to address this issue.

4.4 Some Further Issues

It seemed to take many students some time to realise that unexpected results were often attributable to a line not being selected before a rule is applied. Also, several students suggested that they were confused at certain points in particular conjectures about whether the rule they were about to apply would operate "forwards" or "backwards"; it is not clear yet whether this is purely an interface issue or whether there lurks in this a deeper misconception about the nature of the formal reasoning process.

ItL Jape does not allow students to make the step "∧I forwards", instead forcing students to invoke ∧I only once the ∧ formula has been created (just as "→I backwards" is possible, but "→I forwards" is not). Although this may very well be better proof practice, it might be that students would find it easier to learn how to use ∧I if the interface allowed them to "and" two propositions together. There thus appears to be a tension between ease of learning *adequate* strategies in the first place and learning *good* strategies in the long run.

It was noticeable from the videotapes that students made few comments about the *nature* of the activity of constructing proofs. (One exception was a poignant remark from Lewis, who said, in response to a perhaps less-than-illuminating error message, "It doesn't look like English".) It is not clear whether this lack of comment is because students were absorbed so completely in the activity that the issue was not considered, or because the issue was so central to the activity itself that linguistic formulation of an opinion was too difficult.

In the reflection study, therefore, students will be asked what they have found most difficult about learning natural deduction, and what they found most difficult to do using Jape. It is also hoped to be able to find out more about students' perceptions of formal reasoning, by using talk-aloud protocols and by asking questions such as "How would you explain (in a few sentences) to someone who's never come across natural deduction before what sort of thing a natural deduction proof is?".

5 Further Research

In addition to the reflection study - which the work described in this paper was designed to inform - further data is being collected to explore the benefits of ItL Jape.

It is not yet clear to what extent the software is able to promote proof-generation skills that are transferable to pencil-and-paper, and so it is hoped that test scores can be compared with the level of Jape usage as measured by the logfiles. Another outcome measure is provided by a questionnaire given at the end of the course, in which students were asked about their opinions as to the value, interest and difficulty of the course, and their experiences of ItL Jape. The logfiles may also play an important role in indicating which conjectures were found most difficult.

Because control groups were unavailable, due to ethical and pragmatic considerations, data has also been collected that will enable an analysis of the comparative effects for different backgrounds of the student population. A survey given before the course aimed to elicit insights into students' prior experience, their concerns and their expectations of the course. A test of informal reasoning was also given. This data may indicate whether there are groups of students that gain particularly from the use of ItL Jape.

6 Conclusions

The videotape study has been extremely useful - firstly in enabling the identification of a number of possible strategies that students might be using to help them construct proofs; and secondly in informing the design of a detailed investigation of students' strategies in task-based interviews. Carefully-devised conjectures to be proved should enable a description of the most common proof strategies, including the priorities for different rules, and behaviours when the priorities fail.

A number of aspects should also become clearer. We should have greater insight into perceptions of the nature of proof and into the strengths and limitations of the interface. Most importantly, it is hoped to find out in more detail how ItL Jape can assist in the improvement of strategies. The workshop videotapes should also be crucial in providing examples of typical program usage that illustrate the account provided by the other sources of data; and in highlighting what is absent from that account.

References

[1] Barwise, J. & Etchemendy, J. "The Language of First-Order Logic", Cambridge University Press (1992)

[2] Bornat, R. & Sufrin, B. "Jape" (software)
Unix/Linux version available from http://www.comlab.ox.ac.uk/oucl/users/bernard.sufrin/jape.shtml
Apple Macintosh version available from http://www.dcs.qmw.ac.uk/~richard/jape

[3] Bornat, R. & Sufrin, B. "Animating the formal proof at the surface: the Jape proof calculator", Technical Report, Department of Computer Science, Queen Mary & Westfield College, London, ftp://ftp.dcs.qmw.ac.uk/jape/papers/ (1996)

[4] Dyckhoff, R. "Implementing a simple proof assistant", Workshop on Programming for Logic Teaching, Centre for Theoretical Computer Science, University of Leeds (1987)

[5] van Ditmarsch, H. "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* (1998)

[6] Fitch, F. B. "Symbolic Logic", Ronald Press, New York (1952)

[7] Fung, P. & O'Shea, T. "Using Software Tools to Learn Formal Reasoning: a first assessment", CITE Report No. 168, The Open University, UK (1992)

[8] Fung, P., O'Shea, T., Goldson, D., Reeves, S. & Bornat, R. "Computer Science Students' Perceptions of Learning Formal Reasoning Methods", *International Journal of Mathematical Education in Science and Technology*, 24 (5), 749-760 (1993)

[9] Fung, P., O'Shea, T., Goldson, D., Reeves, S. & Bornat, R. "Why computer science students find formal reasoning frightening", *Journal of Computer Assisted Learning*, 10, 240-250 (1994)

[10] Fung, P., O'Shea, T., Goldson, D., Reeves, S. & Bornat, R. "Computer tools to teach formal reasoning", *Computers in Education*, 27 (1), 59-69 (1996)

[11] Green, T. R. G. "Cognitive Dimensions of Notations" in Winder, R. & Sutcliffe, A. (Eds.) *People and Computers V*, p. 443-460, Cambridge University Press (1989)

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

[13] Kadoda, G. "Cognitive Dimensions Analysis of Theorem Provers", *Proceedings of the 3rd International Workshop on User Interface Design for Theorem Proving Systems*, France (1997)

[14] Scheines, R. & Seig, W. "The Carnegie Mellon Proof Tutor" in Boettcher, J. V. (Ed.) *101 Success Stories of Information Technology in Higher Education: The Joe Wyatt Challenge*, McGraw-Hill (1993)